iCSC
CERN
School of Computing

Welcome to **iCSC2008**, the third edition of the Inverted School, "*Where Students turn into Teachers*".

iCSC is an idea that was experimented for the first time in 2005.

The idea of **iCSC**s comes from the observation that at regular CSCs it is common to find someone in the room who knows more on a particular – usually advanced - topic than the lecturer. So why not to try and exploit this?

CSC2006 and CSC2007 students made proposals via an electronic discussion forum, from which a programme was designed by the CSC track coordinators.

This year's programme focuses on a timely, challenging topic: *Reconfigurable High-Performance Computing*. This is the first time at CERN that a complete ten-hour programme is dedicated to the subject. It is complemented by a short session on two special topics.

I would like to thank all those who developed proposals and those actually lecturing. This is their school and I am confident all will go very well. As this is only the third edition, do not hesitate to comment and advise us on how to improve it.

Enjoy the school.

François Fluckiger
Director of the CERN School of Computing

Disclaimer: Opinions expressed in this booklet are those of the lecturers and do not necessarily reflect those of CERN

iCSC2008 Programme Overview

The programme has one major theme lasting 2 days: Towards Reconfigurable High-Performance Computing. It is followed by a short session on special topics. The programme results from selected proposals developed by students through an electronic forum.

Towards Reconfigurable High-Performance Computing			Special topics		
4 hours	- L1 Basics - L5 Multicores at work: The CELL Processor - L7 Reconfigurable HPC I - Introduction - L10 Summary: Hybrid Platforms, Hybrid Programming?	**Iris Christadler** Leibniz Super-Computing Centre - Germany	*1 hour*	- L1 Overview of advanced aspects of data analysis software	**Alfio Lazzaro** University of Milan and INFN, Milan - Italy
3 hours	- L6 Platforms III - Programmable Logic - L8 Reconfigurable HPC II - HW Design Methodology, Theory & Tools - L9 Advanced and Emerging Parallel Programming Paradigms	**Manfred Muecke** University of Vienna - Austria	*1.5 hours*	- L2 Scalable Image and Video coding	**Jose Dana Perez** CERN, Geneva
3 hours	- L2 Multicore Architectures - L3 Platforms I: Advanced Architectural Features - L4 Platforms II: Special-Purpose Accelerators	**Andrzej Nowak** CERN, Geneva			
10 hours			*2.5 hours*		
Monday 3 & Tuesday 4 March 2008 10:15 - 17:30 Bld 31- 3rd floor IT Amphitheatre			**Wednesday 5 March 2008** 09:00 - 12:00 Bld 31- 3rd floor IT Amphitheatre		

iCSC2008 Schedule

Monday 3 March 2008 IT Auditorium, Bld 31		Tuesday 4 March 2008 IT Auditorium, Bld 31		Wednesday 5 March 2008 IT Auditorium, Bld 31	
Towards Reconfigurable High-Performance Computing		**Towards Reconfigurable High-Performance Computing**		**Special topics**	
		09:00-09:55	Platforms III - Programmable Logic **Manfred Muecke**	09:00-10:00	Overview of advanced aspects of data analysis software **Alfio Lazzaro**
10:15-10:30	School opening	10:05-11:00	Reconfigurable HPC I - Introduction **Iris Christadler**	10:00-10:30	Coffee
10:30-11:25	Basics **Iris Christadler**			10:30-12:00	Scalable Image and Video coding **Jose Dana Perez**
		11:00-11:30	Coffee Break		
11:35-12:30	Multicore Architectures **Andrzej Nowak**	11:30-12:25	Reconfigurable HPC II - HW Design Methodology, Theory & Tools **Manfred Muecke**	12:00	Adjourn
12:30-14:00	Lunch		Lunch		
14:00-14:55	Platforms I: Advanced Architectural Features **Andrzej Nowak**	14:00-14:55	Advanced and Emerging Parallel Programming Paradigms **Manfred Muecke**		
15:05-16:00	Platforms II: Special-Purpose Accelerators **Andrzej Nowak**	15:05-16:00	Summary: Hybrid Platforms, Hybrid Programming? **Iris Christadler**		
16:00-16:30	Coffee Break	16:10-16:25	Transition to Data Analysis topic **Alfio Lazzaro**		
16:30-17:25	Multicores at work: The CELL Processor **Iris Christadler**	16:30	Adjourn		
17:30	Adjourn				
19:00	*Dinner with iCSC2008 lecturers (TBC) (*)*				

(*): as well as with former CSC2007 participants who registered for the dinner

iCSC 2008 Lecturer Biographies

Iris CHRISTADLER	Leibniz Supercomputing Centre - Germany
	I am a member of the grid and the user support teams at LRZ. We operate a 9728 core SGI Altix 4700 and a heterogeneous Linux-Cluster with 800 cores. Together with the MPG a cluster with 800 cores and 350 TB disc space is part of LCG as a Tier-2 Center. In addition to LCG, LRZ is also a member of the DEISA/eDEISA grid infrastructure. My work involves porting of applications to different platforms, optimization, debugging and grid enablement, which is currently limited to DEISA but should be expanded to LCG as well.
Jose Miguel DANA PEREZ	**CERN, Geneva - Switzerland**
	Jose M. Dana studied at University of Almeria (Spain) where he obtained a M.Sc. degree in Computer Science and worked for the "Computer Architecture and Electronics" department for the last two years of his degree. Moreover, he is member of the "Supercomputing: Algorithms" research group of his University since 2004. During his collaboration he has written several papers about scalable image and video coding. He was a CERN Summer Student in 2005 and he worked in compiler optimization related tasks (in CERN openlab). In October 2006 he re-joined CERN openlab as a Fellow working this time in Grid deployment and virtualization subjects. Right now, he is combining his work in CERN openlab with his PhD studies.
Alfio LAZZARO	**University of Milan and INFN, Milan - Italy**
	I am a postdoc in University of Milan, department of Physics, and I'm a member of the BaBar Collaboration and recently a new member of Atlas Collaboration. BaBar is an experiment of High Energy Physics running at SLAC, Menlo Park, CA. Currently I'm the Physics Software Coordinator of the Collaboration. The activity is finalized to develop and maintain the code used for event reconstruction and data analysis. My research is on physics analysis and software used for data analysis. In particular, I study the charmless decays of B mesons to final states containing an eta or eta' meson. For all these studies I have developed a fitting program (maximum likelihood fits) in C++ language on Linux/UNIX platform. This program, called MiFit, uses ROOT and RooFit classes. I use several other techniques, like Fisher Discriminant, Neural Network, Decision Tree.

Manfred MUECKE **University of Vienna - Austria**

I studied electrical engineering with emphasis on telecommunication and computer architectures. I am interested in design and implementation of languages and compilers to enable more efficient description and synthesis of complex FPGA-based computing systems.

I wrote my PhD thesis at CERN, focusing on design methodologies for digital signal processing on FPGAs. All LHC experiments use FPGAs in their data acquisition systems at medium trigger levels. It was therefore a most exciting work environment. Currently, I am working on the optimization of molecular dynamics simulations.

Andrzej NOWAK **CERN, Geneva - Switzerland**

Andrzej Nowak has been working at CERN openlab, a partnership between CERN and the industry (Intel, HP, Oracle), since 2007. His early research concerned operating systems security, mobile systems security, and wireless technologies. During his studies in 2005 and 2006, Andrzej worked at Intel, where he researched custom performance optimizations of the Linux kernel and took part in developing one of the first 802.16e (WiMax Mobile) wireless MAN networking standard implementations. Soon after obtaining his diploma, he joined openlab in January 2007. Andrzej deals mostly with multi- and many-core architectures and parallel processing. Another significant area of his work is platform optimization and performance assessment.

Towards Reconfigurable High-Performance Computing

iCSC2008: Towards Reconfigurable High-Performance Computing

Lecturers:

Iris Christadler - Leibniz Supercomputing Centre - Germany
Manfred Muecke - University of Vienna - Austria
Andrzej Nowak - CERN, Geneva

Moore's law still holds and provides us with unprecedented device integration, resulting in abundant logic resources even on commodity computing platforms. However, computing has failed to take advantage of this gift and **increase in computing performance is constantly lagging behind the increase in logic resource**s.

In short: semiconductor technology has overtaken chip designers, computer scientists and programmers on the right. This is strongly felt in high-performance computing (HPC) where **most applications can no longer take advantage of the many cores** provided by current supercomputers. Stalling cores nevertheless take up energy and in HPC power consumption is becoming an important issue. Among the more promising future solutions to these problems is **reconfigurable computing** (computing on flexible fabrics). In this series of lectures, we want to explore the reasoning behind reconfigurable HPC, its prospects, implications and issues.

We will discuss **different hardware architectures** and their respective requirements and implications to the model of computation applied (or the mismatch thereof). We will **compare** them, **sketch their potentia**l for HPC and introduce a more **unified view o**n them.

We will show that the dominant problem in HPC is not hardware but software. Especially the fact that **our programming models do not match current technology** is the root of much inefficiency.

Reconfigurable Computing is **challenging**, because it asks many questions at the same time. But it is also most **rewarding**, because it forces us to rethink the way we design computers, interconnects, compilers and applications.

The lectures aim at qualifying students to understand **where and why** reconfigurable computing can be expected to have a **considerable impa**ct on tomorrows high-performance computing landscape, and **where not**.

A few questions

- *What will tomorrow's supercomputers look like?*
- *What are the expected changes in commodity PC hardware and why should we care about them?*
- *What should I do to use tomorrow's supercomputers efficiently?*
- *How to port your application to GPUs (without porting it)?.*
- *Why thinking parallel will make all the difference?*
- *Shall we think in local or global address spaces?*
- *Do we need data stream processing to cope efficiently with many-core CPUs?*
- *What to think of the new programming languages … Fortress, Chapel, X10, UPC, Co-Array Fortran?*
- *Do you need to learn VHDL when using FPGAs?*
- *How to define an FPGA's peak performance? (and how to cheat doing so?)*
- *How can FPGAs running at 100MHz outperform CPUs running at 3GHz*
- *Does C-to-Hardware work?*
- *Have you ever wondered what the acronyms DEISA/PRACE/HPCS mean?*
- *Can your playstation save the world?*
- *Why should supercomputers care about the climate change?*
- *Do you still believe that Roadrunner is just a bird and Maxwell is a Scottish physicist?*

All the answers in the Computational Intelligence at iCSC

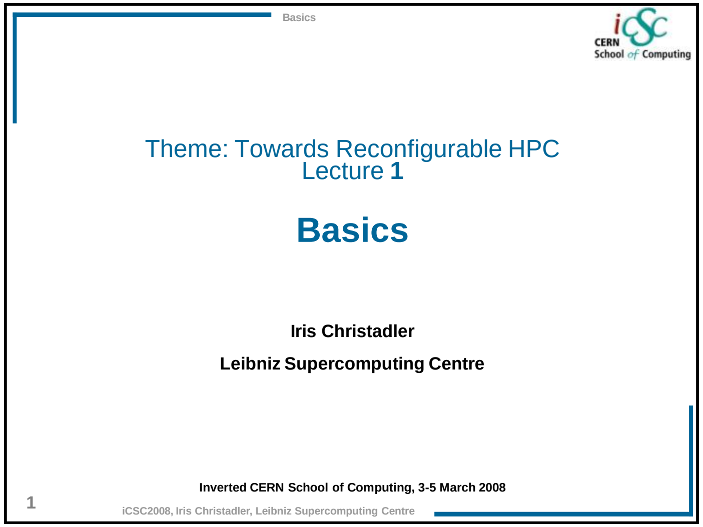
Overview

Slot	Lecture	Description	Lecturer
		Monday 3 March 2008	
10:15-10:30		School opening - **Introduction**	
10:30-11:25	Lecture 1	**Basics**	Iris Christadler
11:35-12:30	Lecture 2	**Multicore Architectures**	Andrzej Nowak
12:30-14:00		Lunch	
14:00-14:55	Lecture 3	**Platforms I: Advanced Architectural Features**	Andrzej Nowak
15:05-16:00	Lecture 4	**Platforms II: Special-Purpose Accelerators**	Andrzej Nowak
16:00-16:30		Coffee break	
16:30-17:25	Lecture 5	**Multicores at work: The CELL Processor**	Iris Christadler
17:30		Adjourn	
		Tuesday 4 March 2008	
09:00-09:55	Lecture 6	**Platforms III - Programmable Logic**	Manfred Muecke
10:05-11:00	Lecture 7	**Reconfigurable HPC I - Introduction**	Iris Christadler
11:00-11:30		Coffee break	
11:30-12:25	Lecture 8	**Reconfigurable HPC II - HW Design Methodology, Theory & Tools**	Manfred Muecke
12:30-14:00		Lunch	
14:00-14:55	Lecture 9	**Advanced and Emerging Parallel Programming Paradigms**	Manfred Muecke
15:05-16:00	Lecture 10	**Summary: Hybrid Platforms, Hybrid Programming?**	Iris Christadler
16:10-16:25	Theme closing	**Transition between HPC and Data Analysis themes: Using HPC concepts in data analysis software** *(short session)*	Alfio Lazzaro
16:30		Adjourn	

LECTURE 1

Basics

Monday 3 March 2008			
10:30 11:25	Lecture 1	This lecture will give an overview of the state-of-the-art, developments and research topics in High-Performance Computing. Important topics: • HPC applications • HPC platforms • HPC users • Preparations for the Petascale Area The motivation of alternative architectures to commodity platforms for HPC users will be addressed. **Audience** The lecture targets all participants with interest in HPC. **Pre-requisite** No prerequisite is necessary.The Data Analysis Process	**Iris Christadler**



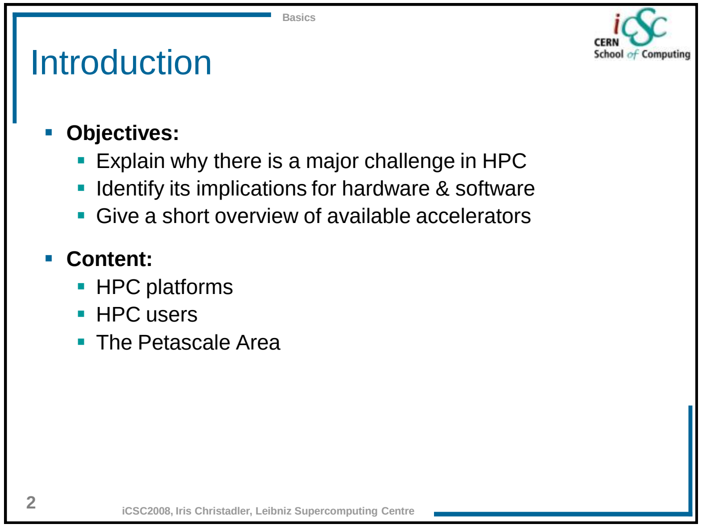
Theme: Towards Reconfigurable HPC
Lecture **1**

Basics

Iris Christadler

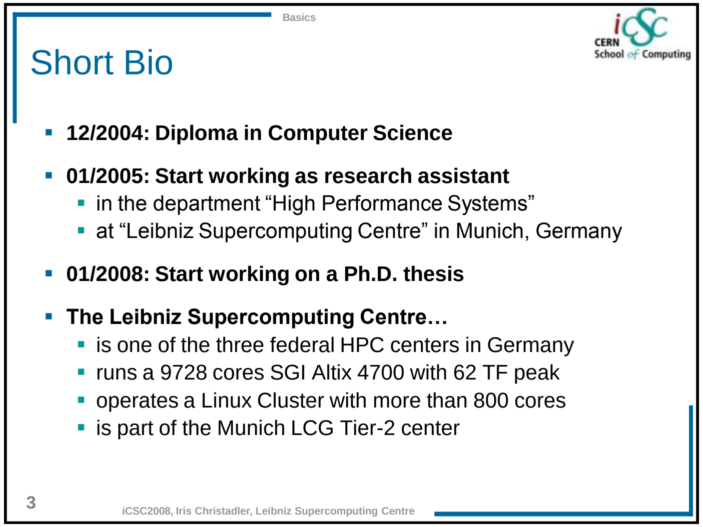
Leibniz Supercomputing Centre

Inverted CERN School of Computing, 3-5 March 2008



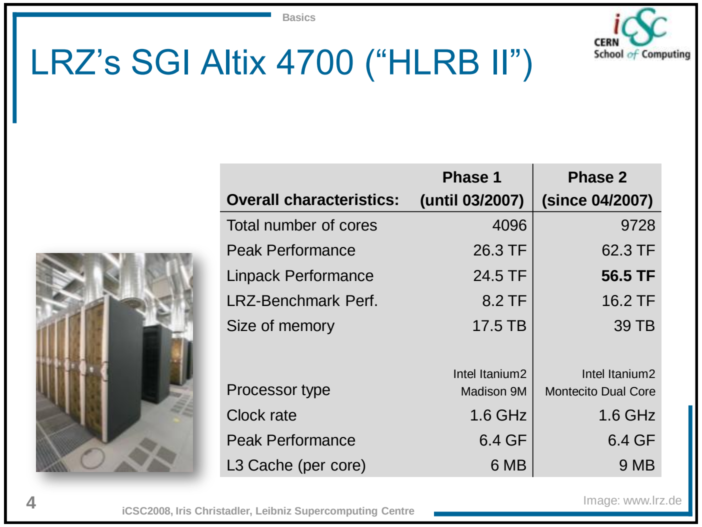
Introduction

- **Objectives:**
  - Explain why there is a major challenge in HPC
  - Identify its implications for hardware & software
  - Give a short overview of available accelerators
- **Content:**
  - HPC platforms
  - HPC users
  - The Petascale Area



Short Bio

- **12/2004: Diploma in Computer Science**
- **01/2005: Start working as research assistant**
  - in the department “High Performance Systems”
  - at “Leibniz Supercomputing Centre” in Munich, Germany
- **01/2008: Start working on a Ph.D. thesis**
- **The Leibniz Supercomputing Centre…**
  - is one of the three federal HPC centers in Germany
  - runs a 9728 cores SGI Altix 4700 with 62 TF peak
  - operates a Linux Cluster with more than 800 cores
  - is part of the Munich LCG Tier-2 center



LRZ’s SGI Altix 4700 (“HLRB II”)

Overall characteristics:	**Phase 1** **(until 03/2007)**	**Phase 2** **(since 04/2007)**
Total number of cores	4096	9728
Peak Performance	26.3 TF	62.3 TF
Linpack Performance	24.5 TF	**56.5 TF**
LRZ-Benchmark Perf.	8.2 TF	16.2 TF
Size of memory	17.5 TB	39 TB
Processor type	Intel Itanium2 Madison 9M	Intel Itanium2 Montecito Dual Core
Clock rate	1.6 GHz	1.6 GHz
Peak Performance	6.4 GF	6.4 GF
L3 Cache (per core)	6 MB	9 MB

Image: www.lrz.de

Juelich’s Blue Gene/P (JUGENE)

- **currently the fastest Computer in Europe**
- **ranked 2 in the top500 (11/2007)**

Overall characteristics	JUGENE
Total number of cores	65536
Peak Performance	222.8 TF
Linpack Performance	**167.3 TF**
Size of memory	32 TByte
Processor type	PowerPC 450
Clock rate	850 MHz
Peak Performance	3.4 GF

Image: www.fz-juelich.de

An introduction to…

HIGH PERFORMANCE COMPUTING

Definition

- **High Performance Computing (HPC):** [http://en.wikipedia.org/wiki/High-performance_computing]
  - use of parallel supercomputers or
  - compute clusters
    - usually mass-produced processors
    - linked together in a single system
    - with commercially available interconnects
  - systems above the teraflops-region
  - used for scientific research
- **High Productivity Computing** [http://en.wikipedia.org/wiki/High-performance_computing]

  “The more current and evolving definition of HPC refers to High Productivity Computing, and reflects the purpose and use model of the myriad of existing and evolving architectures, and the supporting ecosystem of software, middleware, storage, networking and tools behind the next generation of applications.”

top500

- **consists of the top500 most powerful computing systems**
- **updated twice a year**
  - in June at the European ISC
  - in November at the US SC
- **www.top500.org**

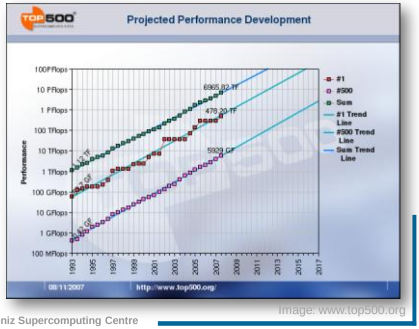
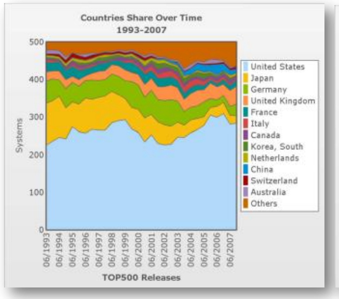
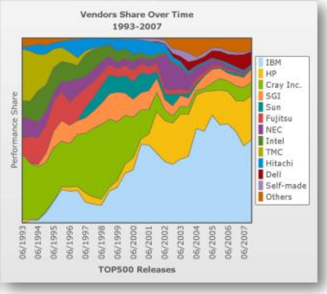


Image: www.top500.org

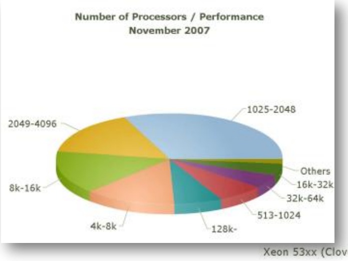
top500 Statistics



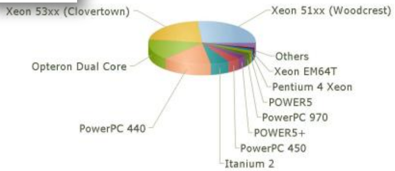


Images: www.top500.org

top500 Statistics cont'd



Processor Generation / Performance
November 2007



Images: www.top500.org

The actual top5 …

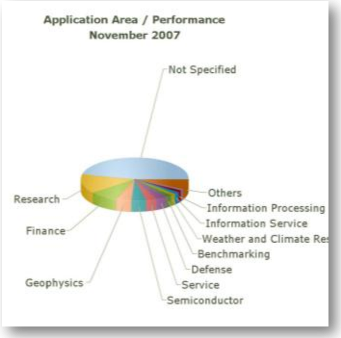
No	Site	System	#Proc	Peak Perf	Linpack	Efficen.
1	LLNL, USA	IBM Blue Gene/L	212992	596378	478200	80%
2	FZJ, Germany	IBM Blue Gene/P	65536	222822	167300	75%
3	NMCAC, USA	SGI Altix ICE 8200	14336	172032	126900	74%
4	India	Cluster Platform, HP	14240	170880	117900	69%
5	Sweden	Cluster Platform, HP	13728	146430	102800	70%
…	…	…	…	…	…	…
15	LRZ, Germany	SGI Altix 4700	9728	62259	56520	91%
16	Japan	Sun + ClearSpeed	11664	102021	56430	55%
17	EPCC, UK	HECToR, Cray XT4	11328	63436	54648	86%

Having a closer look at the LRZ performance figures

MORE STATISTICS

Who is using supercomputers?

Figures taken from LRZ’s SGI Altix (9728 cores)



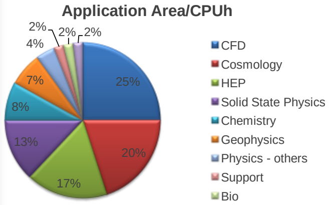


Image: www.top500.org

Image: LRZ

How are supercomputers used?

Figures taken from LRZ’s SGI Altix (9728 cores)

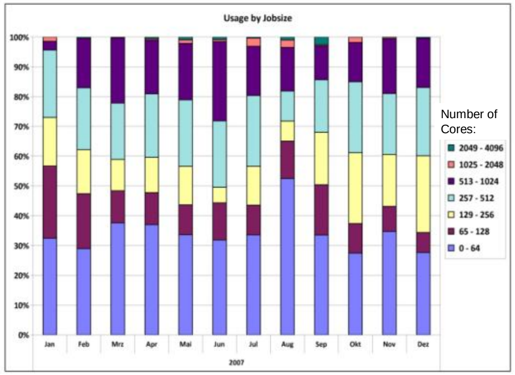


Image: LRZ

How are supercomputers used?

Figures taken from LRZ’s SGI Altix (9728 cores)

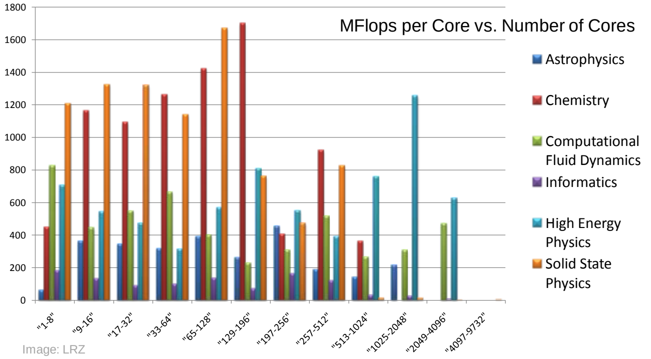
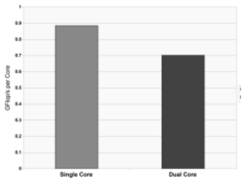


Image: LRZ

From single to dual core

Figures taken from LRZ’s SGI Altix



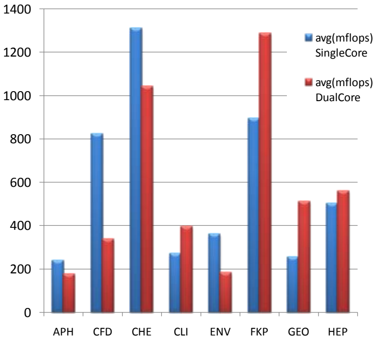
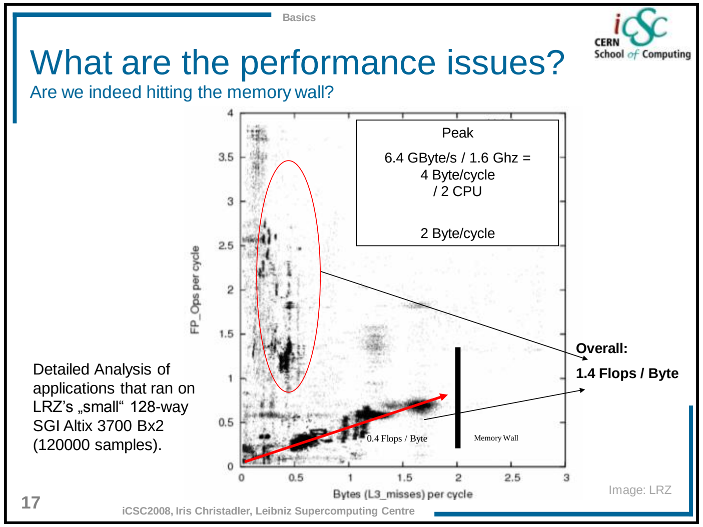


Image: LRZ (2007)

Image: LRZ (2008)



What are the performance issues?

Are we indeed hitting the memory wall?

Overall:

1.4 Flops / Byte

Detailed Analysis of applications that ran on LRZ's „small" 128-way SGI Altix 3700 Bx2 (120000 samples).

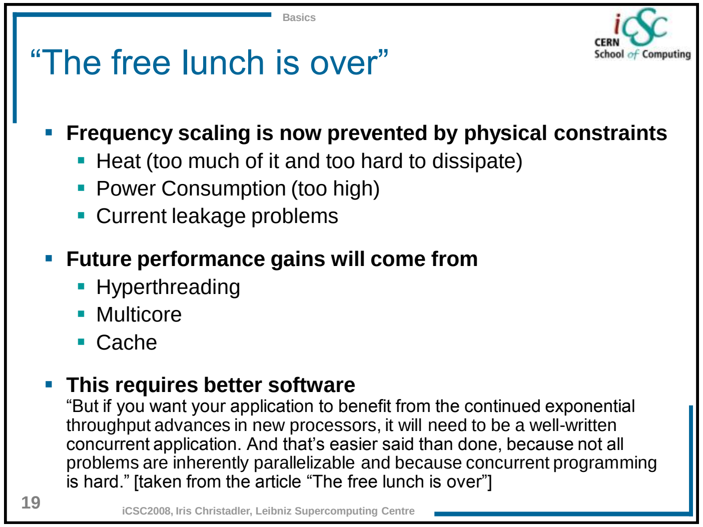
Image: LRZ

The article from Herb Sutter

„THE FREE LUNCH IS OVER"

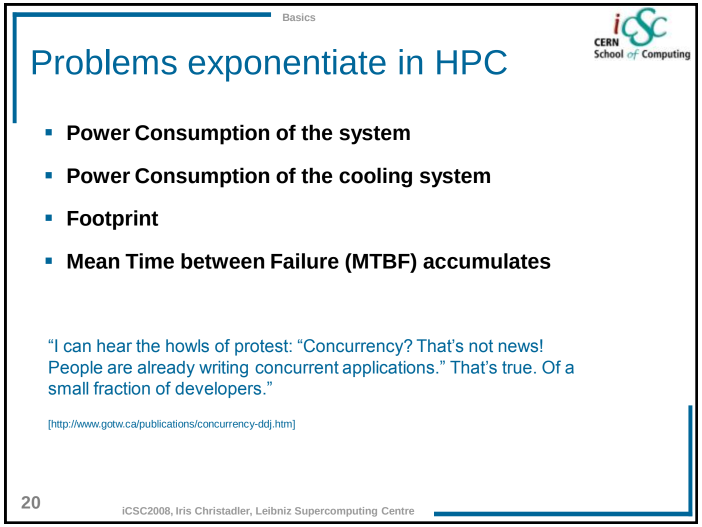
[http://www.gotw.ca/publications/concurrency-ddj.htm]

Image: Guillaume Dargaud



"The free lunch is over"

- **Frequency scaling is now prevented by physical constraints**
  - Heat (too much of it and too hard to dissipate)
  - Power Consumption (too high)
  - Current leakage problems
- **Future performance gains will come from**
  - Hyperthreading
  - Multicore
  - Cache
- **This requires better software**
  "But if you want your application to benefit from the continued exponential throughput advances in new processors, it will need to be a well-written concurrent application. And that's easier said than done, because not all problems are inherently parallelizable and because concurrent programming is hard." [taken from the article "The free lunch is over"]



Problems exponentiate in HPC

- **Power Consumption of the system**
- **Power Consumption of the cooling system**
- **Footprint**
- **Mean Time between Failure (MTBF) accumulates**

"I can hear the howls of protest: "Concurrency? That's not news! People are already writing concurrent applications." That's true. Of a small fraction of developers."

[http://www.gotw.ca/publications/concurrency-ddj.htm]

CERN School of Computing

PETASCALING

Sounds interesting? …. So let's start with…

CERN School of Computing

Politics & Acronyms

- **Distributed European Infrastructure for Supercomputing Applications (DEISA)**
  [www.deisa.org]
  - Consortium of the leading National Supercomputing Centers
  - Integration with Grid-Middleware (Unicore & Globus)
  - Global File system (GPFS)
- **DEISA Extreme Computing Initiative (DECI)**
  [www.deisa.org/applications/deci.php]
  - enable a number of "grand challenge" applications
  - offers European-wide access to extremely nice computers ☺
  - very good hands-on trainings

CERN School of Computing

Politics & Acronyms cont'd

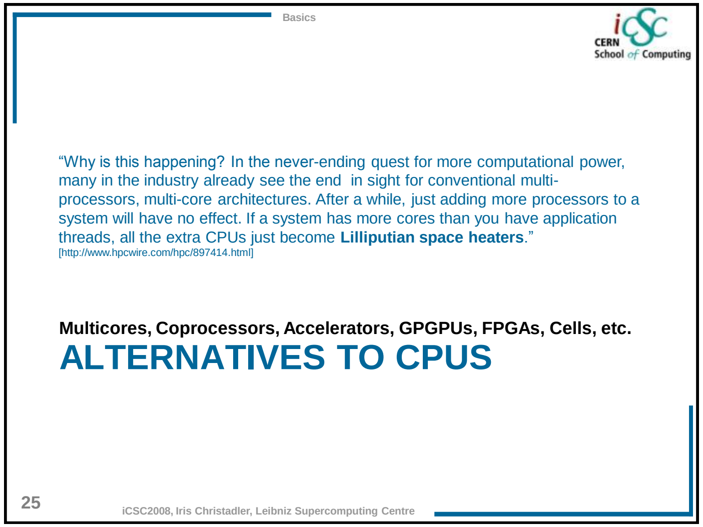
- **High Productivity Computing Systems (HPCS)**
  [www.highproductivity.org]
  - funded by DARPA (DoD)
  - Goal: Support industry to manufacture and deliver a petaflop-class computer that is substantially easier to program and use than the computers the industry is evolving toward today.
- **Partnership for Advanced Computing in Europe (PRACE)**
  [www.prace-project.eu]
  - FP7 funded, 01/08-12/09, 20M EURs
  - 16 entities from 14 countries
  - 80% of the European Linpack Perf. from the top500
  - Goal: Establishment of a European HPC infrastructure

CERN School of Computing

Petascaling…

- **is on it's way both in Europe and in the US**
  - First US petascale system estimated for 2008 (called "Roadrunner"):
    - Hybrid Opteron-Cell system
    - use Cell BE as "accelerators"
    - built at Los Alamos National Laboratory
  - A European petascale supercomputer is expected in 2010
- **Are we ready for petascale?**
- **Are our codes ready for petascale?**

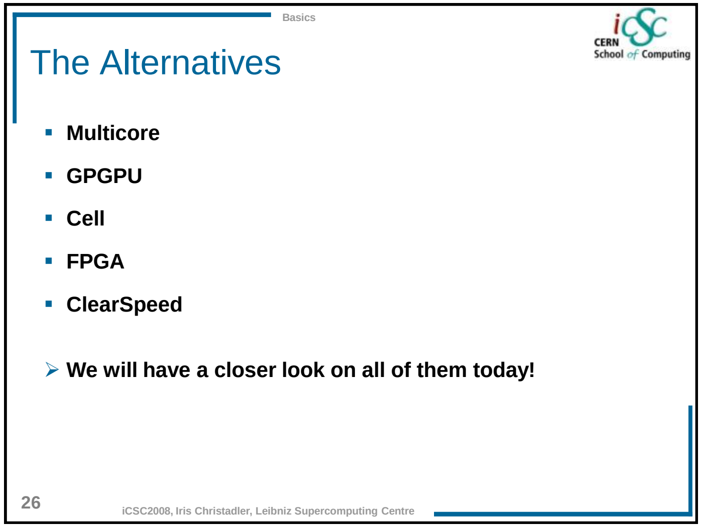
➢ **Strong trend towards "accelerators" in HPC**



"Why is this happening? In the never-ending quest for more computational power, many in the industry already see the end in sight for conventional multi-processors, multi-core architectures. After a while, just adding more processors to a system will have no effect. If a system has more cores than you have application threads, all the extra CPUs just become **Lilliputian space heaters**."
[http://www.hpcwire.com/hpc/897414.html]

Multicores, Coprocessors, Accelerators, GPGPUs, FPGAs, Cells, etc.

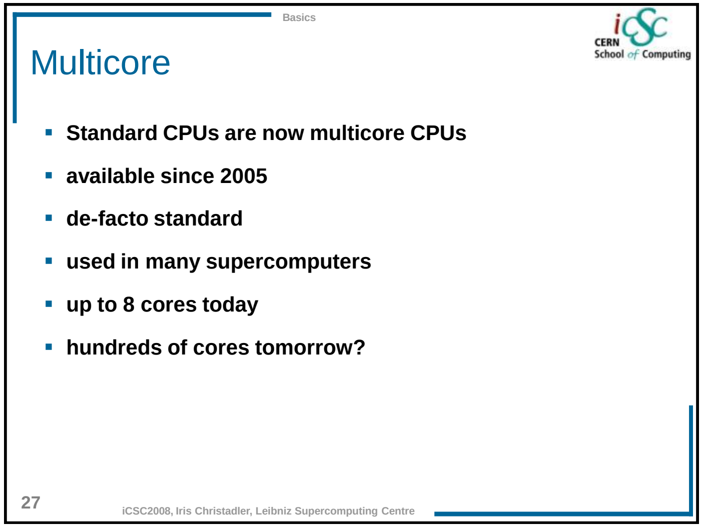
ALTERNATIVES TO CPUS



The Alternatives

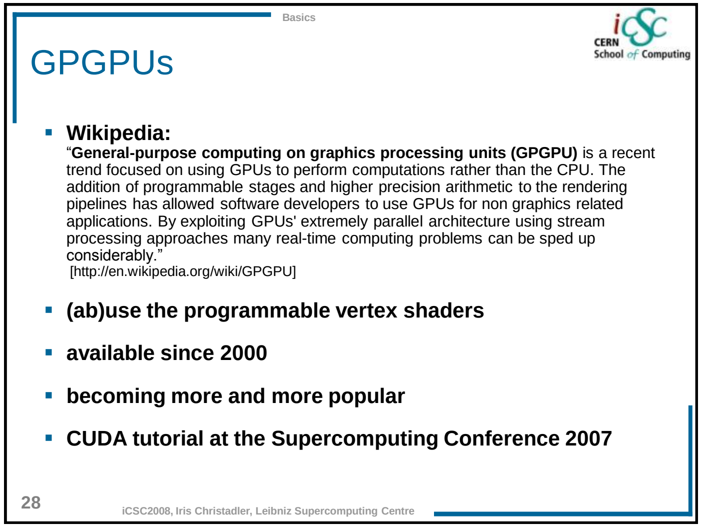
- **Multicore**
- **GPGPU**
- **Cell**
- **FPGA**
- **ClearSpeed**

➢ **We will have a closer look on all of them today!**



Multicore

- **Standard CPUs are now multicore CPUs**
- **available since 2005**
- **de-facto standard**
- **used in many supercomputers**
- **up to 8 cores today**
- **hundreds of cores tomorrow?**



GPGPUs

- **Wikipedia:**
  "**General-purpose computing on graphics processing units (GPGPU)** is a recent trend focused on using GPUs to perform computations rather than the CPU. The addition of programmable stages and higher precision arithmetic to the rendering pipelines has allowed software developers to use GPUs for non graphics related applications. By exploiting GPUs' extremely parallel architecture using stream processing approaches many real-time computing problems can be sped up considerably."
  [http://en.wikipedia.org/wiki/GPGPU]
- **(ab)use the programmable vertex shaders**
- **available since 2000**
- **becoming more and more popular**
- **CUDA tutorial at the Supercomputing Conference 2007**

Cells

- **Wikipedia:**
  “Cell is **a microprocessor architecture** jointly developed by Sony Computer Entertainment, Toshiba, and IBM, an alliance known as "STI". The architectural design and first implementation were carried out at the STI Design Center in Austin, Texas over a four-year period beginning March 2001 on a budget reported by IBM as approaching US$400 million.
  Cell combines a general-purpose Power Architecture core of modest performance with streamlined coprocessing elements which greatly accelerate multimedia and vector processing applications, as well as many other forms of dedicated computation.”
  [http://en.wikipedia.org/wiki/Cell_(microprocessor)]
- **available since 2005**
- **many research CELL clusters**
- **a hybrid Opteron-Cell cluster will become the first petaflop system**

FPGAs

- **Wikipedia:**
  “A **field-programmable gate array** is a semiconductor device containing programmable logic components called "logic blocks", and programmable interconnects. Logic blocks can be programmed to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or simple mathematical functions. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.”
  [http://en.wikipedia.org/wiki/Field-programmable_gate_array]
- **invented 1984**
- **used heavily in embedded and real-time systems**
- **used in supercomputers like Cray XD1, SGI RASC Blades**
- **Programmability!**

ClearSpeed Boards

- **Advertisement claims:**
  - “World’s highest performance processor” (80.64 GF per board)
  - “World’s highest performance per watt” (2 GF/Watt)
- **only accelerated platform in the current top500**
- **Linpack results:**

System Specification	Linpack Result
4 nodes (16GB) w/o Advance boards	136.0 GF
4 nodes (16GB) w/ 2 x Advance boards each	364.2 GF
1 node (16GB) w/o Advance boards	34.0 GF
1 node (16GB) w/ 2 x Advance boards	90.1 GF

[http://www.clearspeed.com/acceleration/performance/benchmarks/]

Overview of available Accelerators

	GPU	FPGA	ClearSpeed	Cell
Price	$	$$-$$$	$$$	$-$$$
Power	high	low	medium	medium
Good at	graphics, 32bit	integer	64bit	graphics, 32bit
64bit?	no	yes	yes	yes
64bit Perf.	-	low	high	high in 2008
IEEE-754	no	no	yes	expensive

Source: White Paper from HP
[http://www.hp.com/techservers/hpccn/hpccollaboration/ADCatalyst/downloads/accelerators.pdf]

Our answer to the challenge in HPC?

RECONFIGURABLE COMPUTING

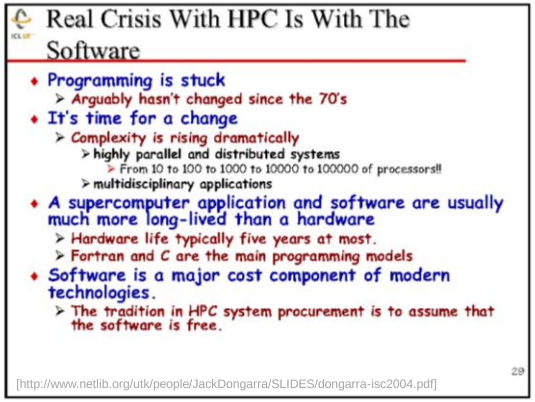
Definition

- **Reconfigurable Computing (RC)**
  "Reconfigurable computing is a computing paradigm combining some of the flexibility of software with the high performance of hardware by processing with very flexible high speed computing fabrics like FPGAs."
- **Concept exists since 1960s (Paper by Gerald Estrin)**
  "Unfortunately this idea was far ahead of its time in needed electronic technology."
- **Rennaisance in the 80s/90s**
  "The world's first commercial reconfigurable computer, the Algotronix CHS2X4, was completed in 1991. It was not a commercial success."
- **Reconfigurable HPC (RHPC)**
  "Currently there are a number of vendors with commercially available reconfigurable computers aimed at the high performance computing market."
  [http://en.wikipedia.org/wiki/Reconfigurable_computing]
- **Pros and Cons discussed during the next two days!**

Alternative Platforms need …

ALTERNATIVE PROGRAMMING MODELS

My favorite "Dongarra Slide"



Real Crisis With HPC Is With The Software

- Programming is stuck
  - Arguably hasn't changed since the 70's
- It's time for a change
  - Complexity is rising dramatically
    - highly parallel and distributed systems
      - From 10 to 100 to 1000 to 10000 to 100000 of processors!!
    - multidisciplinary applications
- A supercomputer application and software are usually much more long-lived than a hardware
  - Hardware life typically five years at most.
  - Fortran and C are the main programming models
- Software is a major cost component of modern technologies.
  - The tradition in HPC system procurement is to assume that the software is free.

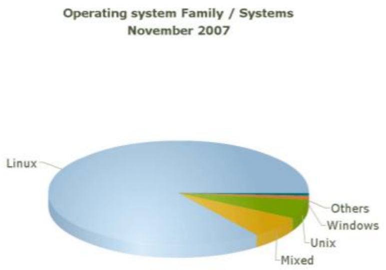
29

[http://www.netlib.org/utk/people/JackDongarra/SLIDES/dongarra-isc2004.pdf]

Parallel Programming Languages

- **Parallel Global Address Space (PGAS) languages**
  Unified Parallel C (UPC), Co-Array Fortran (CAF), Titanium
- **HPCS languages**
  Fortress, Chapel, X10
- **Data-stream languages**
  Brook, CUDA, RapidMind
- **Others**
  Ct, STM, …

➢ **All of them will be introduced during our lectures!**



Last but not least, the (scary) Microsoft ideas:

“HOW TO MAKE MONEY IN HPC”

[http://www.eetimes.com/showArticle.jhtml?articleID=201200019]

Image: www.top500.org

Conclusion

- **Expected major change in the basic hardware architecture**

➢ **Lead to the necessity of new programming models**

- **Ideas exists but are mostly research projects**
- **With new programming models reconfigurable computing is becoming more and more interesting**
- **This workshop will give an introduction of alternatives that are currently investigated**
- **You, as programmers, have to choose and let the dream of easy-to-use, easy-to-change, errorless code “come true”**

References and further reading

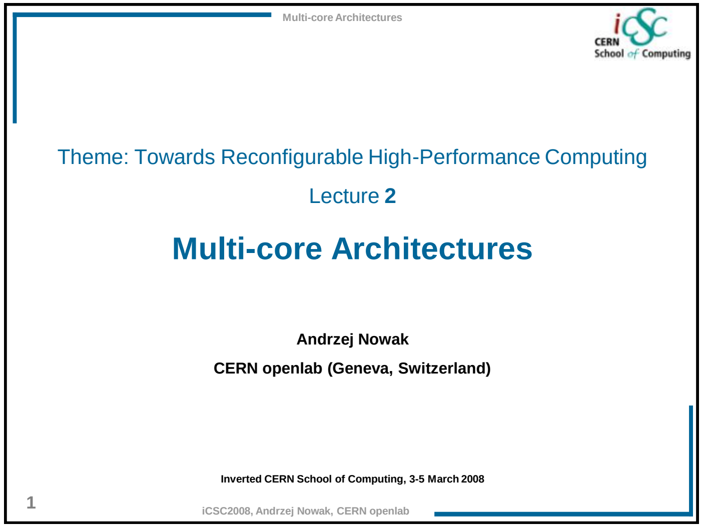
- **The Landscape of Parallel Computing Research: A View from Berkeley**
  http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf
- **The Free Lunch Is Over**
  http://www.gotw.ca/publications/concurrency-ddj.htm
- **Accelerators For HPC, White Paper, HP**
  http://www.hp.com/techservers/hpccn/hpccollaboration/ADCatalyst/downloads/accelerators.pdf
- **M'soft: Parallel programming model 10 years off**
  http://www.eetimes.com/showArticle.jhtml?articleID=201200019

CERN
School of Computing

LECTURE 2

Multicore Architectures

Monday 3 March 2008			
11:35 12:30	Lecture 2	This lecture will explain why multi-core architectures have become so popular and why parallelism is such a good bet for the near future. Important topics: • Scalability and parallelism • General multi-core architecture characteristics • Multi-core caveats and trade-offs In particular, the changes in computing landscape will be discussed, as well as the impact that modern hardware has on software. **Audience - Pre-requisite** Listeners don't need to have advanced knowledge about parallel computing.	**Andrzej Nowak**



Theme: Towards Reconfigurable High-Performance Computing

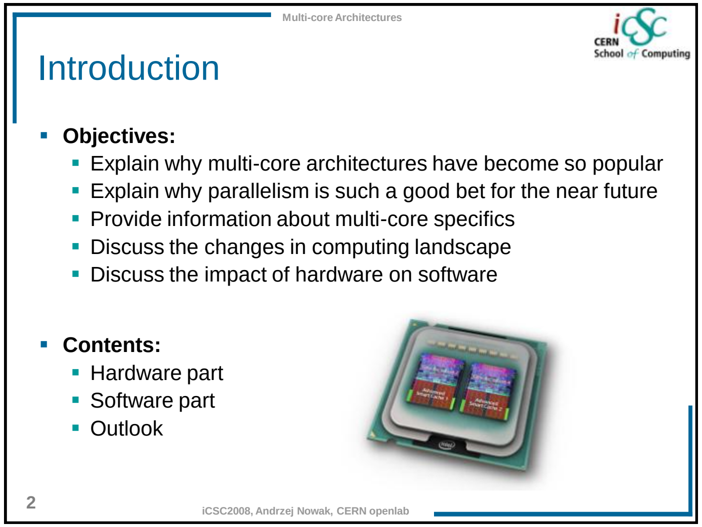
Lecture **2**

Multi-core Architectures

Andrzej Nowak

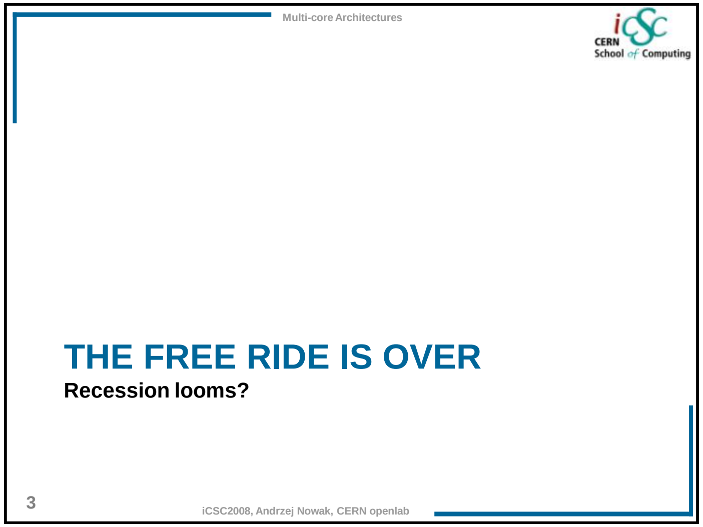
CERN openlab (Geneva, Switzerland)

Inverted CERN School of Computing, 3-5 March 2008



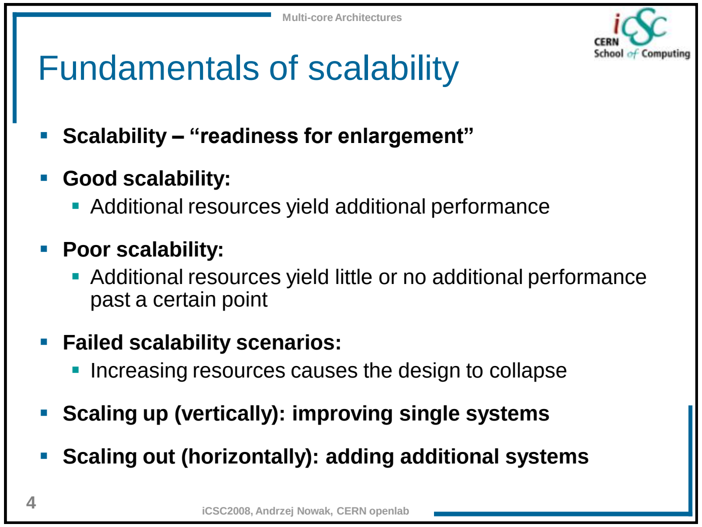
Introduction

- **Objectives:**
  - Explain why multi-core architectures have become so popular
  - Explain why parallelism is such a good bet for the near future
  - Provide information about multi-core specifics
  - Discuss the changes in computing landscape
  - Discuss the impact of hardware on software
- **Contents:**
  - Hardware part
  - Software part
  - Outlook



THE FREE RIDE IS OVER

Recession looms?



Fundamentals of scalability

- **Scalability – "readiness for enlargement"**
- **Good scalability:**
  - Additional resources yield additional performance
- **Poor scalability:**
  - Additional resources yield little or no additional performance past a certain point
- **Failed scalability scenarios:**
  - Increasing resources causes the design to collapse
- **Scaling up (vertically): improving single systems**
- **Scaling out (horizontally): adding additional systems**

CERN School of Computing

Moore's Law (1)

- **An observation made in 1965 by Gordon Moore, the co-founder of Intel Corporation:**
  - "The amount of transistors in an integrated circuit will double every 24 (18) months"
- **What is Moore's Law?**
  - A self fulfilling prophecy?
  - A common observation?

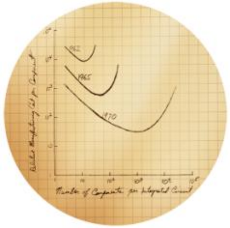


Image: Intel Corporation

CERN School of Computing

Moore's Law (2)

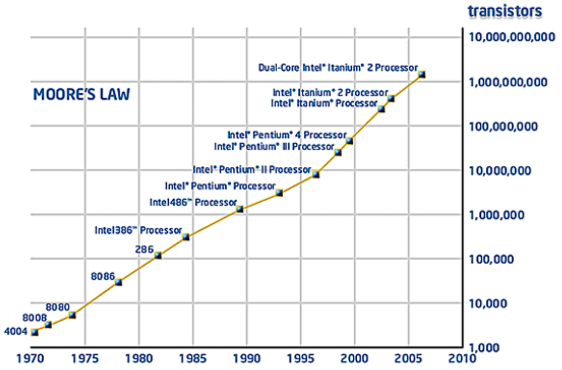


Image: Intel Corporation

CERN School of Computing

Improving CPU performance

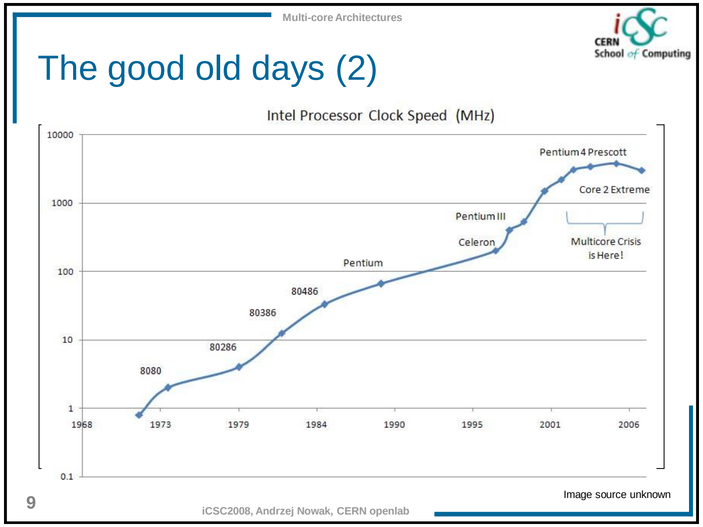
- **Common ways to improve the performance of a CPU on the hardware level**

Technique	Advantages	Disadvantages
Frequency scaling	Nearly no design overhead, immediate scaling	Some manufacturing process overhead, leakage problems
Architectural changes	Increased versatility, performance	Huge design and manufacturing overhead, minor (20%) speedups
Simultaneous multi-threading	Medium design overhead, up to 30% performance improvement	Requires more memory, single thread performance hit (~10-15%)
Cloning chips (MCP)	Minimal design overhead	Requires parallel software & more memory, inter-chip communication difficult
Adding processing cores	Small design overhead, easy to scale, 50%+ performance improvement	Requires parallel software or more memory

CERN School of Computing

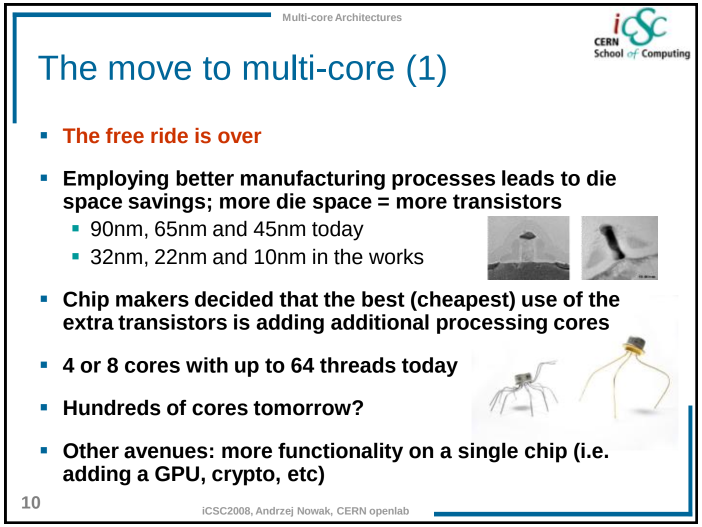
The good old days (1)

- **In the good old days one just had to upgrade the frequency of a CPU / BUS to get better performance**
- **The painful lesson of the Intel Pentium 4 (Netburst microarchitecture)**
  - Ca. 20% better performance than Pentium III (rough figure)
  - Good scaling to large frequencies – in theory up to 10 GHz
  - Surprise: very serious leakage problems (20-30W cited)
    - 130nm -> 90nm process transition
    - Relatively unsophisticated materials used for production
  - AMD chose "more logic" instead of "more speed" and began to win in certain market segments
- **In a nutshell: power dissipation problems are closing this avenue, and call for improvements even in modern designs**



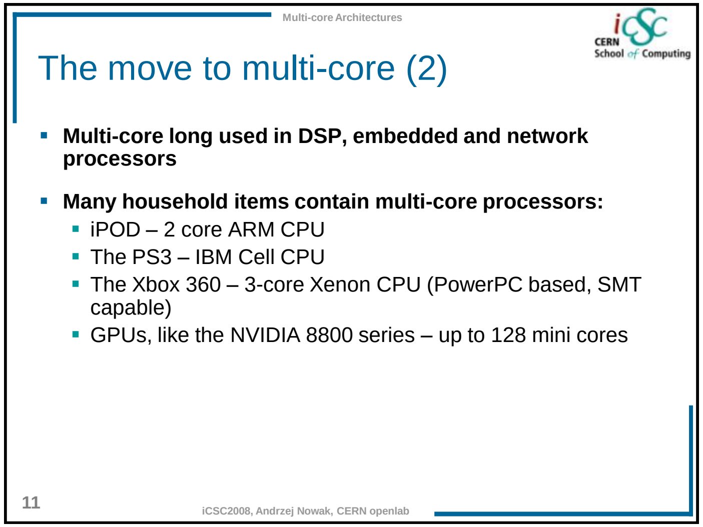
The good old days (2)

Image source unknown



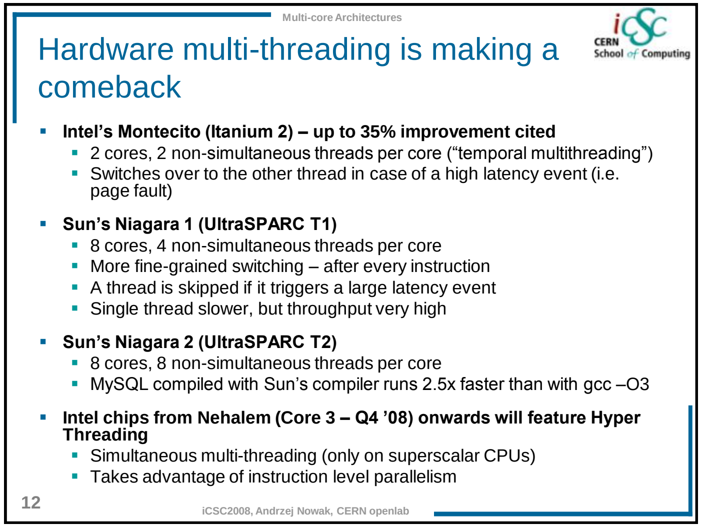
The move to multi-core (1)

- **The free ride is over**
- **Employing better manufacturing processes leads to die space savings; more die space = more transistors**
  - 90nm, 65nm and 45nm today
  - 32nm, 22nm and 10nm in the works
- **Chip makers decided that the best (cheapest) use of the extra transistors is adding additional processing cores**
- **4 or 8 cores with up to 64 threads today**
- **Hundreds of cores tomorrow?**
- **Other avenues: more functionality on a single chip (i.e. adding a GPU, crypto, etc)**



The move to multi-core (2)

- **Multi-core long used in DSP, embedded and network processors**
- **Many household items contain multi-core processors:**
  - iPOD – 2 core ARM CPU
  - The PS3 – IBM Cell CPU
  - The Xbox 360 – 3-core Xenon CPU (PowerPC based, SMT capable)
  - GPUs, like the NVIDIA 8800 series – up to 128 mini cores



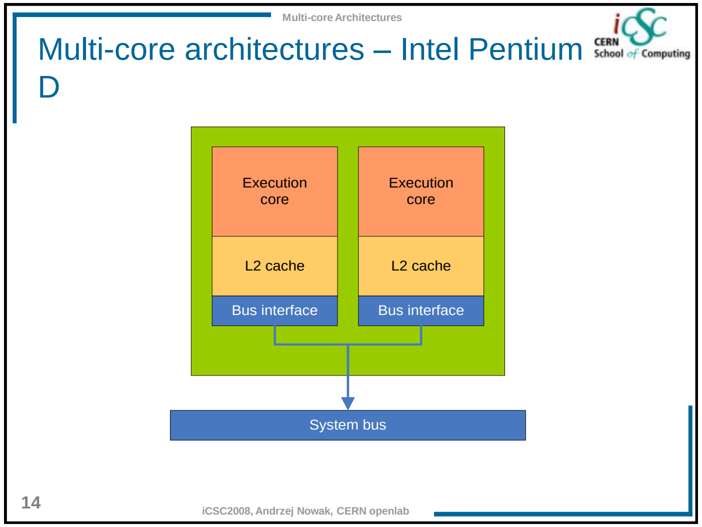
Hardware multi-threading is making a comeback

- **Intel's Montecito (Itanium 2) – up to 35% improvement cited**
  - 2 cores, 2 non-simultaneous threads per core ("temporal multithreading")
  - Switches over to the other thread in case of a high latency event (i.e. page fault)
- **Sun's Niagara 1 (UltraSPARC T1)**
  - 8 cores, 4 non-simultaneous threads per core
  - More fine-grained switching – after every instruction
  - A thread is skipped if it triggers a large latency event
  - Single thread slower, but throughput very high
- **Sun's Niagara 2 (UltraSPARC T2)**
  - 8 cores, 8 non-simultaneous threads per core
  - MySQL compiled with Sun's compiler runs 2.5x faster than with gcc –O3
- **Intel chips from Nehalem (Core 3 – Q4 '08) onwards will feature Hyper Threading**
  - Simultaneous multi-threading (only on superscalar CPUs)
  - Takes advantage of instruction level parallelism

CERN School of Computing

BASIC MULTI-CORE ARCHITECTURES

Common goods



CERN School of Computing

Multi-core architectures – Intel Pentium D

Execution core

Execution core

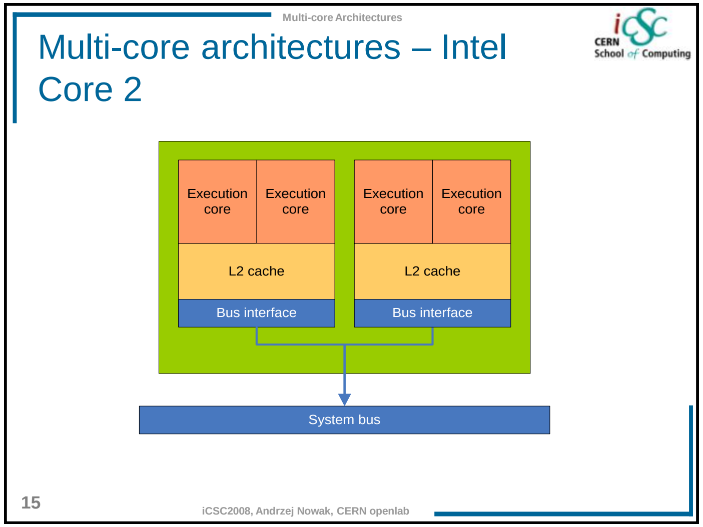
L2 cache

L2 cache

Bus interface

Bus interface

System bus



CERN School of Computing

Multi-core architectures – Intel Core 2

Execution core

Execution core

Execution core

Execution core

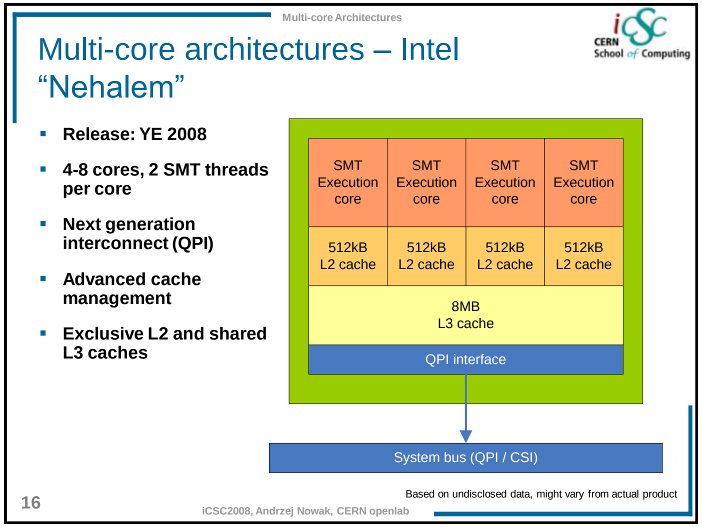
L2 cache

L2 cache

Bus interface

Bus interface

System bus



CERN School of Computing

Multi-core architectures – Intel “Nehalem”

- **Release: YE 2008**
- **4-8 cores, 2 SMT threads per core**
- **Next generation interconnect (QPI)**
- **Advanced cache management**
- **Exclusive L2 and shared L3 caches**

SMT Execution core

SMT Execution core

SMT Execution core

SMT Execution core

512kB L2 cache

512kB L2 cache

512kB L2 cache

512kB L2 cache

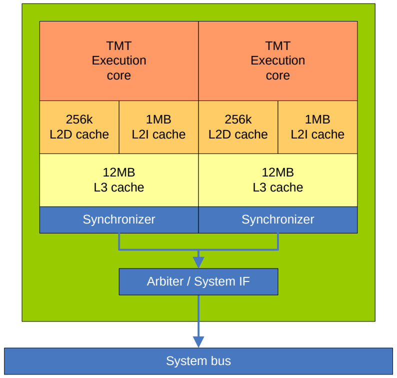
8MB L3 cache

QPI interface

System bus (QPI / CSI)

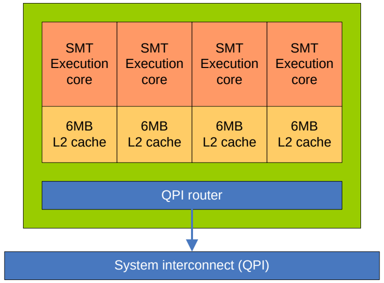
Based on undisclosed data, might vary from actual product

Multi-core architectures – Intel Itanium 2 (“Montecito”)

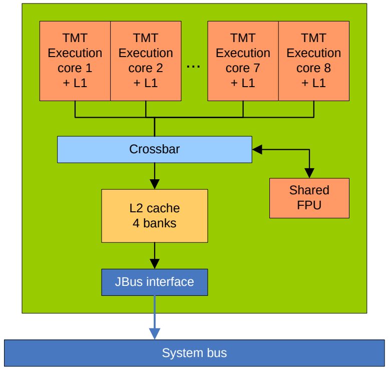

TMT Execution core
TMT Execution core
256k L2D cache
1MB L2I cache
256k L2D cache
1MB L2I cache
12MB L3 cache
12MB L3 cache
Synchronizer
Synchronizer
Arbiter / System IF
System bus

Multi-core architectures – Intel Itanium 3 (“Tukwila”)

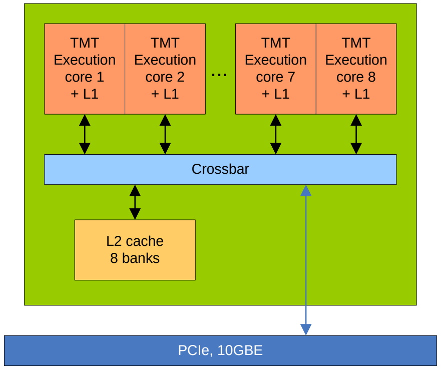
- **Release: YE2008**
- **Estimated 40GFlops / socket @ 2.5 GHz**
- **24MB L2 cache**
- **Next generation interconnect (QPI)**
- **30% improvement over “Montecito” (Itanium 2) per core**
- **Socket compatible with Xeon MP**


SMT Execution core
SMT Execution core
SMT Execution core
SMT Execution core
6MB L2 cache
6MB L2 cache
6MB L2 cache
6MB L2 cache
QPI router
System interconnect (QPI)

Multi-core architectures – UltraSPARC T1


TMT Execution core 1 + L1
TMT Execution core 2 + L1
…
TMT Execution core 7 + L1
TMT Execution core 8 + L1
Crossbar
Shared FPU
L2 cache 4 banks
JBus interface
System bus

Multi-core architectures – UltraSPARC T2


TMT Execution core 1 + L1
TMT Execution core 2 + L1
…
TMT Execution core 7 + L1
TMT Execution core 8 + L1
Crossbar
L2 cache 8 banks
PCIe, 10GBE

ADVANCED MULTI-CORE ARCHITECTURES

Just a taste

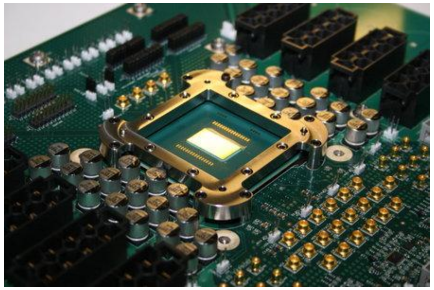
Multi-core architectures – Intel Larrabee

- **45nm process**
- **1.7 – 2.5 GHz, > 150W**
- **2 memory controllers, 2 texture samplers**
- **16-24 in order cores for pixel/vertex shading**
- **LRB core characteristics:**
  - 4 threads
  - capable of 2 double-precision FP ops per cycle
  - 32kB L1 cache, 256kB L2 cache
  - Rumor has it that the architecture is based on the Pentium MMX

SPECULATIVE INFORMATION. Source: ArsTechnica

Multi-core architectures – Intel Polaris

- **80 cores**
- **~1 TFLOPs @ ~50 Watts, ~2 TFLOPS @ ~200 Watts**



Multi-core architectures – NVIDIA G80

- **128 stream processors**
- **330 GFlops (today’s general purpose CPUs have ~10)**
- **150W**
- **Top of the line graphics hardware (along with the G92)**

Multi-core architectures – IBM Cell

- **Joint SONY / IBM / Toshiba design used primarily in the Playstation 3 gaming console**
- **Central PowerPC based processing element**
  - 2 threads
  - RISC, in order
  - AltiVec
- **A number of additional vector processing elements (SPEs)**
  - In general: 8 available
  - In the PS3: 6 available + 1 for the OS + 1 locked
- **CELL based servers and blades are available on the market**
- **IBM is building a CELL based supercomputer – Roadrunner**

Multi-core architectures – IBM Xenon

- **Used in the XBOX 360**
- **Three cores**
  - 3.2 GHz
  - 2 threads (SMT)
  - VMX SIMD instructions
  - In-order
- **Up to 116 GFLOPS cited**

Towards many-core – hardware summary

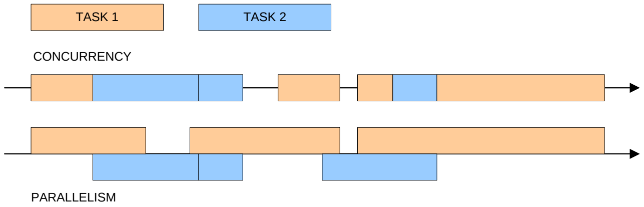
- **Sun's "Niagara 2"**
  - 64 threads running concurrently
- **Intel's Nehalem:**
  - Up to 8 cores and 16 simultaneous threads
- **Intel's Larrabee:**
  - 16-24 cores, 4 threads per core (speculated)
- **Pat Gelsinger, Intel VP on Larrabee:**
  - "different versions of the chip will have a different number of cores" ("The Register", 17th April 2007)

TAKE ALL YOU WANT

Eat all you take

Common parallelism concepts (1)

- **Concurrency is not parallelism**



- **Synchronization and race conditions**
- **Sharing and exclusion**
- **Deadlock and livelock**
- **Producing, consuming and starvation**

Common parallelism concepts (2)

- **Turnaround – complete one task as soon as possible**
  - Example: Reviewing a paper, each of the three collaborators reads one part
- **Throughput – complete the most tasks in a given amount of time**
  - Example: PC farm
- **Capabilities – better or more detailed results**
- **Efficiency – keeping the hardware busy**
  - Load balancing is important
- **Granularity – at which level is the work being split?**
  - Consider the overhead costs

Amdahl's Law (1)

- **Your principal enemy…**
  - …"keep your friends close, and your enemies closer"
- **Describes the relationship between the parallelized portion of code and the expected speedup**
  - P – parallelized portion
  - S – parallelized portion speedup

$$Speedup = \frac{1}{(1-P)+\frac{P}{S}}$$

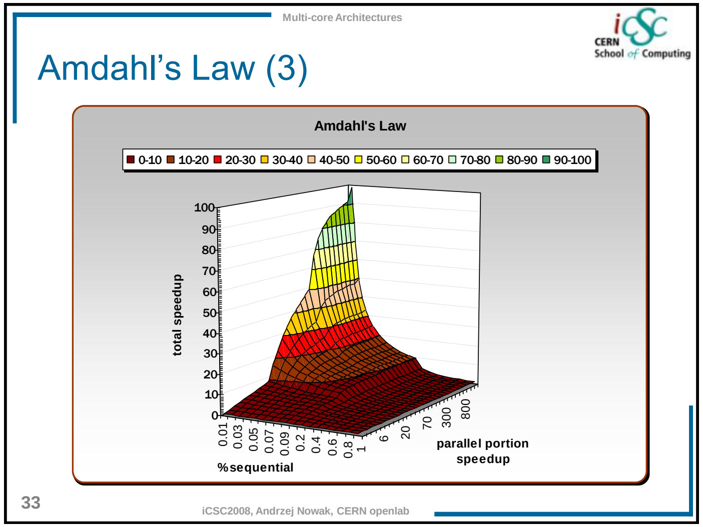
Amdahl's Law (2)

- **Baking a delicious cake**
  - Preparing the ingredients – 40% of the time (parallel)
  - Baking the cake – 60% of the time (sequential)
- **Cake process speedup with 4 people:**

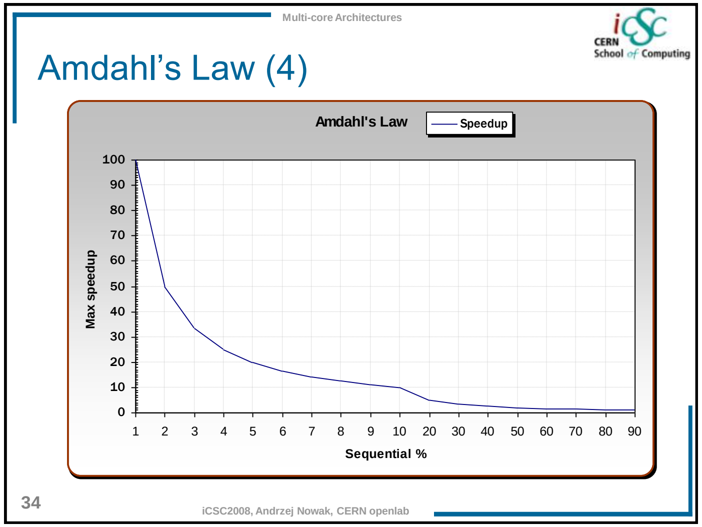
$$S_4 = \frac{1}{(1-0.4)+\frac{0.4}{4}} = 1.43$$

- **Maximum cake process speedup according to Amdahl's law:**

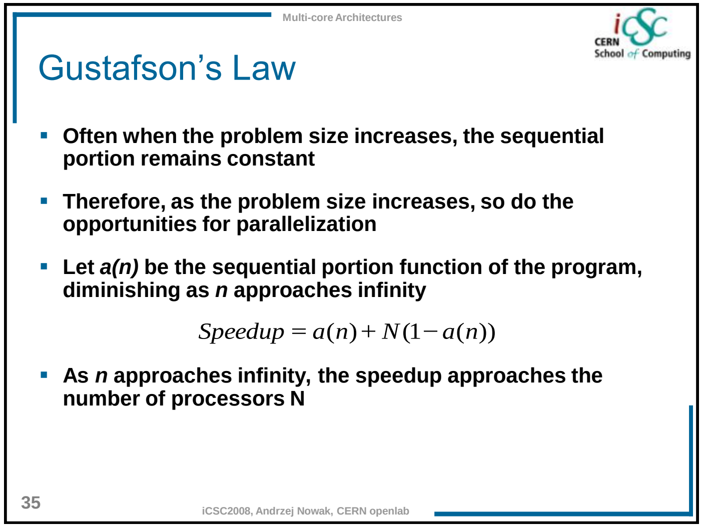
$$S_{\max} = \frac{1}{(1-0.4)+\frac{0.4}{\infty}} = 1.66$$



Amdahl's Law (3)



Amdahl's Law (4)

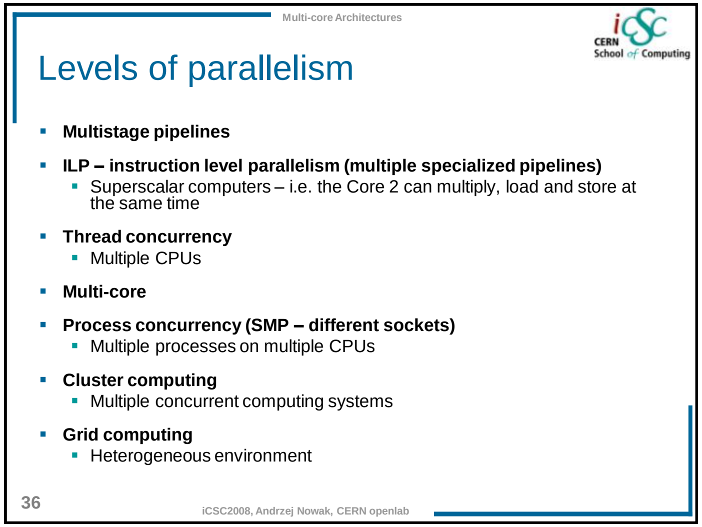


Gustafson's Law

- **Often when the problem size increases, the sequential portion remains constant**
- **Therefore, as the problem size increases, so do the opportunities for parallelization**
- **Let *a(n)* be the sequential portion function of the program, diminishing as *n* approaches infinity**

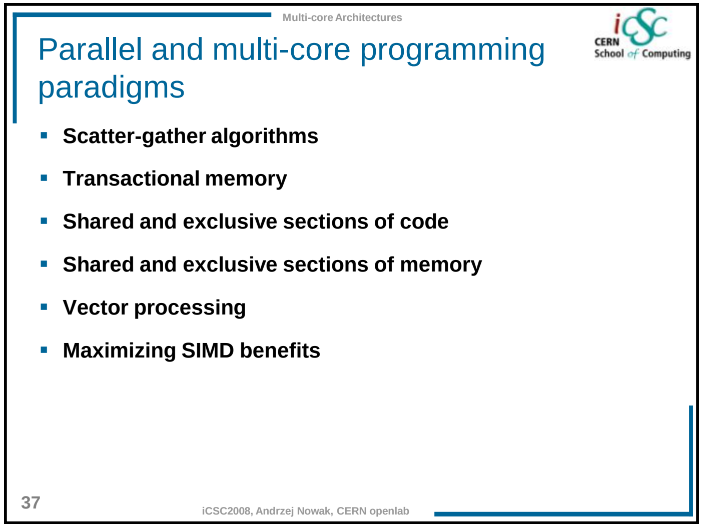
$$Speedup = a(n) + N(1 - a(n))$$

- **As *n* approaches infinity, the speedup approaches the number of processors N**



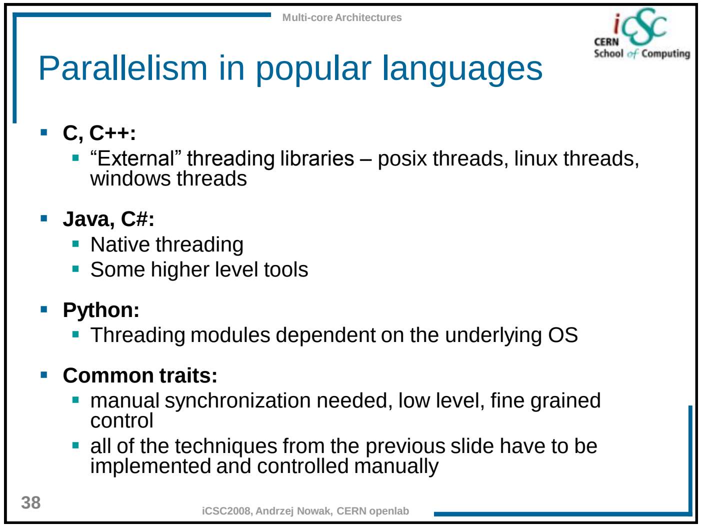
Levels of parallelism

- **Multistage pipelines**
- **ILP – instruction level parallelism (multiple specialized pipelines)**
  - Superscalar computers – i.e. the Core 2 can multiply, load and store at the same time
- **Thread concurrency**
  - Multiple CPUs
- **Multi-core**
- **Process concurrency (SMP – different sockets)**
  - Multiple processes on multiple CPUs
- **Cluster computing**
  - Multiple concurrent computing systems
- **Grid computing**
  - Heterogeneous environment



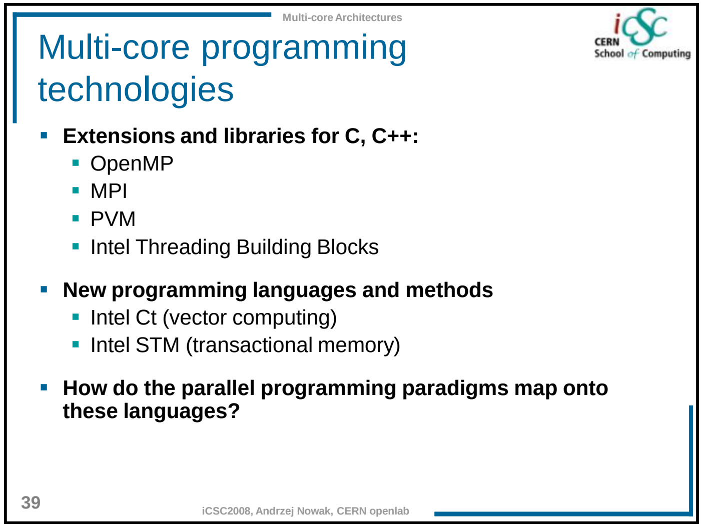
Parallel and multi-core programming paradigms

- **Scatter-gather algorithms**
- **Transactional memory**
- **Shared and exclusive sections of code**
- **Shared and exclusive sections of memory**
- **Vector processing**
- **Maximizing SIMD benefits**



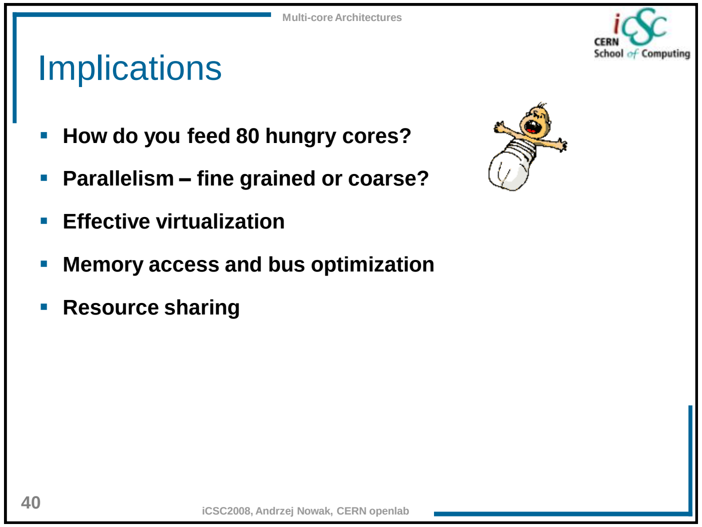
Parallelism in popular languages

- **C, C++:**
  - “External” threading libraries – posix threads, linux threads, windows threads
- **Java, C#:**
  - Native threading
  - Some higher level tools
- **Python:**
  - Threading modules dependent on the underlying OS
- **Common traits:**
  - manual synchronization needed, low level, fine grained control
  - all of the techniques from the previous slide have to be implemented and controlled manually



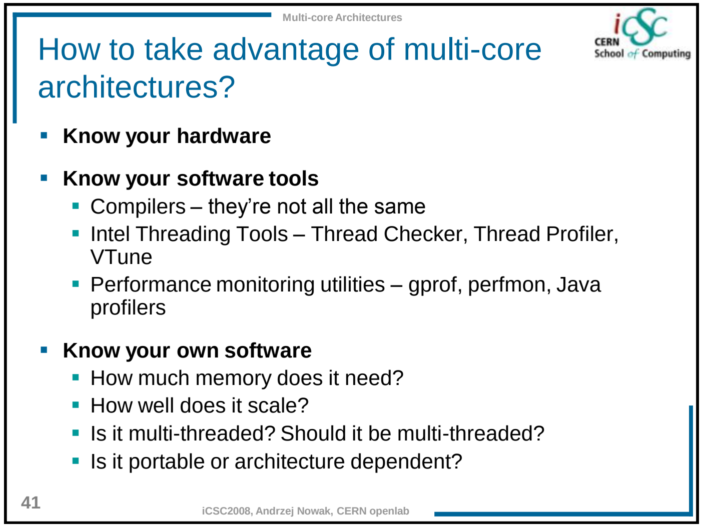
Multi-core programming technologies

- **Extensions and libraries for C, C++:**
  - OpenMP
  - MPI
  - PVM
  - Intel Threading Building Blocks
- **New programming languages and methods**
  - Intel Ct (vector computing)
  - Intel STM (transactional memory)
- **How do the parallel programming paradigms map onto these languages?**



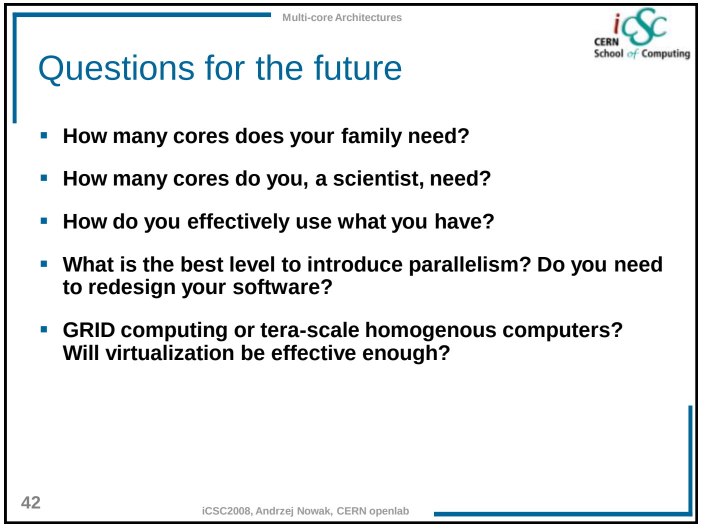
Implications

- **How do you feed 80 hungry cores?**
- **Parallelism – fine grained or coarse?**
- **Effective virtualization**
- **Memory access and bus optimization**
- **Resource sharing**



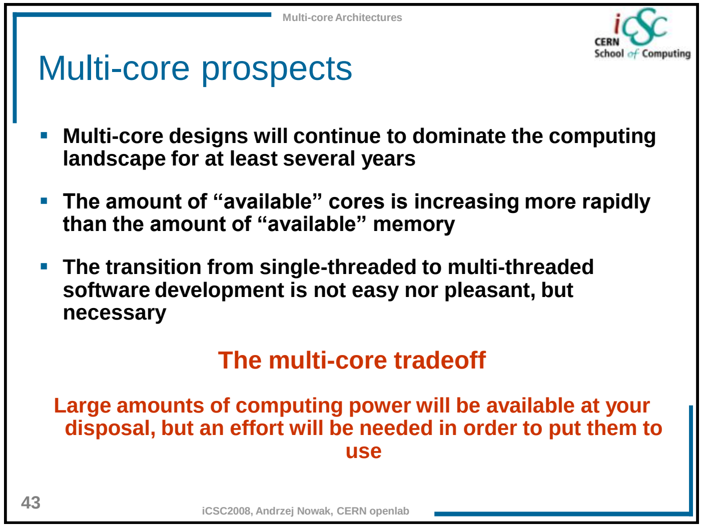
How to take advantage of multi-core architectures?

- **Know your hardware**
- **Know your software tools**
  - Compilers – they're not all the same
  - Intel Threading Tools – Thread Checker, Thread Profiler, VTune
  - Performance monitoring utilities – gprof, perfmon, Java profilers
- **Know your own software**
  - How much memory does it need?
  - How well does it scale?
  - Is it multi-threaded? Should it be multi-threaded?
  - Is it portable or architecture dependent?



Questions for the future

- **How many cores does your family need?**
- **How many cores do you, a scientist, need?**
- **How do you effectively use what you have?**
- **What is the best level to introduce parallelism? Do you need to redesign your software?**
- **GRID computing or tera-scale homogenous computers? Will virtualization be effective enough?**

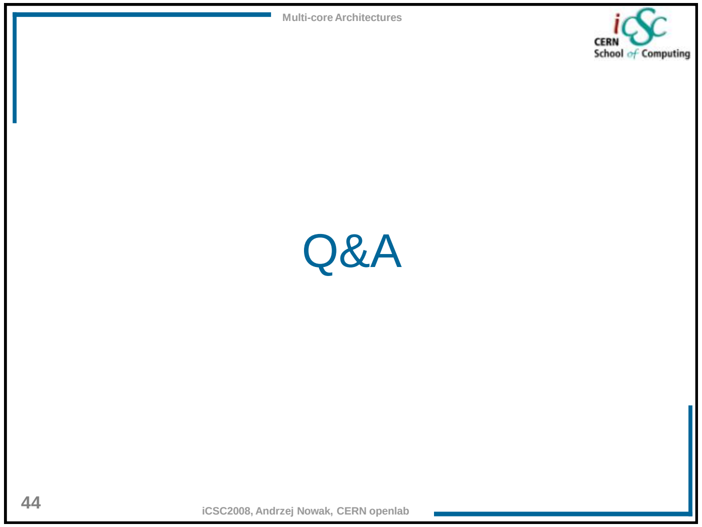


Multi-core prospects

- **Multi-core designs will continue to dominate the computing landscape for at least several years**
- **The amount of "available" cores is increasing more rapidly than the amount of "available" memory**
- **The transition from single-threaded to multi-threaded software development is not easy nor pleasant, but necessary**

The multi-core tradeoff

Large amounts of computing power will be available at your disposal, but an effort will be needed in order to put them to use



Q&A

CERN School of Computing

LECTURE 3

Platforms I: Advanced Architectural Features

Monday 3 March 2008

14:00
14:55

Lecture 3

This lecture describes some advanced architectural features of modern and upcoming CPU designs. It also addresses the problems of effective coding, especially for parallel and distributed environments.

Important topics:

- Hardware extensions
- Language extensions
- New and experimental languages and compilers
- Interesting current and upcoming hardware designs

The highlight of this lecture is the way that software relates to hardware, especially when new computational hardware comes into play.

Audience
Lecture 3 targets listeners who need to follow developments in the hardware and compiler domains.

Pre-requisite
It is recommended that the listeners follow Lecture 2 before attending this one, but it is not required.

Andrzej Nowak

Theme: Towards Reconfigurable High-Performance Computing

Lecture **3**

Platforms I: Advanced Architectural Features

Andrzej Nowak

CERN openlab (Geneva, Switzerland)

Inverted CERN School of Computing, 3-5 March 2008

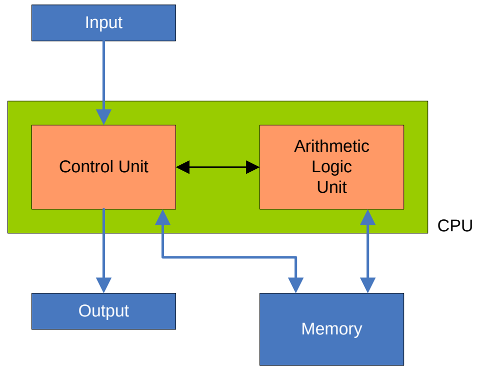
Introduction

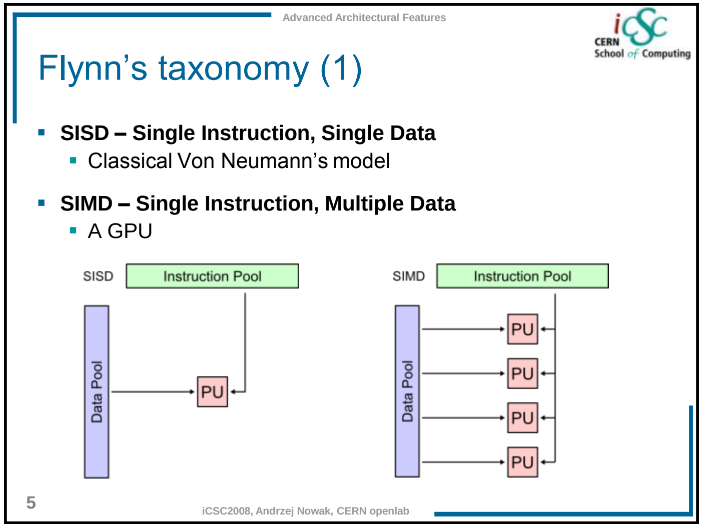
- **Recap**
  - Multi-core hardware is becoming prevalent, and is tightly coupled with the software which drives it
- **Objectives:**
  - Explain key architectural concepts
  - Discuss x86 architectural extensions
  - Discover interesting multi-core designs and interconnects
- **Contents:**
  - Systems architecture basics
  - Instruction set extensions
  - Compilers and parallelism
  - Advanced multi-core architecture discussion

COMPUTER ARCHITECTURES

And their extensions

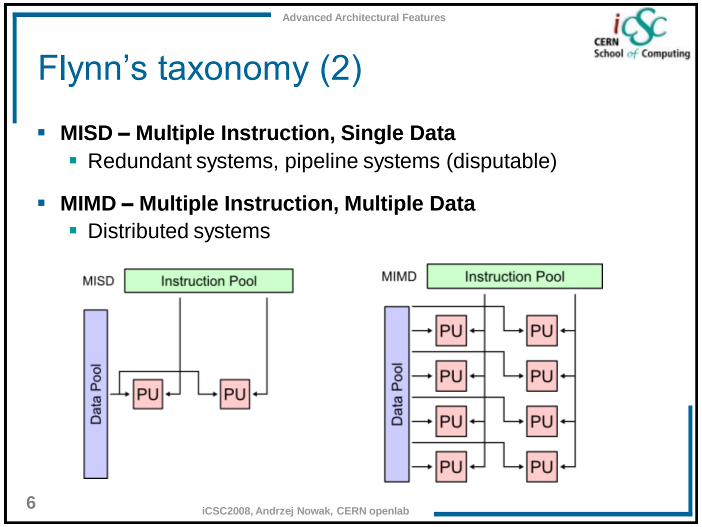
Von Neumann architecture





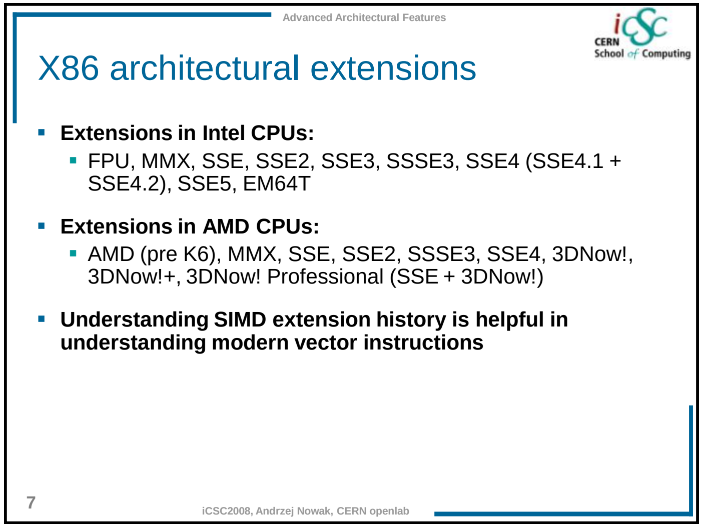
Flynn’s taxonomy (1)

- **SISD – Single Instruction, Single Data**
  - Classical Von Neumann’s model
- **SIMD – Single Instruction, Multiple Data**
  - A GPU



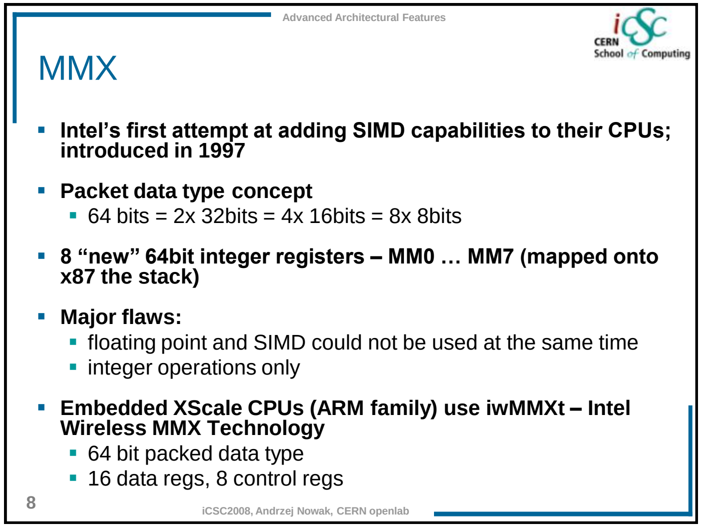
Flynn’s taxonomy (2)

- **MISD – Multiple Instruction, Single Data**
  - Redundant systems, pipeline systems (disputable)
- **MIMD – Multiple Instruction, Multiple Data**
  - Distributed systems



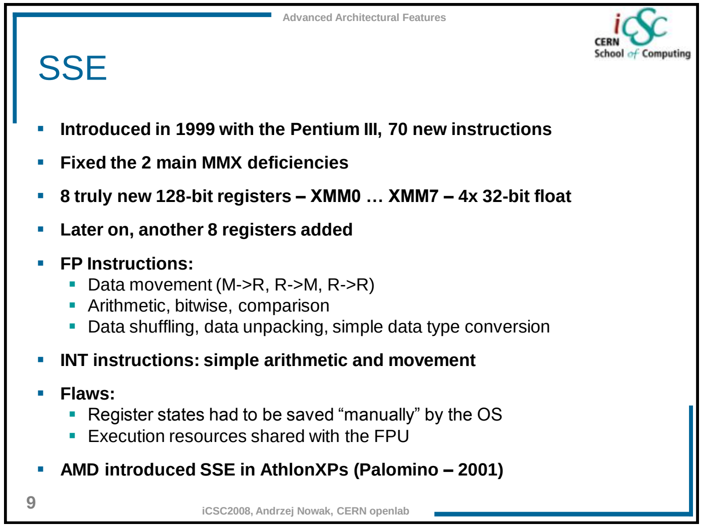
X86 architectural extensions

- **Extensions in Intel CPUs:**
  - FPU, MMX, SSE, SSE2, SSE3, SSSE3, SSE4 (SSE4.1 + SSE4.2), SSE5, EM64T
- **Extensions in AMD CPUs:**
  - AMD (pre K6), MMX, SSE, SSE2, SSSE3, SSE4, 3DNow!, 3DNow!+, 3DNow! Professional (SSE + 3DNow!)
- **Understanding SIMD extension history is helpful in understanding modern vector instructions**



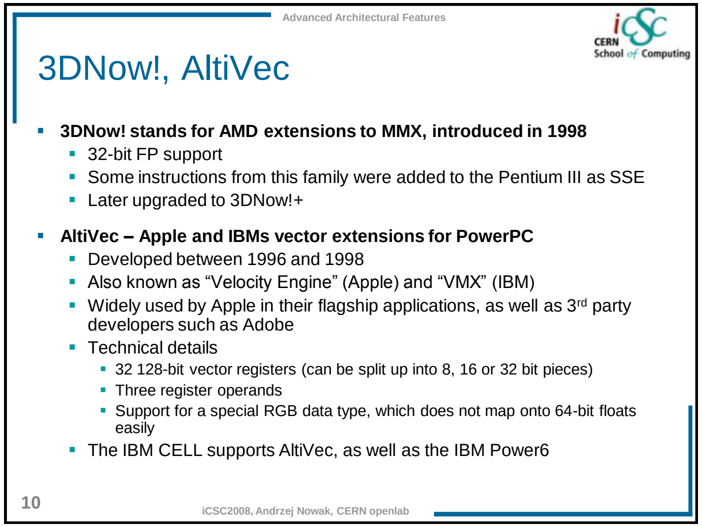
MMX

- **Intel’s first attempt at adding SIMD capabilities to their CPUs; introduced in 1997**
- **Packet data type concept**
  - 64 bits = 2x 32bits = 4x 16bits = 8x 8bits
- **8 “new” 64bit integer registers – MM0 … MM7 (mapped onto x87 the stack)**
- **Major flaws:**
  - floating point and SIMD could not be used at the same time
  - integer operations only
- **Embedded XScale CPUs (ARM family) use iwMMXt – Intel Wireless MMX Technology**
  - 64 bit packed data type
  - 16 data regs, 8 control regs



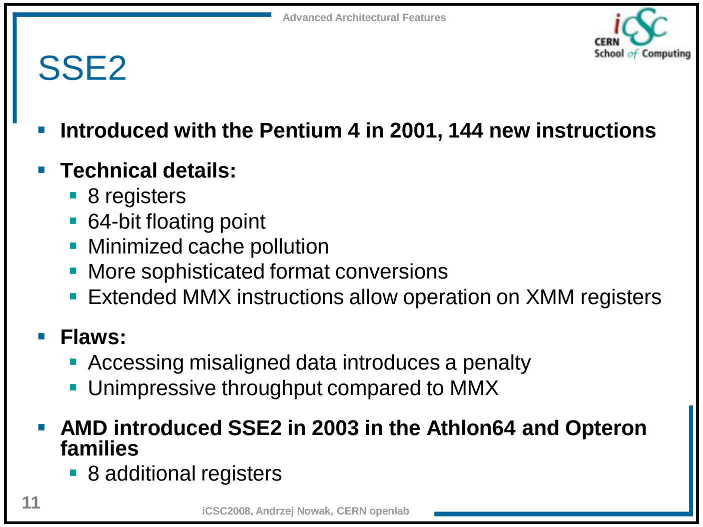
SSE

- **Introduced in 1999 with the Pentium III, 70 new instructions**
- **Fixed the 2 main MMX deficiencies**
- **8 truly new 128-bit registers – XMM0 … XMM7 – 4x 32-bit float**
- **Later on, another 8 registers added**
- **FP Instructions:**
  - Data movement (M->R, R->M, R->R)
  - Arithmetic, bitwise, comparison
  - Data shuffling, data unpacking, simple data type conversion
- **INT instructions: simple arithmetic and movement**
- **Flaws:**
  - Register states had to be saved “manually” by the OS
  - Execution resources shared with the FPU
- **AMD introduced SSE in AthlonXPs (Palomino – 2001)**



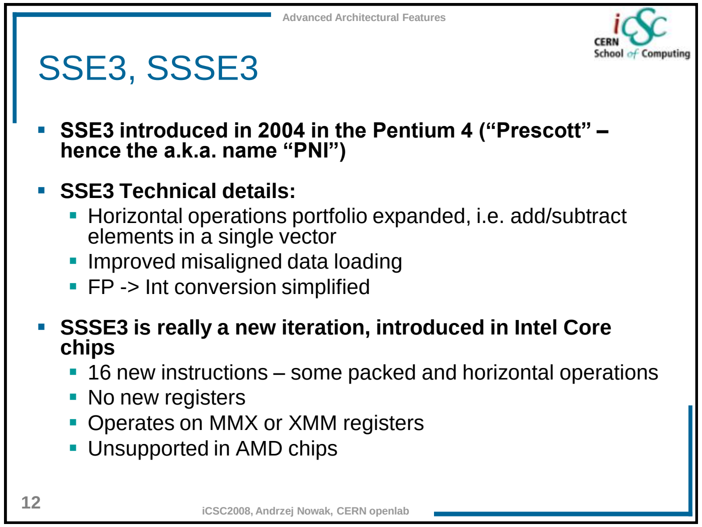
3DNow!, AltiVec

- **3DNow! stands for AMD extensions to MMX, introduced in 1998**
  - 32-bit FP support
  - Some instructions from this family were added to the Pentium III as SSE
  - Later upgraded to 3DNow!+
- **AltiVec – Apple and IBMs vector extensions for PowerPC**
  - Developed between 1996 and 1998
  - Also known as “Velocity Engine” (Apple) and “VMX” (IBM)
  - Widely used by Apple in their flagship applications, as well as 3rd party developers such as Adobe
  - Technical details
    - 32 128-bit vector registers (can be split up into 8, 16 or 32 bit pieces)
    - Three register operands
    - Support for a special RGB data type, which does not map onto 64-bit floats easily
  - The IBM CELL supports AltiVec, as well as the IBM Power6



SSE2

- **Introduced with the Pentium 4 in 2001, 144 new instructions**
- **Technical details:**
  - 8 registers
  - 64-bit floating point
  - Minimized cache pollution
  - More sophisticated format conversions
  - Extended MMX instructions allow operation on XMM registers
- **Flaws:**
  - Accessing misaligned data introduces a penalty
  - Unimpressive throughput compared to MMX
- **AMD introduced SSE2 in 2003 in the Athlon64 and Opteron families**
  - 8 additional registers



SSE3, SSSE3

- **SSE3 introduced in 2004 in the Pentium 4 (“Prescott” – hence the a.k.a. name “PNI”)**
- **SSE3 Technical details:**
  - Horizontal operations portfolio expanded, i.e. add/subtract elements in a single vector
  - Improved misaligned data loading
  - FP -> Int conversion simplified
- **SSSE3 is really a new iteration, introduced in Intel Core chips**
  - 16 new instructions – some packed and horizontal operations
  - No new registers
  - Operates on MMX or XMM registers
  - Unsupported in AMD chips

SSE4

- **54 new instructions introduced publicly in 2007**
  - SSE4.1: 47, SSE4.2: 7 (only in Nehalem)
- **Technical details:**
  - No new data types, no new registers
  - Compiler vectorization improved
  - Significant packed dword computation improvement
  - Some instructions are not multimedia related
  - Some instructions take an implicit third operand
- **You can use SSE4 with Intel compilers from version 10.0 onwards**

SSE5

- **AMD specific – a 128-bit extension of a 64-bit extension to the 32-bit original x86 instruction set; targeted for 2009**
- **170 new instructions, targeting:**
  - HPC
  - Multimedia
  - Security applications
- **Features:**
  - 3 operand ops
  - Fused instructions
  - MADD instructions
- **SSE5 software simulator**

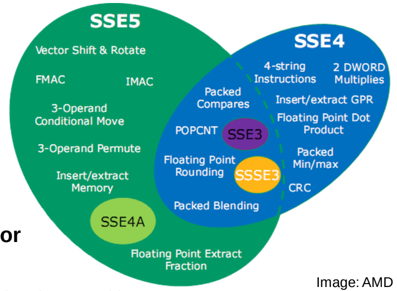


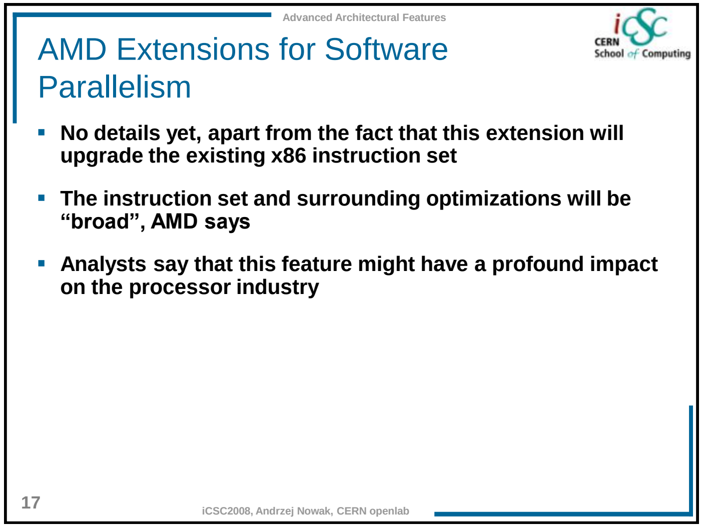
Image: AMD

AMD x86-64 (a.k.a. EM64T or Intel64)

- **Roles reversed – Intel had to follow AMD's lead**
- **64-bit operations fully supported**
  - Arithmetic
  - Registers
  - Virtual addresses
- **Expanded virtual and physical address space**
- **SSE, SSE2 and SSE3 (Intel) instructions included**
- **Cleanups**
- **There are some differences between AMD's and Intel's implementations**

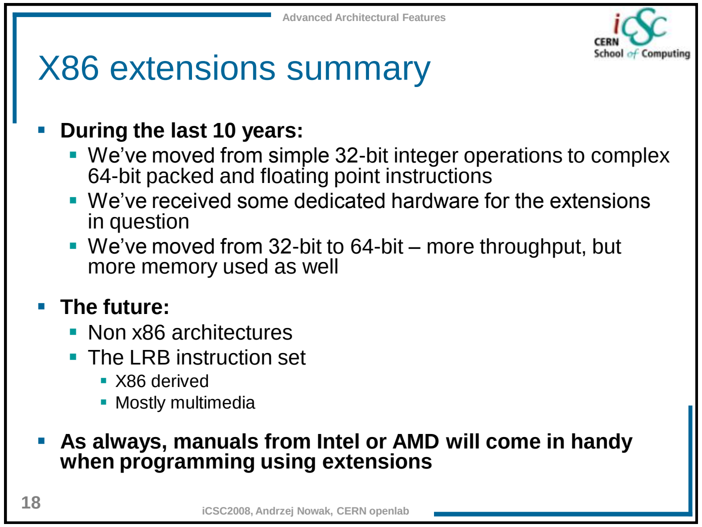
AMD Lightweight Profiling

- **Only 2 new instructions**
  - Enable/disable profiling
  - Retrieve results
- **No interrupts needed (current situation is the opposite)**
- **Profiling on the fly supported**
- **Drawbacks:**
  - New silicon needed
  - Profiling on the fly might not be that easy due to OS designs
- **Introduced no sooner than late 2008-2009**
- **We already know that upcoming Performance Monitoring Units will not differ greatly from what we have today**



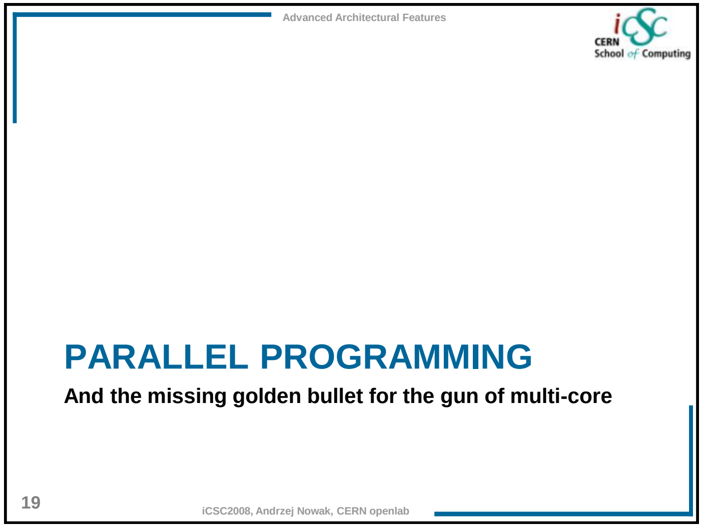
AMD Extensions for Software Parallelism

- **No details yet, apart from the fact that this extension will upgrade the existing x86 instruction set**
- **The instruction set and surrounding optimizations will be “broad”, AMD says**
- **Analysts say that this feature might have a profound impact on the processor industry**



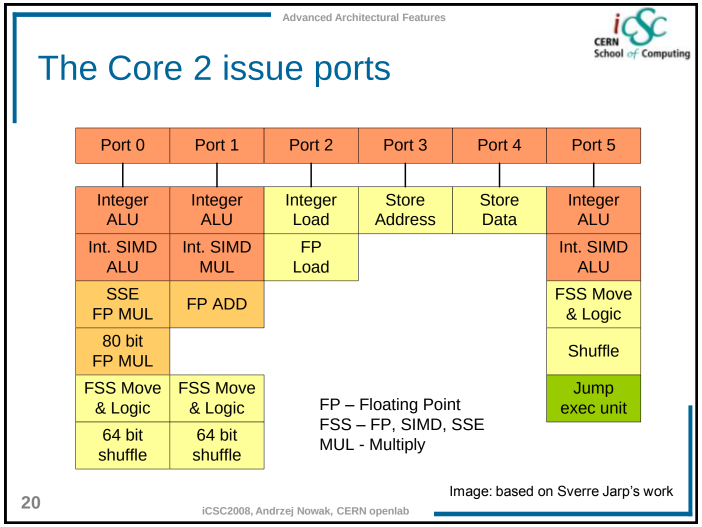
X86 extensions summary

- **During the last 10 years:**
  - We’ve moved from simple 32-bit integer operations to complex 64-bit packed and floating point instructions
  - We’ve received some dedicated hardware for the extensions in question
  - We’ve moved from 32-bit to 64-bit – more throughput, but more memory used as well
- **The future:**
  - Non x86 architectures
  - The LRB instruction set
    - X86 derived
    - Mostly multimedia
- **As always, manuals from Intel or AMD will come in handy when programming using extensions**



PARALLEL PROGRAMMING

And the missing golden bullet for the gun of multi-core

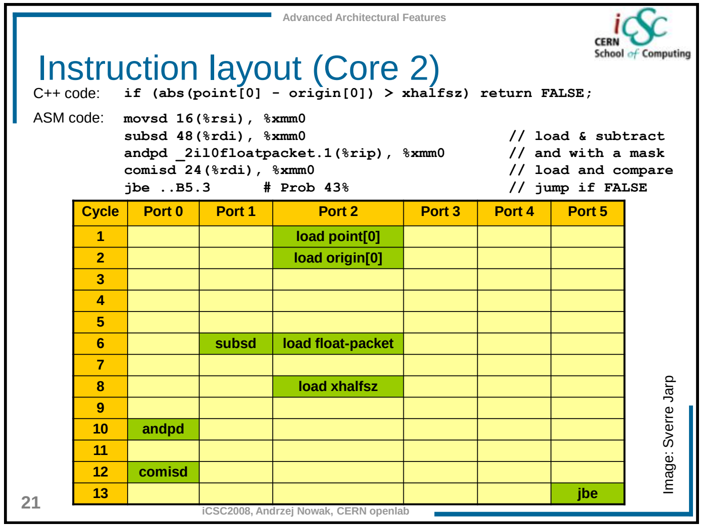


The Core 2 issue ports

Port 0	Port 1	Port 2	Port 3	Port 4	Port 5
Integer ALU	Integer ALU	Integer Load	Store Address	Store Data	Integer ALU
Int. SIMD ALU	Int. SIMD MUL	FP Load			Int. SIMD ALU
SSE FP MUL	FP ADD				FSS Move & Logic
80 bit FP MUL					Shuffle
FSS Move & Logic	FSS Move & Logic				Jump exec unit
64 bit shuffle	64 bit shuffle				

FP – Floating Point
FSS – FP, SIMD, SSE
MUL - Multiply

Image: based on Sverre Jarp’s work



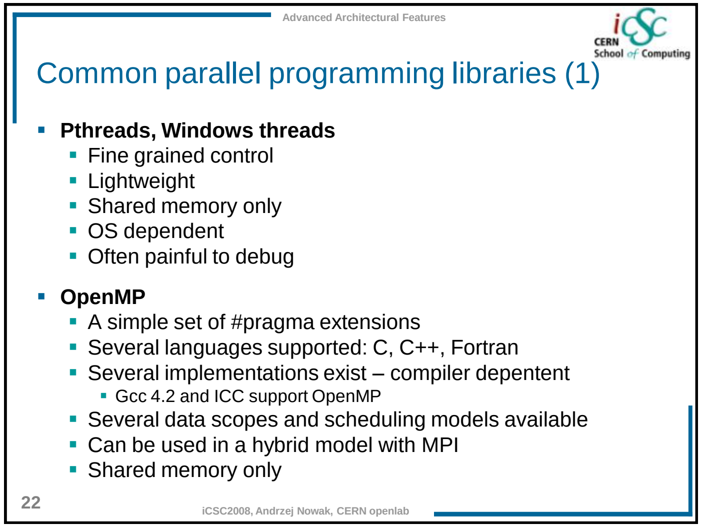
Instruction layout (Core 2)

C++ code:
```
if (abs(point[0] - origin[0]) > xhalfsz) return FALSE;
```

ASM code:
```
movsd 16(%rsi), %xmm0
subsd 48(%rdi), %xmm0                         // load & subtract
andpd _2il0floatpacket.1(%rip), %xmm0         // and with a mask
comisd 24(%rdi), %xmm0                        // load and compare
jbe ..B5.3        # Prob 43%                  // jump if FALSE
```

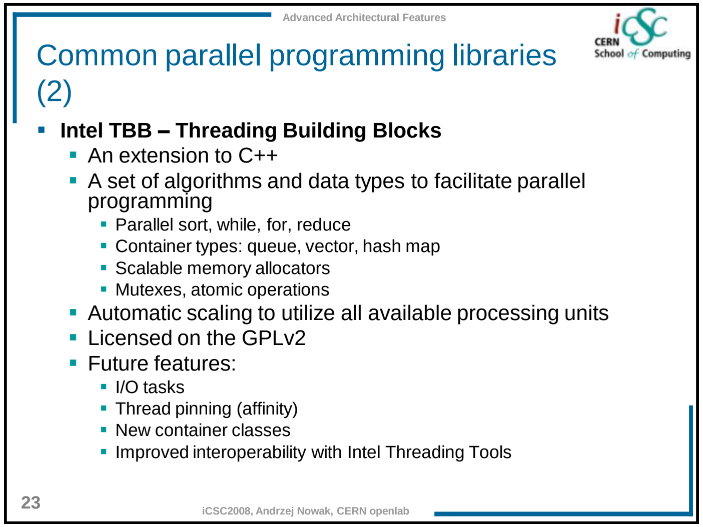
Cycle	Port 0	Port 1	Port 2	Port 3	Port 4	Port 5
1			load point[0]			
2			load origin[0]			
3						
4						
5						
6		subsd	load float-packet			
7						
8			load xhalfsz			
9						
10	andpd					
11						
12	comisd					
13						jbe

Image: Sverre Jarp



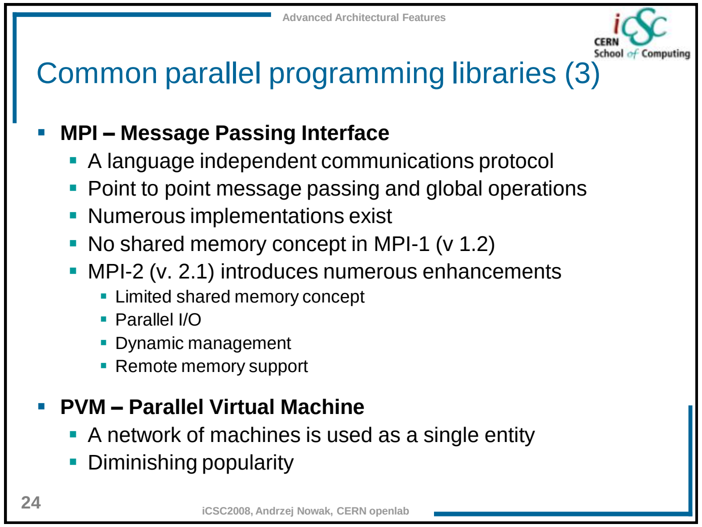
Common parallel programming libraries (1)

- **Pthreads, Windows threads**
  - Fine grained control
  - Lightweight
  - Shared memory only
  - OS dependent
  - Often painful to debug
- **OpenMP**
  - A simple set of #pragma extensions
  - Several languages supported: C, C++, Fortran
  - Several implementations exist – compiler depentent
    - Gcc 4.2 and ICC support OpenMP
  - Several data scopes and scheduling models available
  - Can be used in a hybrid model with MPI
  - Shared memory only



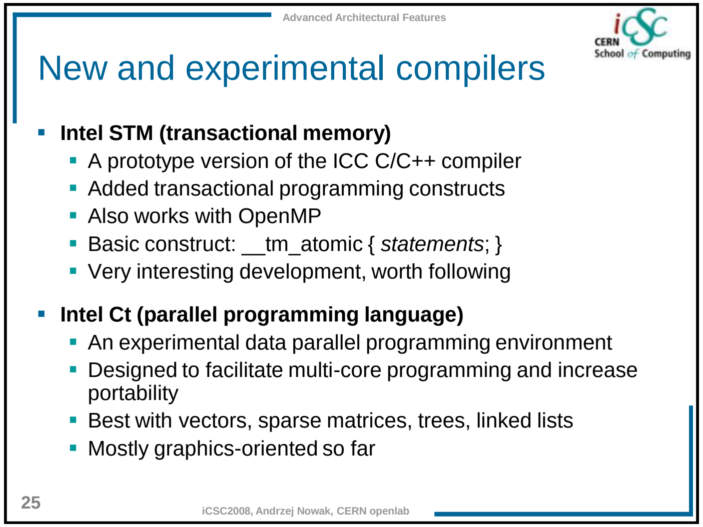
Common parallel programming libraries (2)

- **Intel TBB – Threading Building Blocks**
  - An extension to C++
  - A set of algorithms and data types to facilitate parallel programming
    - Parallel sort, while, for, reduce
    - Container types: queue, vector, hash map
    - Scalable memory allocators
    - Mutexes, atomic operations
  - Automatic scaling to utilize all available processing units
  - Licensed on the GPLv2
  - Future features:
    - I/O tasks
    - Thread pinning (affinity)
    - New container classes
    - Improved interoperability with Intel Threading Tools



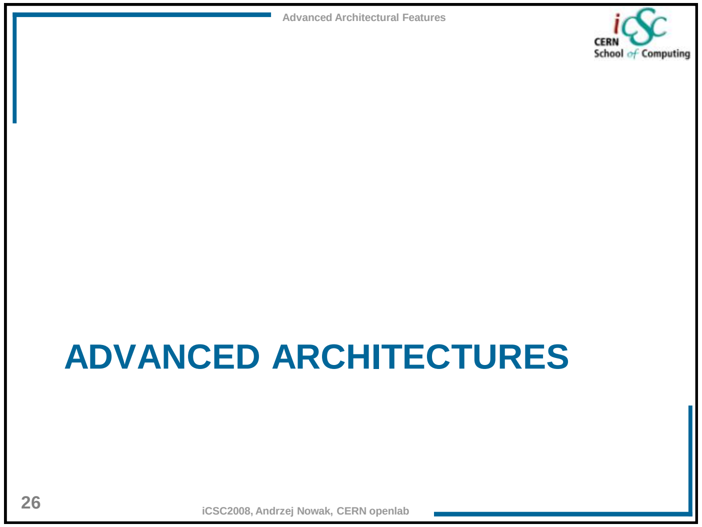
Common parallel programming libraries (3)

- **MPI – Message Passing Interface**
  - A language independent communications protocol
  - Point to point message passing and global operations
  - Numerous implementations exist
  - No shared memory concept in MPI-1 (v 1.2)
  - MPI-2 (v. 2.1) introduces numerous enhancements
    - Limited shared memory concept
    - Parallel I/O
    - Dynamic management
    - Remote memory support
- **PVM – Parallel Virtual Machine**
  - A network of machines is used as a single entity
  - Diminishing popularity

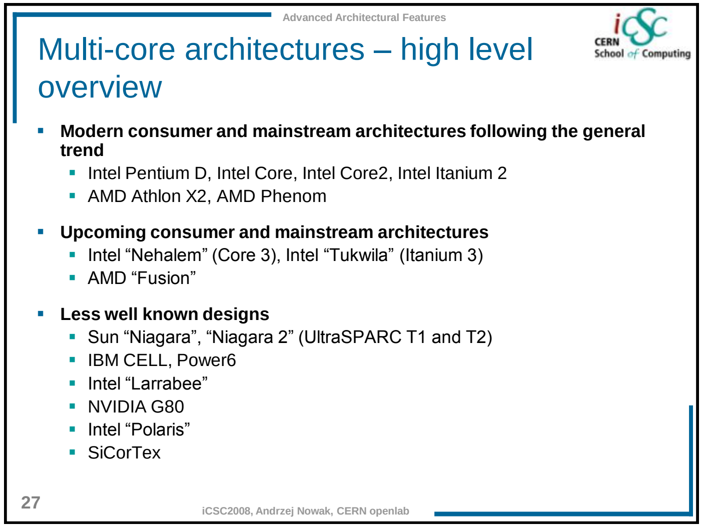


New and experimental compilers

- **Intel STM (transactional memory)**
  - A prototype version of the ICC C/C++ compiler
  - Added transactional programming constructs
  - Also works with OpenMP
  - Basic construct: __tm_atomic { *statements*; }
  - Very interesting development, worth following
- **Intel Ct (parallel programming language)**
  - An experimental data parallel programming environment
  - Designed to facilitate multi-core programming and increase portability
  - Best with vectors, sparse matrices, trees, linked lists
  - Mostly graphics-oriented so far

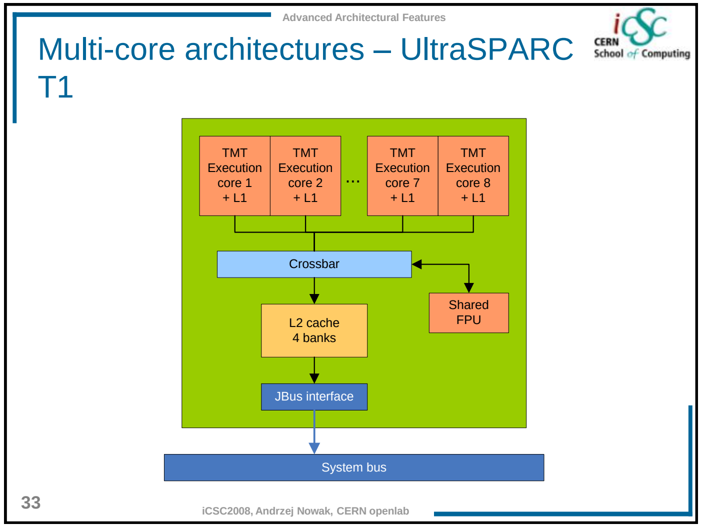


ADVANCED ARCHITECTURES



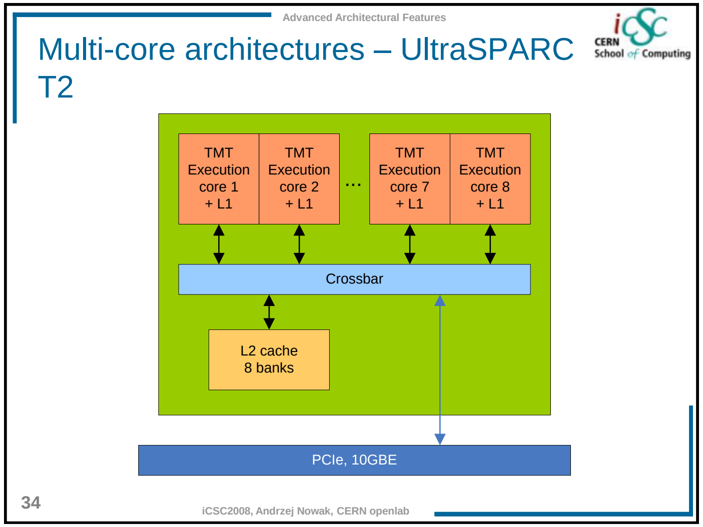
Multi-core architectures – high level overview

- **Modern consumer and mainstream architectures following the general trend**
  - Intel Pentium D, Intel Core, Intel Core2, Intel Itanium 2
  - AMD Athlon X2, AMD Phenom
- **Upcoming consumer and mainstream architectures**
  - Intel “Nehalem” (Core 3), Intel “Tukwila” (Itanium 3)
  - AMD “Fusion”
- **Less well known designs**
  - Sun “Niagara”, “Niagara 2” (UltraSPARC T1 and T2)
  - IBM CELL, Power6
  - Intel “Larrabee”
  - NVIDIA G80
  - Intel “Polaris”
  - SiCorTex



Multi-core architectures – UltraSPARC T1

TMT Execution core 1 + L1
TMT Execution core 2 + L1
…
TMT Execution core 7 + L1
TMT Execution core 8 + L1
Crossbar
Shared FPU
L2 cache 4 banks
JBus interface
System bus



Multi-core architectures – UltraSPARC T2

TMT Execution core 1 + L1

TMT Execution core 2 + L1

…

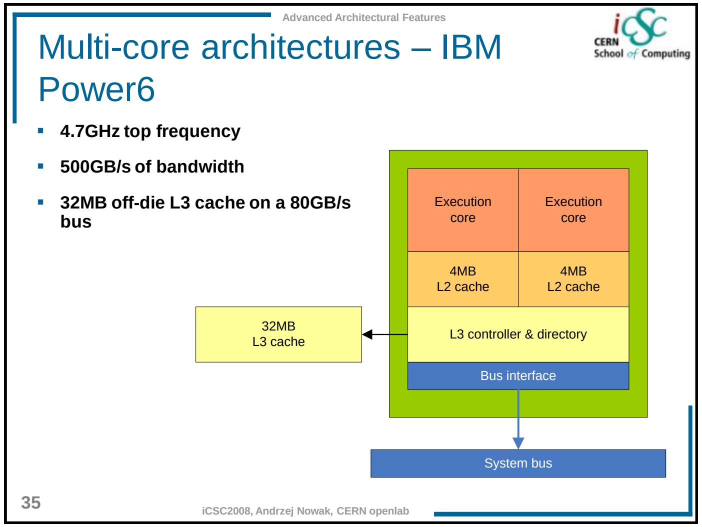
TMT Execution core 7 + L1

TMT Execution core 8 + L1

Crossbar

L2 cache 8 banks

PCIe, 10GBE



Multi-core architectures – IBM Power6

- 4.7GHz top frequency
- 500GB/s of bandwidth
- 32MB off-die L3 cache on a 80GB/s bus

Execution core

Execution core

4MB L2 cache

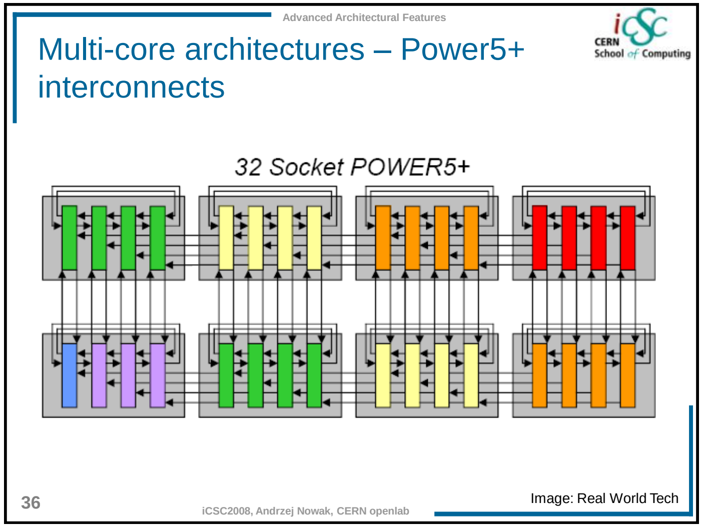
4MB L2 cache

32MB L3 cache

L3 controller & directory

Bus interface

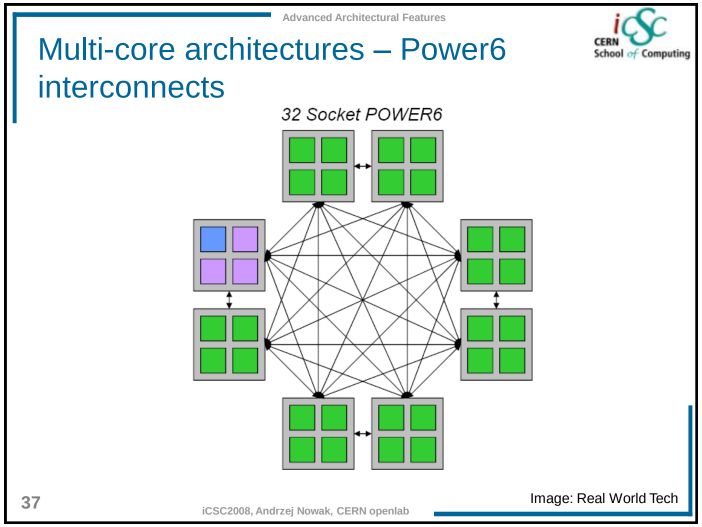
System bus



Multi-core architectures – Power5+ interconnects

32 Socket POWER5+

Image: Real World Tech

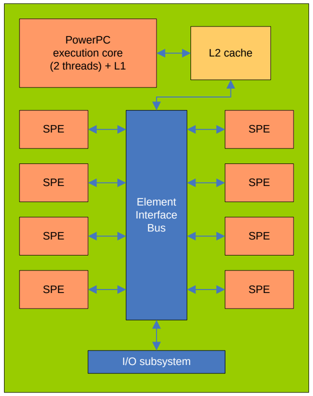


Multi-core architectures – Power6 interconnects

32 Socket POWER6

Image: Real World Tech

Interesting architectures – IBM CELL



PowerPC execution core (2 threads) + L1

L2 cache

SPE

SPE

SPE

SPE

Element Interface Bus

SPE

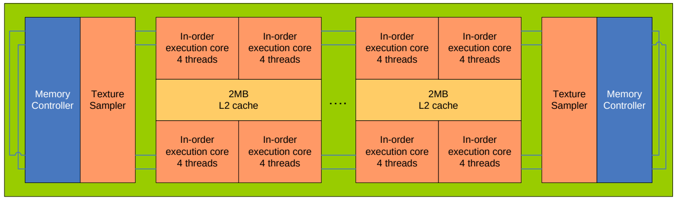
SPE

SPE

SPE

I/O subsystem

Multi-core architectures – Intel Larrabee



Memory Controller

Texture Sampler

In-order execution core 4 threads

In-order execution core 4 threads

2MB L2 cache

In-order execution core 4 threads

In-order execution core 4 threads

....

In-order execution core 4 threads

In-order execution core 4 threads

2MB L2 cache

In-order execution core 4 threads

In-order execution core 4 threads

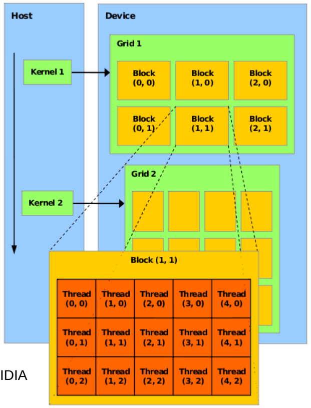
Texture Sampler

Memory Controller

SPECULATIVE INFORMATION. Source: ArsTechnica

Multi-core architectures – NVIDIA G80

- **Moved away from traditional GPU design**
- **128 stream processors**
- **330 GFLOPS peak**
- **Second generation: G92**



Host

Kernel 1

Kernel 2

Device

Grid 1

Block (0, 0)	Block (1, 0)	Block (2, 0)
Block (0, 1)	Block (1, 1)	Block (2, 1)

Grid 2

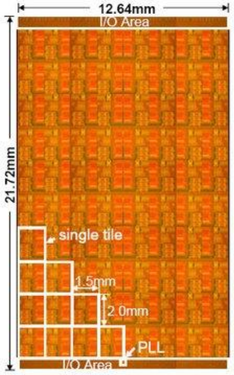
Block (1, 1)

Thread (0, 0)	Thread (1, 0)	Thread (2, 0)	Thread (3, 0)	Thread (4, 0)
Thread (0, 1)	Thread (1, 1)	Thread (2, 1)	Thread (3, 1)	Thread (4, 1)
Thread (0, 2)	Thread (1, 2)	Thread (2, 2)	Thread (3, 2)	Thread (4, 2)

Graphics: NVIDIA

Multi-core architectures – Intel Polaris (1)

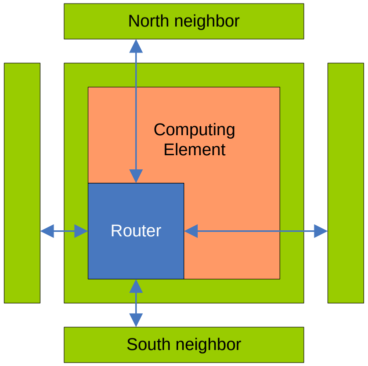
- **80 cores**
- **Tiled (mesh) architecture**
- **Array area: 13mm x 28mm**
  - Single core: 2mm x 1.5mm
- **Modular, scalable design**
- **Fine grained power management**
- **Approximate performance:**
  - 1 TFLOP @ 50-60W
  - 1.5 TFLOPS @ ~100W
  - 2 TFLOPS @ ~200-250W



Data source: computerbase.de

Multi-core architectures – Intel Polaris (2)

- **Core data:**
  - 2kB data memory
  - 3kB instruction memory
  - 32GBps interconnect
  - Tile area: 3mm^2
  - Versatile, scalable design



North neighbor

Computing Element

Router

South neighbor

Data source: computerbase.de

Interesting architectures – SiCorTex (1)

1 GFLOP 600mW power

1.5MB L2

10W

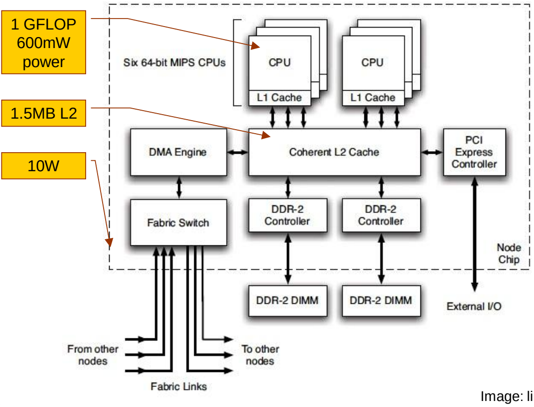


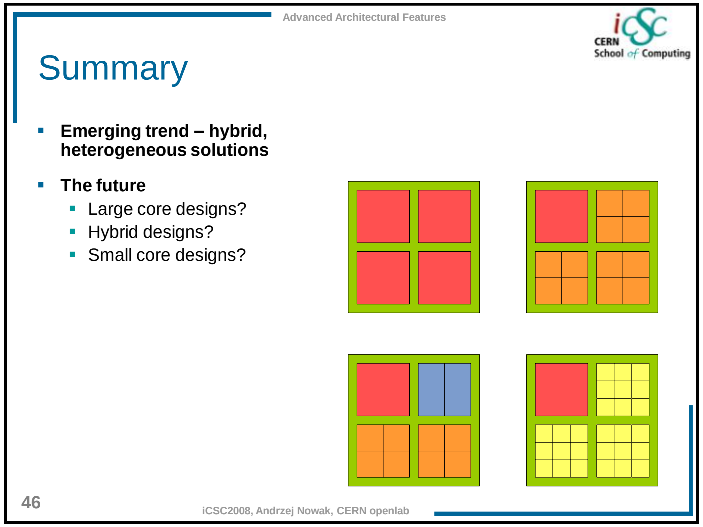
Image: linuxdevices.com

Interesting architectures – SiCorTex (2)

- **27 6-core nodes make up one blade**
- **SC5832**
  - 36 blades
  - 5832 cores
  - 5.8 TFLOPS
  - 8 TB of DDR2 memory
  - The only computing system on the Top500 list with a single backplane
  - **18 kW**
  - ~$2.5 M
- **SC648**
  - 0.648 TFLOP
  - 2kW
  - ~$200 k

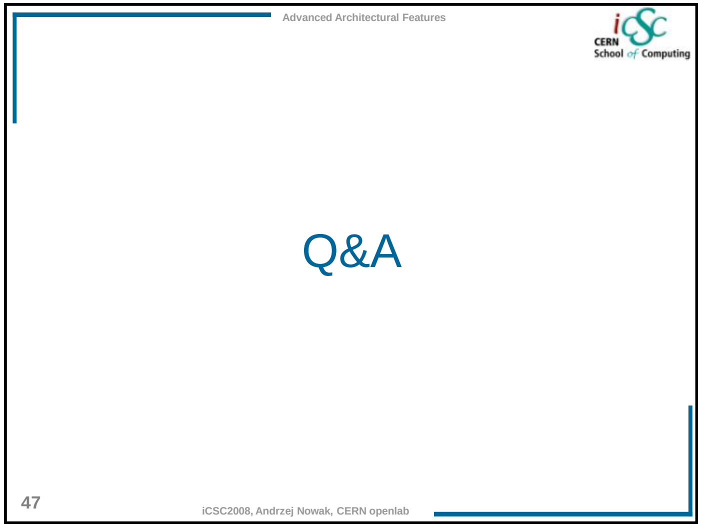
Interesting architectures – FPGAs

- **Programmable hardware**
  - Programmed using a low level hardware description language (commonly VHDL or Verilog)
  - Some higher level languages and methods are being developed
- **Heavily used in the industry, becoming popular in HPC**
- **Well suited for data streaming**
- **Common method: moving inner loops into very fast custom instructions**
- **Advantages**
  - Very fast
  - Can execute all implemented operations in parallel
- **More later**



Summary

- **Emerging trend – hybrid, heterogeneous solutions**
- **The future**
  - Large core designs?
  - Hybrid designs?
  - Small core designs?



Q&A

LECTURE 4

Platforms II: Special-Purpose Accelerators

Monday 3 March 2008			
15:05 16:00	Lecture 4	This lecture aims to familiarize the listeners with special-purpose hardware accelerators and processing units which have become popular in recent times. Important topics: • Hardware acceleration concepts and philosophy • GPUs and gaming hardware • Future directions for hardware accelerators, future scenario discussion This lecture will stress the tradeoffs which users face when reaching for ultra-fast special purpose hardware, such as GPUs. **Audience** Lecture 4 targets listeners who are interested in hardware acceleration using off-the-shelf hardware. **Pre-requisite** It is advisable to follow Lecture 3 before attending this one.	**Andrzej Nowak**

Theme: Towards Reconfigurable High-Performance Computing

Lecture **4**

Platforms II: Special Purpose Accelerators

Andrzej Nowak

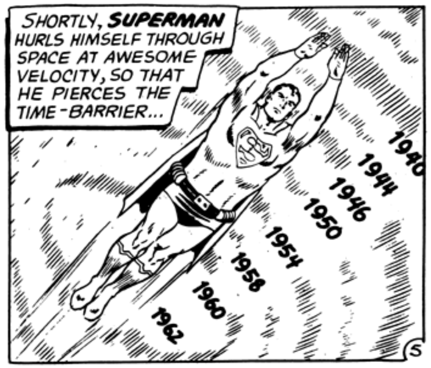
CERN openlab (Geneva, Switzerland)

Inverted CERN School of Computing, 3-5 March 2008

Introduction

- **Recap:**
  - General purpose processors excel at various jobs, but are no match for accelerators when dealing with specialized tasks
- **Objectives:**
  - Define the role and purpose of modern accelerators
  - Provide information about General Purpose GPU computing
- **Contents:**
  - Hardware accelerators
  - GPUs and general purpose computing on GPUs
  - Related hardware and software technologies

Hardware acceleration philosophy


SHORTLY, SUPERMAN HURLS HIMSELF THROUGH SPACE AT AWESOME VELOCITY, SO THAT HE PIERCES THE TIME-BARRIER...
1940
1944
1946
1950
1954
1958
1960
1962

Popular accelerators in general

- **Floating point units**
  - Old CPUs were really slow
  - Embedded CPUs often don’t have a hardware FPU
  - 1980’s PCs – the FPU was an optional add on, separate sockets for the 8087 coprocessor
- **Video and image processing**
  - MPEG decoders
  - DV decoders
  - HD decoders
- **Digital signal processing (including audio)**
  - Sound Blaster Live and friends

Mainstream accelerators today

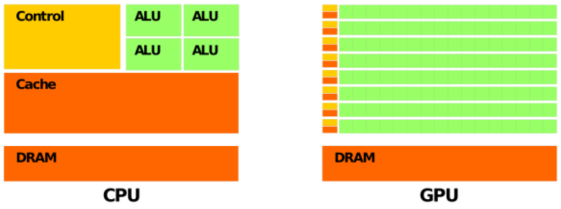
- **Integrated FPUs**
- **Realtime graphics**
  - Gaming cards
- **Gaming physics**
  - AGEIA PhysX gaming card
- **Digital audio processing**
  - Creative Sound Blaster X-FI
- **Networking**
  - KillerNIC
- **Encryption**
  - Add on and on-board dedicated crypto modules
- **Platform development**
  - AMD Torrenza (coprocessor integration initiative)
  - Intel/IBM Geneseo (PCIe extensions)

GPUs

Bobby wants to play a game

The rise of the GPUs

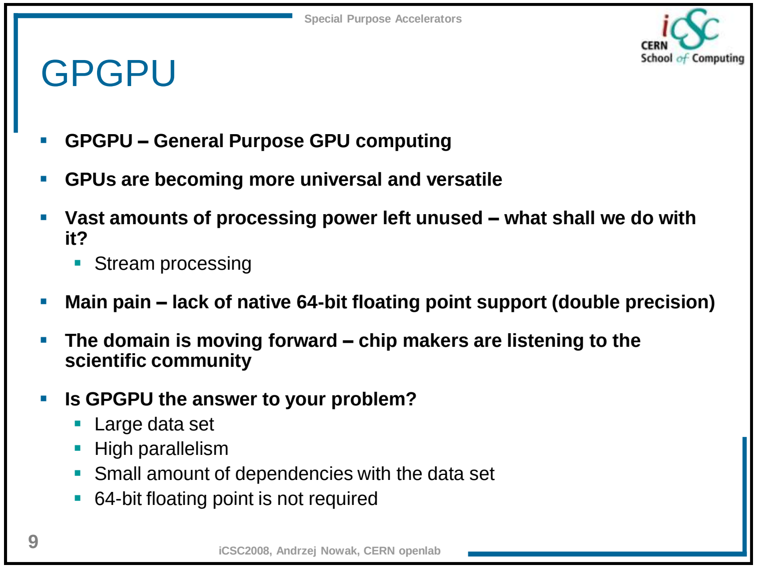
- **Graphics Processing Units – A mainstream, market-driven vector computing accelerator family**
  - Simple operations
  - Large width and throughput
  - Medium frequencies



Graphics: NVIDIA

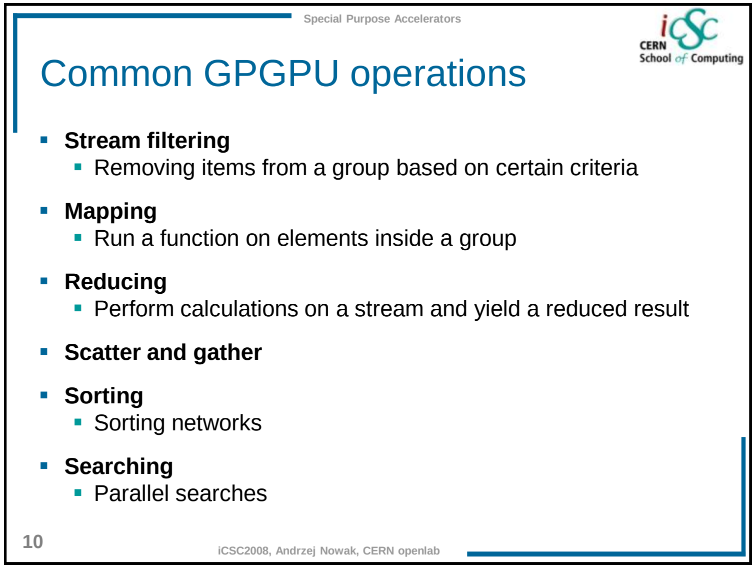
Modern GPU features

- **Dozens of processing cores**
  - Some cores usually end up disabled due to manufacturers' yield problems
- **A lot of power consumed compared to CPUs - ~150 W**
- **Very fast in vector calculations, up to hundreds of GFLOPS**
- **Market driven features**
  - Main actors: Red, Green, Blue and Alpha
  - DirectX 10 or 10.1 compatibility
  - Different shader model support
- **Active ongoing development**



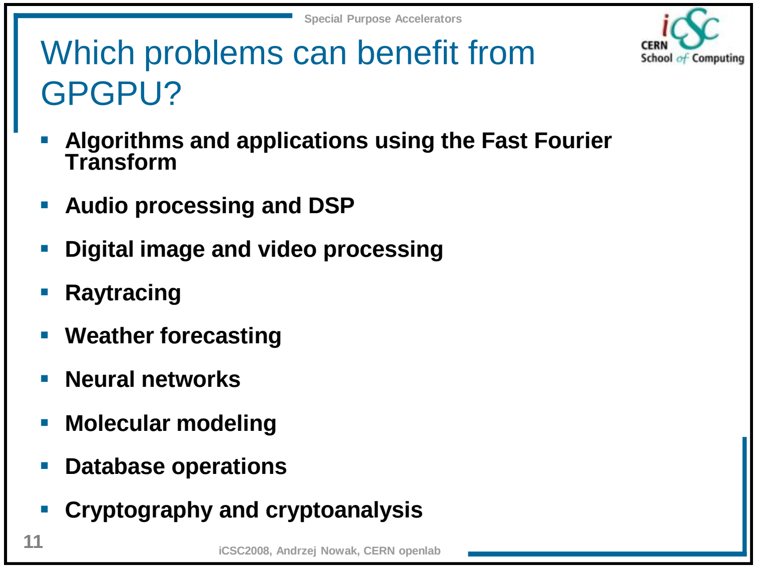
GPGPU

- **GPGPU – General Purpose GPU computing**
- **GPUs are becoming more universal and versatile**
- **Vast amounts of processing power left unused – what shall we do with it?**
  - Stream processing
- **Main pain – lack of native 64-bit floating point support (double precision)**
- **The domain is moving forward – chip makers are listening to the scientific community**
- **Is GPGPU the answer to your problem?**
  - Large data set
  - High parallelism
  - Small amount of dependencies with the data set
  - 64-bit floating point is not required



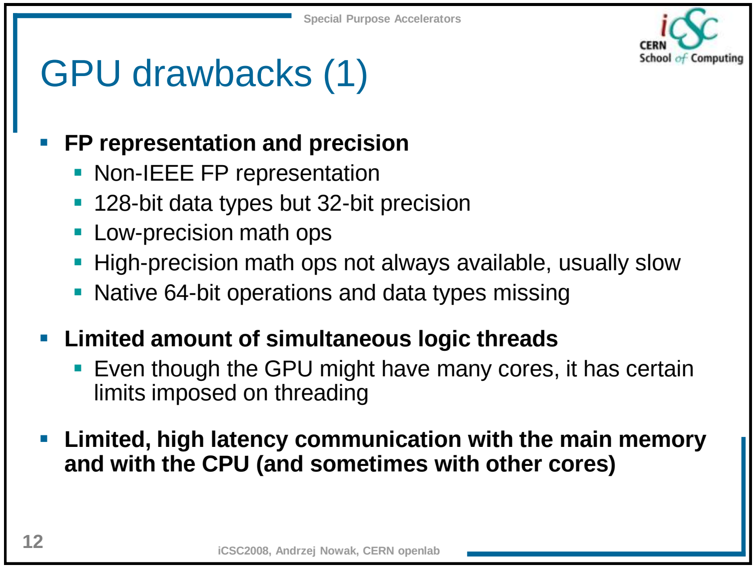
Common GPGPU operations

- **Stream filtering**
  - Removing items from a group based on certain criteria
- **Mapping**
  - Run a function on elements inside a group
- **Reducing**
  - Perform calculations on a stream and yield a reduced result
- **Scatter and gather**
- **Sorting**
  - Sorting networks
- **Searching**
  - Parallel searches



Which problems can benefit from GPGPU?

- **Algorithms and applications using the Fast Fourier Transform**
- **Audio processing and DSP**
- **Digital image and video processing**
- **Raytracing**
- **Weather forecasting**
- **Neural networks**
- **Molecular modeling**
- **Database operations**
- **Cryptography and cryptoanalysis**



GPU drawbacks (1)

- **FP representation and precision**
  - Non-IEEE FP representation
  - 128-bit data types but 32-bit precision
  - Low-precision math ops
  - High-precision math ops not always available, usually slow
  - Native 64-bit operations and data types missing
- **Limited amount of simultaneous logic threads**
  - Even though the GPU might have many cores, it has certain limits imposed on threading
- **Limited, high latency communication with the main memory and with the CPU (and sometimes with other cores)**

GPU drawbacks (2)

- **Heat problems**
  - Modern cards can easily achieve 150W
  - Projected Larrabee power is said to be around 150-200W
- **Feeding the beast**
  - A modern CPU is required to feed a modern GPU at full speed
- **Rudimentary development tools**
  - General purpose libraries and utilities are often absent
  - Lacking especially in higher-level languages
- **Vector processor**
  - Limited scientific applications
  - Limited flexibility
- **Data control paths unprotected, fault handling lacks robustness**

FEEDING THE BEASTS

Programming GPUs

Development kits for GPUs - CUDA

- **CUDA stands for “Compute Unified Device Architecture”**
- **General purpose development kit for the G80 chip**
- **C supported**
- **Open64 based compiler**
- **CUDA software includes BLAS and FFT libraries; areas of application:**
  - Parallel bitonic sort
  - Matrix multiplication
  - Matrix transposition
  - Performance profiling using timers
  - Parallel prefix sum of large arrays
  - Image convolution
- **Deviations from the IEEE floating point standard**

Development kits for GPUs - CTM

- **ATI/AMDs counterpart to CUDA**
- **CTM stands for “Close To Metal”**
  - A little bit too close, perhaps…
- **Good access to the native instruction set and memory**
- **Supported by Radeon cards (from R580 on) and FireStream processors (based on the X1900)**
- **AMD claims CTM delivers 8x the performance of “traditional” GPGPU methods – OpenGL or DirectX**
- **Open source**

Development kits for GPUs – Rapid Mind (1)

- **Multi-core and GPGPU development platform**
- **Mostly for graphics processing**
- **An API library for C++**

Development kits for GPUs – Rapid Mind (2)

- **Features**
  - Code optimization
  - Automatic load balancing
  - Data management and diagnostics
- **Backends:**
  - Intel and AMD CPUs
  - NVIDIA and ATI/AMD GPUs
  - IBM Cell Processor
    - Cell Blade
    - Cell Accelerator Board
    - Sony Playstation 3

Development kits for GPUs - Brook

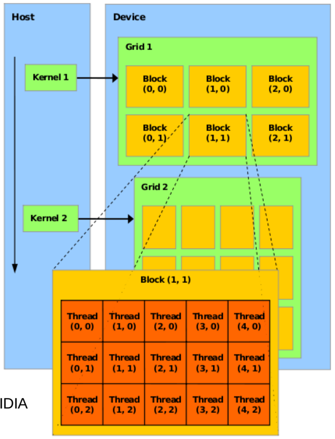
- **Stanford University’s GPGPU library**
- **A derivative of ANSI C**
- **Backends: OpenGL 1.3+, DirectX 9+, CTM**
- **Runs on Linux, Windows, Mac OS X; BSD license**
- **410 GFLOPS cited (DX9, ATI HD 2900 XT)**
- **Development picked up again in 2007**

INSIDE THE HARDWARE

A peek into commodity gaming gear of today and tomorrow

NVIDIA G80

- **Stream processor developed by NVIDIA**
- **Moved away from traditional GPU design**
  - Uniform shader model
  - DirectX 10 support
- **128 stream processors**
- **330 GFLOPS peak**
- **Second generation: G92**


Host
Device
Grid 1
Kernel 1
Block (0, 0)
Block (1, 0)
Block (2, 0)
Block (0, 1)
Block (1, 1)
Block (2, 1)
Grid 2
Kernel 2
Block (1, 1)
Thread (0, 0)
Thread (1, 0)
Thread (2, 0)
Thread (3, 0)
Thread (4, 0)
Thread (0, 1)
Thread (1, 1)
Thread (2, 1)
Thread (3, 1)
Thread (4, 1)
Thread (0, 2)
Thread (1, 2)
Thread (2, 2)
Thread (3, 2)
Thread (4, 2)

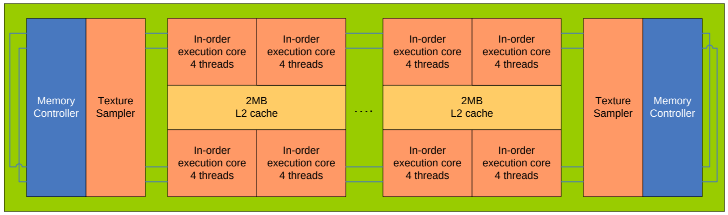
Graphics: NVIDIA

AMD FireStream

- **Stream processor developed by ATI**
- **Targets not only gamers, but the HPC community as well**
- **A FireStream general purpose extension card exists**
  - Can be used as a floating point coprocessor
- **Specs:**
  - 48 pixel shaders
  - 600 MHz clock
- **Part of AMD Torrenza**

Intel Larrabee

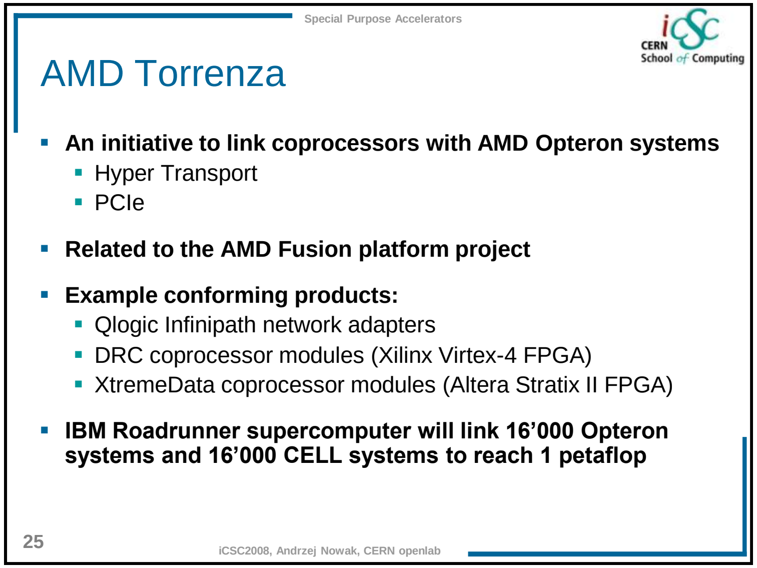
- **45nm process, 1.7 – 2.5 GHz, > 150W**
- **16-24 in order cores for pixel/vertex shading**
  - 4 threads per core, capable of 2 double-precision FP ops per cycle


Memory Controller
Texture Sampler
In-order execution core 4 threads
In-order execution core 4 threads
2MB L2 cache
In-order execution core 4 threads
In-order execution core 4 threads
….
In-order execution core 4 threads
In-order execution core 4 threads
2MB L2 cache
In-order execution core 4 threads
In-order execution core 4 threads
Texture Sampler
Memory Controller

SPECULATIVE INFORMATION. Source: ArsTechnica

ClearSpeed cards

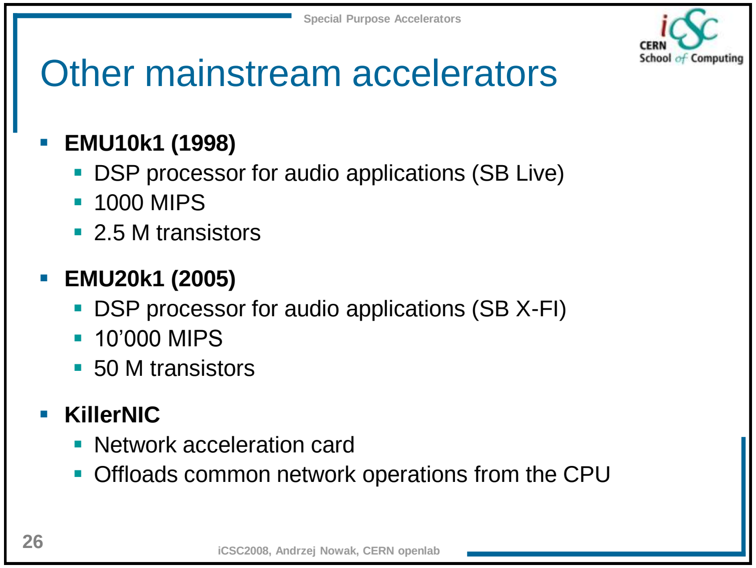
- **Attached to the PCI bus (or PCIe)**
- **Central point: the CSX600 chip**
  - 96 compute engines
  - 64-bit floating point capability
  - Full IEEE floating point compliance
- **2 chips, 80 GFLOPS per board**
- **They claim to have the highest FLOP/Watt (2GFLOP/Watt)**
  - 30 Watts per board
- **Toolkit available**
  - C-based compiler
  - Development tools – assembler, debugger, profiling tools
  - BLAS, LAPACK available



CERN School of Computing

AMD Torrenza

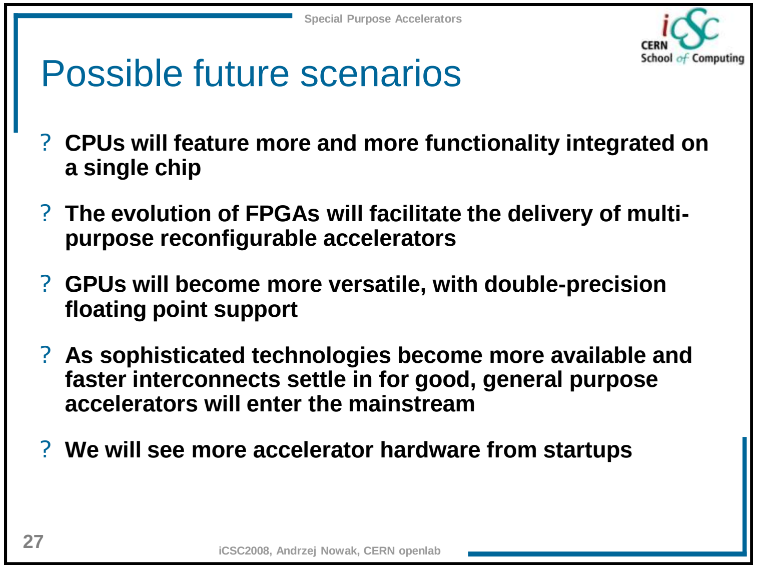
- **An initiative to link coprocessors with AMD Opteron systems**
  - Hyper Transport
  - PCIe
- **Related to the AMD Fusion platform project**
- **Example conforming products:**
  - Qlogic Infinipath network adapters
  - DRC coprocessor modules (Xilinx Virtex-4 FPGA)
  - XtremeData coprocessor modules (Altera Stratix II FPGA)
- **IBM Roadrunner supercomputer will link 16’000 Opteron systems and 16’000 CELL systems to reach 1 petaflop**



CERN School of Computing

Other mainstream accelerators

- **EMU10k1 (1998)**
  - DSP processor for audio applications (SB Live)
  - 1000 MIPS
  - 2.5 M transistors
- **EMU20k1 (2005)**
  - DSP processor for audio applications (SB X-FI)
  - 10’000 MIPS
  - 50 M transistors
- **KillerNIC**
  - Network acceleration card
  - Offloads common network operations from the CPU



CERN School of Computing

Possible future scenarios

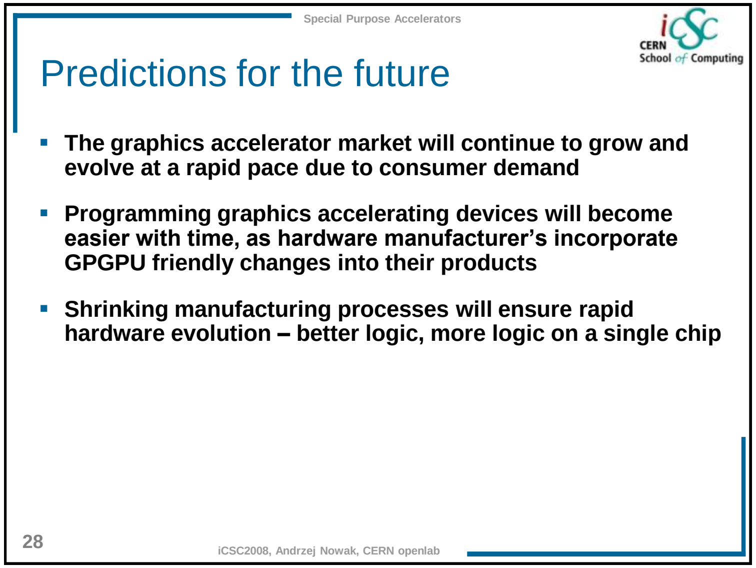
? **CPUs will feature more and more functionality integrated on a single chip**

? **The evolution of FPGAs will facilitate the delivery of multi-purpose reconfigurable accelerators**

? **GPUs will become more versatile, with double-precision floating point support**

? **As sophisticated technologies become more available and faster interconnects settle in for good, general purpose accelerators will enter the mainstream**

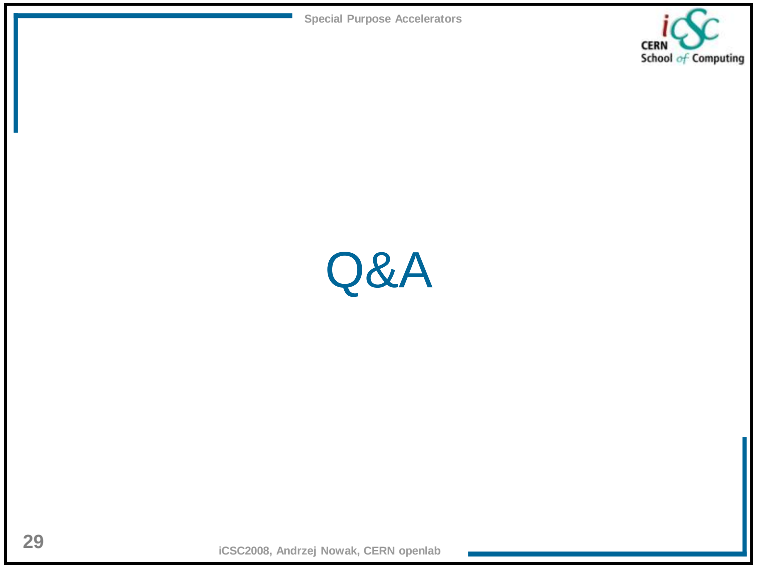
? **We will see more accelerator hardware from startups**



CERN School of Computing

Predictions for the future

- **The graphics accelerator market will continue to grow and evolve at a rapid pace due to consumer demand**
- **Programming graphics accelerating devices will become easier with time, as hardware manufacturer’s incorporate GPGPU friendly changes into their products**
- **Shrinking manufacturing processes will ensure rapid hardware evolution – better logic, more logic on a single chip**



CERN School of Computing

Q&A

iCSC
CERN
School of Computing

LECTURE 5

Multicores at work: The CELL Processor

Monday 3 March 2008			
16:30 17:25	Lecture 5	This lecture will give a deeper insight into one of the special-purpose accelerators, the CELL processor, that is actively being investigated for HPC. Important topics: • (Ab)Using your Playstation • Cell Clusters • The Roadrunner Project This lecture will show the basic steps to CELL programming in a simple and easy to follow manner such that no prerequisite is necessary to follow the lecture. **Audience - Pre-requisite** The aim is to show the practical realization of theoretical concepts introduced in lectures 2, 3 and 4.	**Iris Christadler**

Theme: Towards Reconfigurable HPC
Lecture **5**

Multicores at Work: The CELL Processor

Iris Christadler

Leibniz Supercomputing Centre

Inverted CERN School of Computing, 3-5 March 2008

Introduction

- **Objectives:**
  - Explain the peculiarities of the Cell architecture
  - Show examples of Cell and Playstation clusters
  - Introduce the Roadrunner project
  - Give an overview of software development for Cell

Cell Broadband Engine Architecture

ARCHITECTURAL DETAILS

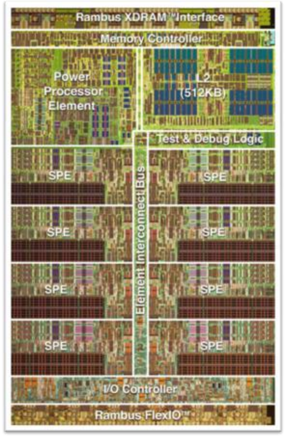


Image: IBM

STI CBEA (= Cell)

- **developed under Sony/Toshiba/IBM (STI) efforts**
- **current Cell chip is used in Sony's PlayStation3 (PS3)**
- **radical departure from previous mainstream processors**
- **8+1 way heterogeneous parallel processor**
- **high performance @ 3.2 GHz**
  - 204.8 GF/s single precision
  - 14.63 GF/s double precision
- **good ratio between performance and floor space as well as power consumption**
- **"nearly" IEEE-754 conform**

A Closer Look at the Hardware

- **1 Power Processing Element (PPE)**
  - general-purpose 64-bit RISC processor (PowerPC)
  - two way hardware multithreading
  - on-chip L1/L2 cache
    - 512 KB L2 cache
    - 32 KB L1 instruction cache
    - 32 KB L1 data cache
- **8 Synergistic Processor Elements (SPEs)**
  - special-purpose RISC processor
  - 128-bit SIMD
  - 256 KB local Memory
- **connected with the Element Interconnect Bus (EIB)**

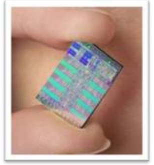


Image: IBM

A Closer Look at the Hardware (2)

SPE: Synergistic Processor Element
SPU: Synergistic Processor Unit
PPE: Power Processing Element

128-bit Vector Engine

256 KB local memory
(LS= Local Store)

Direct Memory Access Engine
(25.6 GB/s)

Chip Interconnect
(EIB= Element Interconnect Bus)

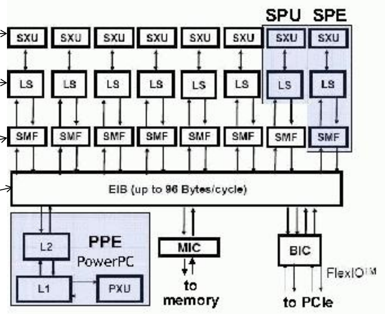


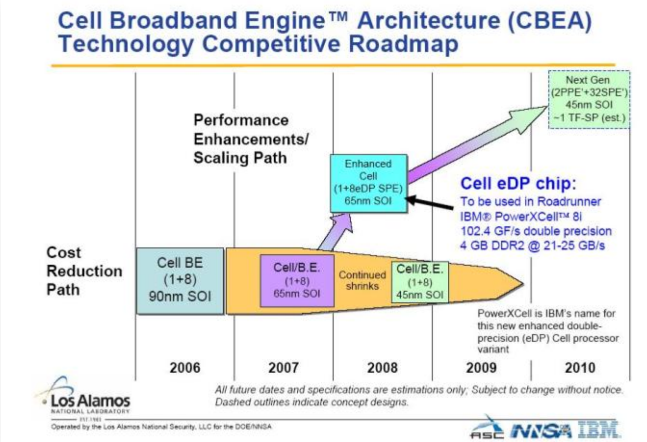
Image: IBM

IBM QS20 Cell Blade

- **2 3.2 GHz Cell BE processors**
- **1 GB Memory (512 per processor)**
- **40 GB hard disk**
- **Dual Gigabit Ethernet**
- **InfiniBand 4x adapters**
- **Fedora Linux available**

Image: IBM

"…is especially suitable for computationally intense, high performance workloads across a number of industries including *digital media, medical imaging, aerospace, defense and communications*." (Advertisement claim)



Cell Broadband Engine™ Architecture (CBEA) Technology Competitive Roadmap

Performance Enhancements/ Scaling Path
Cost Reduction Path
Cell BE (1+8) 90nm SOI
Cell/B.E. (1+8) 65nm SOI
Continued shrinks
Cell/B.E. (1+8) 45nm SOI
Enhanced Cell (1+8eDP SPE) 65nm SOI
Next Gen (2PPE'+32SPE') 45nm SOI ~1 TF-SP (est.)
2006
2007
2008
2009
2010
Los Alamos NATIONAL LABORATORY
Operated by the Los Alamos National Security, LLC for the DOE/NNSA
ASC NNSA IBM

Cell eDP chip:
To be used in Roadrunner
IBM® PowerXCell™ 8i
102.4 GF/s double precision
4 GB DDR2 @ 21-25 GB/s

PowerXCell is IBM's name for this new enhanced double-precision (eDP) Cell processor variant

All future dates and specifications are estimations only; Subject to change without notice. Dashed outlines indicate concept designs.

Can your Playstation save the world?

(AB)USING YOUR PLAYSTATION 3

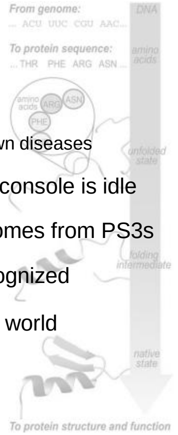
Installing Linux on PS3

- **Different Linux flavours available**
  (Fedora, Yellow Dog, …)
- **Possibility to use a**
  - bootable Linux from CD
  - permanent dual-boot installation setup
- **It's really simple**
  - download the necessary stuff
    (Linux images, PS3 add-on, kernel sources)
  - PS3: Settings -> System Settings -> Format Utility
  - PS3: Settings -> System Settings -> Install Other OS
  - PS3: Settings -> System Settings -> Default System
  - Install Linux

Detailed Installation Instructions e.g. at www.cellperformance.com

Folding@Home

- **Distributed computing project**
  [folding.stanford.edu]
- **Simulates protein folding**
  in order to understand and find cures for many well known diseases
- PS3 is able to perform computations while the console is idle
- ~75% of the combined computational power comes from PS3s
- Thanks to PS3s Folding@Home has been recognized by the Guinness World Records as the *most powerful distributed network* in the world
  - September 16, 2007, project surpassed 1 PF
  - September 23, 2007, PS3s alone reached 1PF



PLAYSTATION/ CELL CLUSTERS

Overview

- **Two ways of using Cell in HPC**
  - combined in clusters of Cells
  - attached as accelerators to commodity CPUs
- **Two different kind of Cell clusters**
  - “cheap” PS3 clusters with some drawbacks
  - more expensive IBM Blade Server Solution
- **Drawbacks of PlayStation Clusters**
  - 256 MB main memory is not enough
  - Slow network becomes the bottleneck
  - Linux is running under a hypervisor
  - PS3 comes with a special edition NVIDIA graphics card but unfortunately does not allow access to this resource

Univ. of Tennessee: PS3 Cluster

- **PlayStation3 Cluster**
  - 4 Playstations
  - GigaBit Ethernet switch
- **Power Consumption:**
  - ~ 800 Watt
  - ~ 0.5 -1 GF/Watt (sp)
- **Performance:**
  - 800 GF (sp)
- **Price:**
  - ~ 2400 $
  - ~ 330 MF/$

Image: Univ. of Tennessee

JUICE: Juelich Initiative Cell Cluster

- **Configuration**
  - 2 Chassis
  - each equipped with 6 Q20 cell blades
  - each of them include 2 Cell BEs
  - Frontend: Xeon
- **Networks**
  - Gigabit Ethernet
  - InfiniBand
- **Performance** (single prec., non IEEE)
  - 204.8 GF per Cell
  - 4.9 TF total

Image: FZJ

Running fast in New Mexico

THE ROADRUNNER PROJECT

http://www.lanl.gov/roadrunner/

The Roadrunner Project

- **Los Alamos National Laboratory (LANL)**
  Department of Energy (DoE)
- **First petaflop system worldwide**
  Peak Performance will be 1.33 PF
- **A hybrid Opteron-Cell system**
  Cells used as accelerators
- **due in Q3/2008**
- **Early installation of a migration system to resolve programming issues**

„Will HPC turn away from homogeneous architectures and go hetero?"
[http://www.hpcwire.com/hpc/897414.html]

„IBM Unlocks the Cell"
[http://www.hpcwire.com/hpc/893353.html]

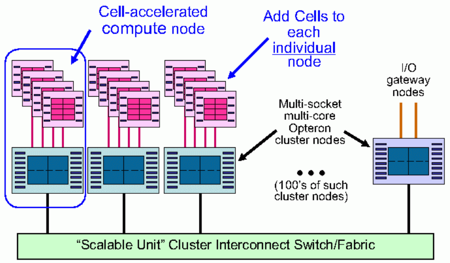
Facts and Figures

- **Cluster of:**
  - 18 Connected Units (CU)
  - 6,912 AMD dual-core Opterons (1.8 GHz)
  - 12,960 IBM PowerXCells (~100 GF dp peak each)
- **Performance:**
  - 9.8 teraflops peak (Opteron)
  - 1.33 petaflops peak (Cell eDP)
  - designed for a sustained 1.0 petaflop LINPACK performance
  - side note: This is an efficiency of 75%

Connected Unit Specification

- **360 1.8 GHz dual-core Opterons**
  - 2.8 TF DP peak Opteron
  - 1.5 TB Opteron memory
- **720 3.2 GHz Cell eDP chips**
  - 73.7 TF peak Cell
  - 2.9 TB Cell memory
  - 15.4 TB/s Cell memory BW
- **192 InfiniBand (IB) 4x DDR cluster links**
  - 768 GB/s aggregate BW (bi-dir)
  - 384 GB/s bi-section BW (bi-dir)
- **24 10 GigE I/O links on 12 I/O nodes**
  - 24 GB/s aggregate I/O BW (uni-dir) (IB limited)

Roadrunner Phase 3 is Cell-accelerated, not a cluster of Cells


Cell-accelerated compute node
Add Cells to each individual node
I/O gateway nodes
Multi-socket multi-core Opteron cluster nodes
• • •
(100's of such cluster nodes)
"Scalable Unit" Cluster Interconnect Switch/Fabric

Node-attached Cells is what makes Roadrunner different!

Los Alamos NATIONAL LABORATORY EST. 1943
Operated by the Los Alamos National Security, LLC for the DOE/NNSA

Slide: LANL

Power Consumption, Footprint,…

- **Power Consumption**
  - 3.9 MW Power
  - 0.35 GF/Watt (peak)
  - 0.25 GF/Watt (linpack)
- **Footprint**
  - 296 racks
  - 5500 ft^2
- **OS & SW**
  - OS: RHEL and Fedora Linux
  - IBM SDK for Multicore Acceleration
  - Cell compilers, libraries, SDK tools

IBMs Blue Gene

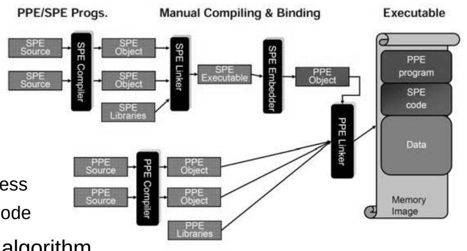
- **IBMs Blue Gene series is currently the second guess if you are talking about petaflop systems**
- **Rank 1 and 2 in the top500 list from Nov 2007 are BlueGenes**
  - Installation at LLNL with nearly 600 TF peak
  - Installation in Juelich with 220 TF peak
- **David Turek (vice president Deep Computing at IBM) in an hpcwire interview mid 2006:**
  "[…] if you look at Blue Gene today, the only thing that separates you from the deployment of a petaflop system is money. The future designs factor in a whole lot of other things - not only how you make a petaflop affordable, but also how do you open the aperture to an enhanced set of applications. […]
  And it's not at all in conflict with what we're doing here with Roadrunner because they're different programming models. For us, that's a key point of differentiation. Right now it looks like they may serve different application sets differently. "
  [http://www.hpcwire.com/hpc/893353.html]

Unfortunately not that easy ☺

The faster your code, the longer the development phase…

SOFTWARE DEVELOPMENT FOR CELL

Software Development for Cell



Highest performance: use platform specific instrinsics
Higher productivity using automatic simdization compiler

- **Examples**
  - SpMV
    - one month learning process
    - 600 lines of code
  - Stencil-based algorithm
    - still required one week of work
    - 250 lines of code (scalar version is 15 lines)

[Examples taken from "The Potential of the Cell Processor for Scientific Computing"]

Cell Software Development

- **Cell full system simulator for free download**
  [http://www.alphaworks.ibm.com/tech/cellsystemsim]
- **Important feature I: SIMD**
  To enable vectorization use for example loop-unrolling
- **Important feature II: Software-controlled memories**
  Data movement between registers, local memory and external DRAM is explicitly controlled by the application
- **Use intrinsics**
  Start with “Cell Broadband Engine Tutorial”, end up with the ultimate reference “C/C++ Language Extensions for Cell Broadband Engine Architecture”

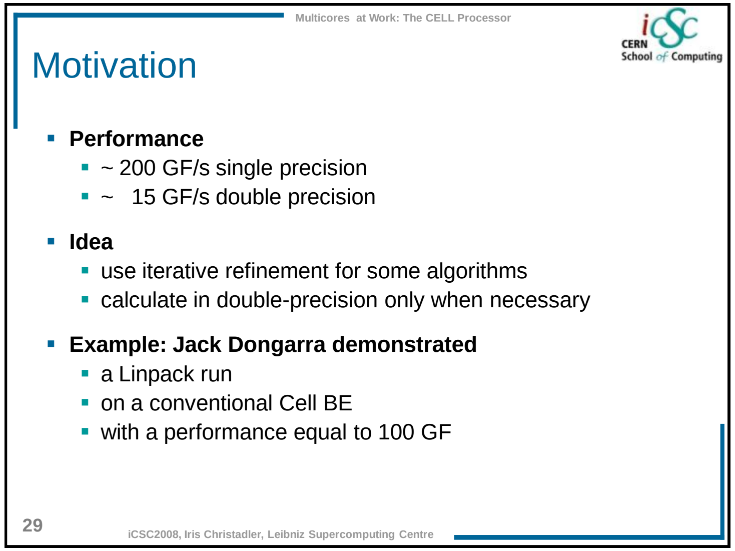
Parallelization Models

- **Function offload model**
  main application executes on PPE,
  performance critical functions are offloaded to SPE
- **Parallelization models for the SPEs**
  application is mainly executed on the 8 SPEs
  - *Task Parallelism*
    independent tasks scheduled on each SPE
  - *Pipelined Parallelism*
    large blocks are passed from one SPE to the next
  - *Data Parallelism*
    processors perform identical computation on distinct data (most common, used in many scientific applications)

Software Development Kits

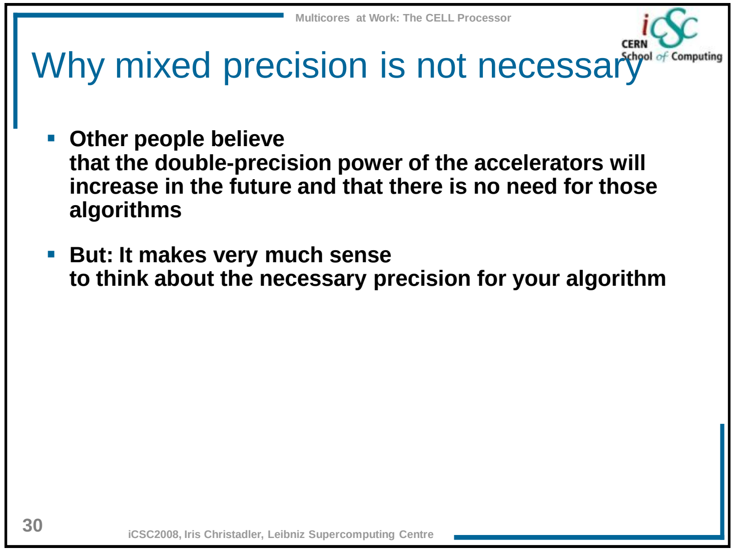
- **IBM SDK for Multicore Acceleration Version 3.0**
- **CorePy**
- **Octopiler**
- **Rapid-Mind**
- **(PeakStream)**
- **Cell Superscalar (CellSs)**
- **The Sequoia Language**
- **Mercury Multi-Core Framework**
- **…**

MIXED PRECISION PROGRAMMING FOR CELL



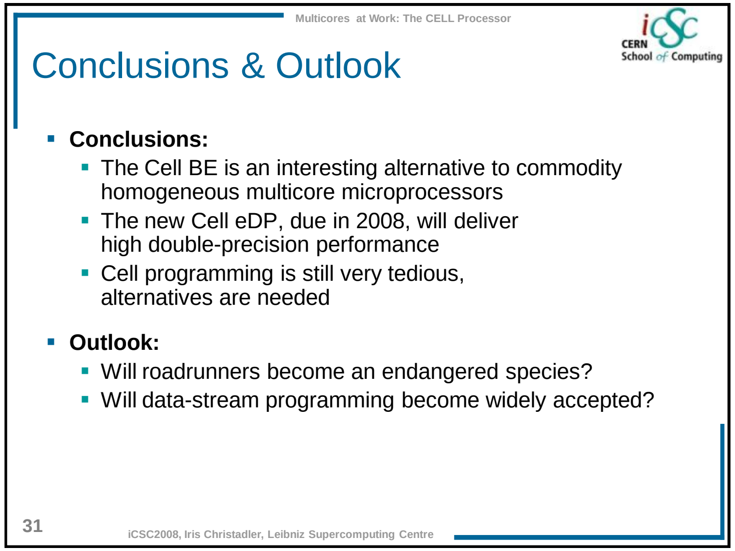
Motivation

- **Performance**
  - ~ 200 GF/s single precision
  - ~ 15 GF/s double precision
- **Idea**
  - use iterative refinement for some algorithms
  - calculate in double-precision only when necessary
- **Example: Jack Dongarra demonstrated**
  - a Linpack run
  - on a conventional Cell BE
  - with a performance equal to 100 GF



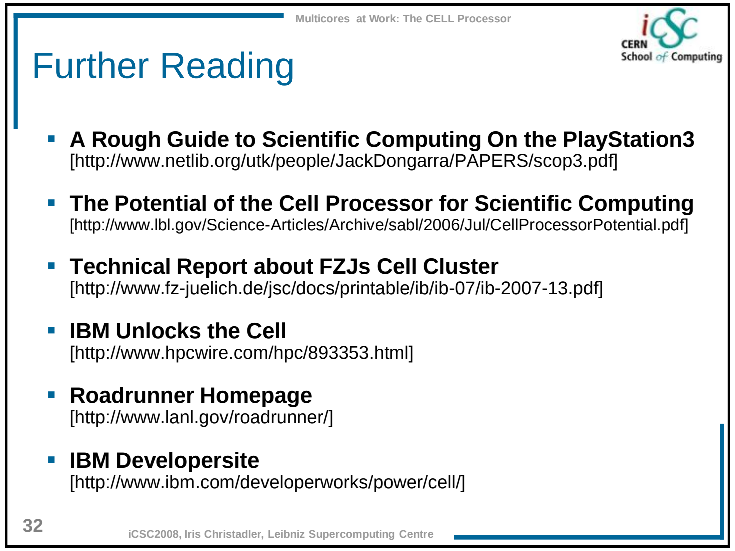
Why mixed precision is not necessary

- **Other people believe**
  that the double-precision power of the accelerators will increase in the future and that there is no need for those algorithms
- **But: It makes very much sense**
  to think about the necessary precision for your algorithm



Conclusions & Outlook

- **Conclusions:**
  - The Cell BE is an interesting alternative to commodity homogeneous multicore microprocessors
  - The new Cell eDP, due in 2008, will deliver high double-precision performance
  - Cell programming is still very tedious, alternatives are needed
- **Outlook:**
  - Will roadrunners become an endangered species?
  - Will data-stream programming become widely accepted?



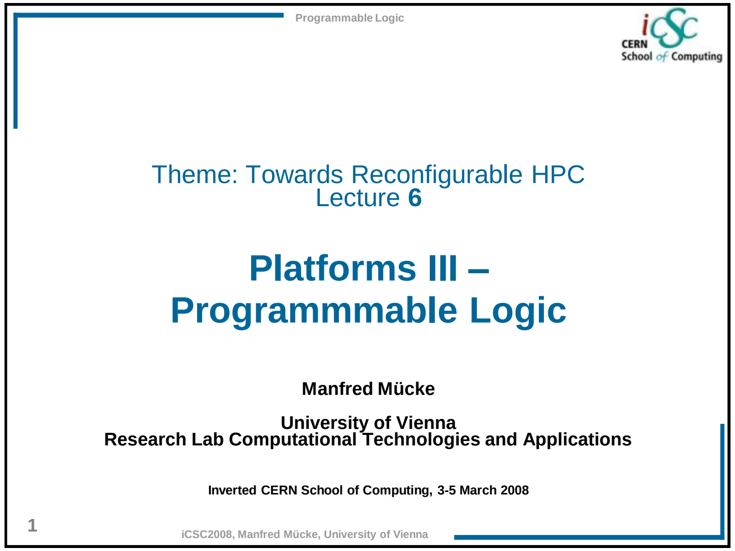
Further Reading

- **A Rough Guide to Scientific Computing On the PlayStation3**
  [http://www.netlib.org/utk/people/JackDongarra/PAPERS/scop3.pdf]
- **The Potential of the Cell Processor for Scientific Computing**
  [http://www.lbl.gov/Science-Articles/Archive/sabl/2006/Jul/CellProcessorPotential.pdf]
- **Technical Report about FZJs Cell Cluster**
  [http://www.fz-juelich.de/jsc/docs/printable/ib/ib-07/ib-2007-13.pdf]
- **IBM Unlocks the Cell**
  [http://www.hpcwire.com/hpc/893353.html]
- **Roadrunner Homepage**
  [http://www.lanl.gov/roadrunner/]
- **IBM Developersite**
  [http://www.ibm.com/developerworks/power/cell/]

LECTURE 6

Platforms III - Programmable Logic

Tuesday 4 March 2008			
09:00 09:55	Lecture 6	This lecture introduces programmable logic from a hardware point of view (with a bias towards their potential use in HPC). Important topics: • Introduction to FPGAs. • Hardware Description Languages. • Understanding FPGA performance figures. • Gap between potential computing performance and programmer productivity. **Audience - Pre-requisite** The lecture is designed for students with no prior knowledge in programmable logic. It does not build upon any of the preceding lectures. It is however the basis for all further lectures and is highly recommended to all students including ones having prior knowledge in programmable logic.	**Manfred Muecke**



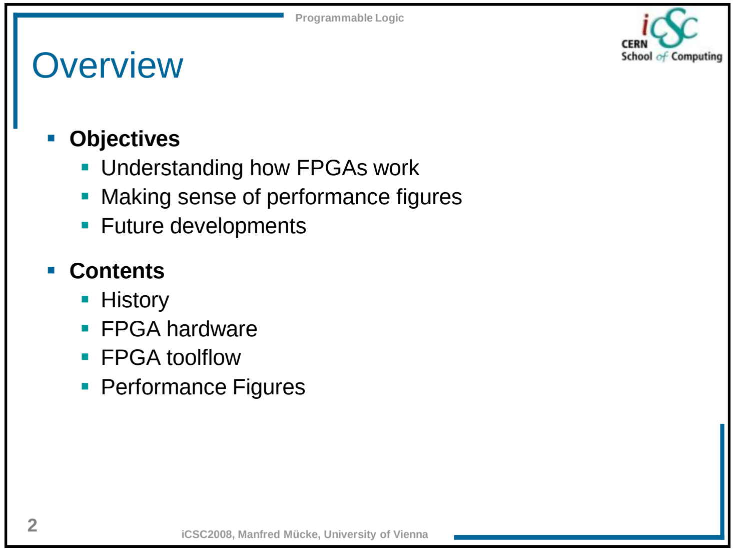
Theme: Towards Reconfigurable HPC
Lecture **6**

Platforms III –
Programmmable Logic

Manfred Mücke

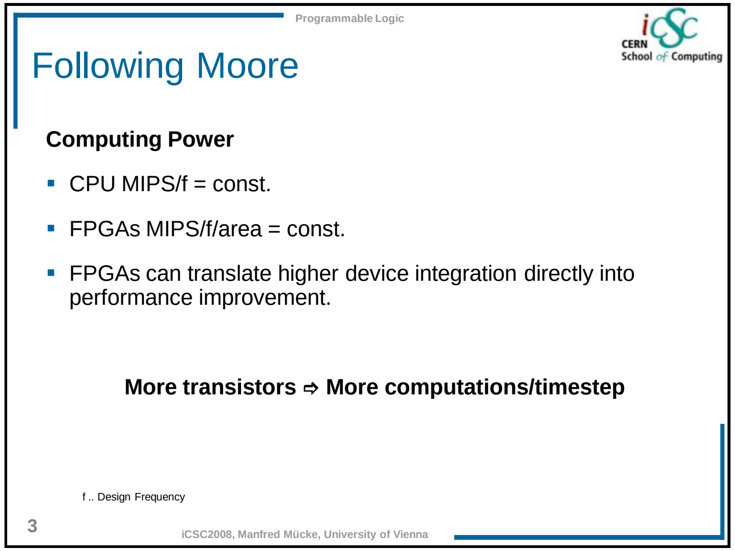
University of Vienna
Research Lab Computational Technologies and Applications

Inverted CERN School of Computing, 3-5 March 2008



Overview

- **Objectives**
  - Understanding how FPGAs work
  - Making sense of performance figures
  - Future developments
- **Contents**
  - History
  - FPGA hardware
  - FPGA toolflow
  - Performance Figures



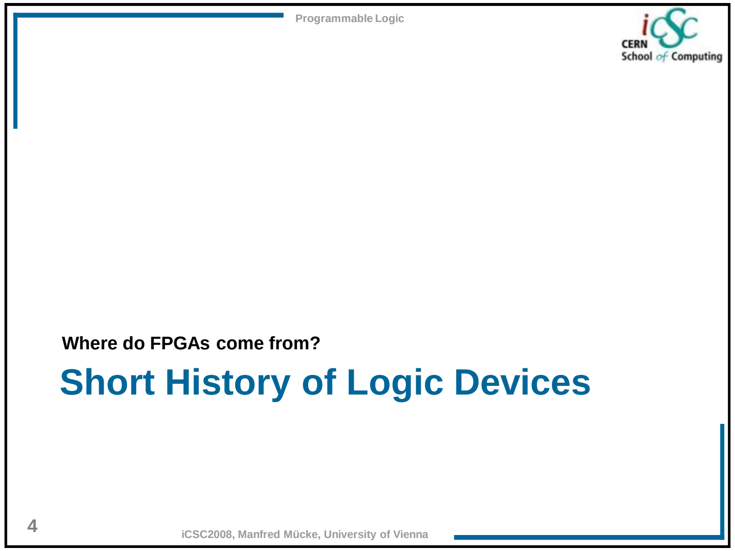
Following Moore

Computing Power

- CPU MIPS/f = const.
- FPGAs MIPS/f/area = const.
- FPGAs can translate higher device integration directly into performance improvement.

More transistors ⇨ More computations/timestep

f .. Design Frequency



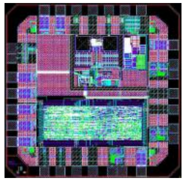
Where do FPGAs come from?

Short History of Logic Devices

ASICs

ASIC = Application-Specific Integrated Circuit

- Full-custom design
- Mixed-signal possible
- Highly efficient
- Matches exactly your needs
- Expensive (NRE)
- Extensive know-how required
- Long design cycles



QPLL ASIC, CERN MIC

Standard Logic Wired-Up

CPLD

- **CPLD** = Complex Programmable Logic Device
  = Logic Device + Configuration Memory
- Available since 1970's (and still evolving)
- Macro cells implement disjunctive normal form
  (A and B) or (A and (not C))
- Registered Outputs
- Different technologies to keep configuration (now Flash)
  +Low cost
  +Non-volatile configuration
  +Predictable timing characteristics
  - Limited functional complexity

FPGAs

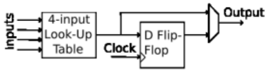
- **FPGA** = Field-Programmable Gate Array
- Invented 1984, evolved from CPLDs
  CPLDs separate logic and register
  FPGAs combine logic and register
- Emphasis on interconnect of small configurable blocks
  + Arbitrary complex functionality
  + Most area for interconnect
  - Variable timing
  - Demanding tool requirements
- ⇨ The "winning" (most versatile) architecture

How does it work?

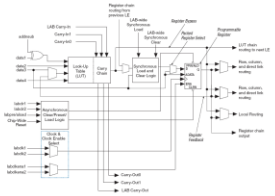
Introduction to FPGAs

Look-Up Tables (LUT)

- Any boolean function of n variables can be implemented by a memory of size 2^n ⇨ LUT = small memory
- LUTs in FPGAs use typically 4 inputs.
  ⇨ 1 LUT needs 16 configuration bits.

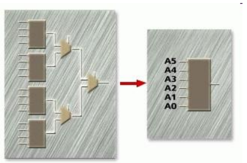


- Add a single-bit register
- Memory is typically SRAM
  (needs to be written at power-up)
- Find a marketing name
  Xilinx: Logic Cells
  Altera: Logic Elements
- Fit as many as possible (> 200.000)

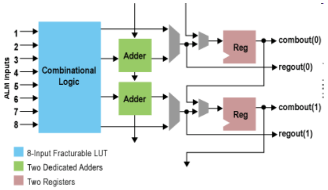


Bigger LUTs

- Balance routing and logic granularity
- Routing dominates ⇨ LUTs can grow



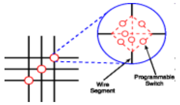
Picture: Xilinx

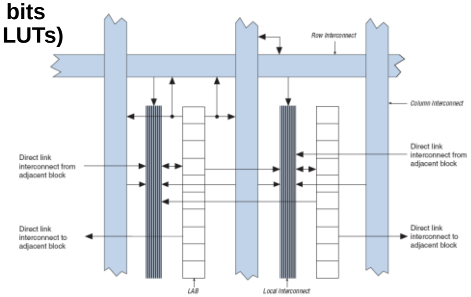


Picture: Altera

Interconnect

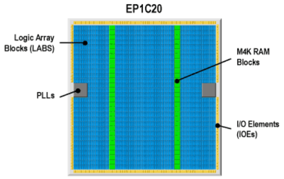
- **Guarantee that all (many) LUTs can be interconnected**
- **Configurable Switches**
- **Configuration bits in SRAM (like LUTs)**
- **Most area is interconnect**
- **Huge delay variations**




Row Interconnect
Column Interconnect
Direct link interconnect from adjacent block
Direct link interconnect from adjacent block
Direct link interconnect to adjacent block
Direct link interconnect to adjacent block
LAB
Local Interconnect

Basic FPGA

- **LUTs + Interconnect**



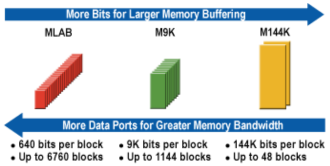
Altera Cyclone				Xilinx Spartan-IIE		
	EP1C6	EP1C20			XC2S 300E	XC2S 400E
LE	5.980	20.060		LC	6.912	10.800
Pins	185	301		Pins	329	410

Memories

- LUTs come with 1bit of memory only.
- If you need some memory, you waste a lot of LUTs ☹

⇨ Integrate some dedicated memory blocks

Each memory is accessible in parallel
⇨ Huge memory bandwidth possible



Picture: Altera (StratixIII TriMatrix)

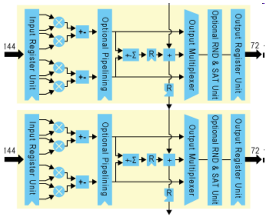
Hard IP Blocks

- Some functionality maps bad into LUTs (Multiplication)
- Some functionality is required by most designs (Addition)

⇨ Provide hardwired macros (IP) for frequently requested functions

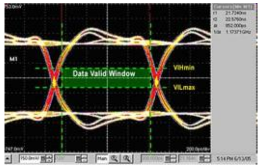
⇨Multiplier
⇨Adder
⇨MAC
⇨ALUs (DSP blocks**)**

+ runs much faster (~500MHz)
+ costs no routing resources



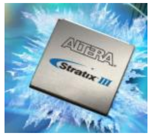
I/O

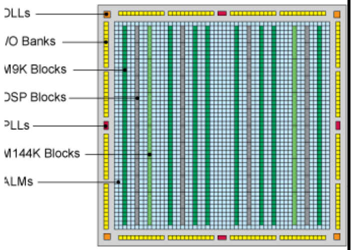
- **Pins connect the FPGA to the outside world**
- **Originally low-speed TTL**
  Today: Everything ☺
  Especially:
  LVDS (SPI-4.2, SFI-4, SGMII, Utopia IV, 10 GbE XSBI, RapidIO®, SerialLite.),
  RAM (DDR SDRAM, DDR2 SDRAM, DDR3, QDRII, QDRII+, RLDRAMII)
- **High pin-count (> 1000)**



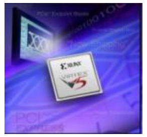
Contemporary FPGAs

- **Altera Stratix III**
- **Cyclone III**

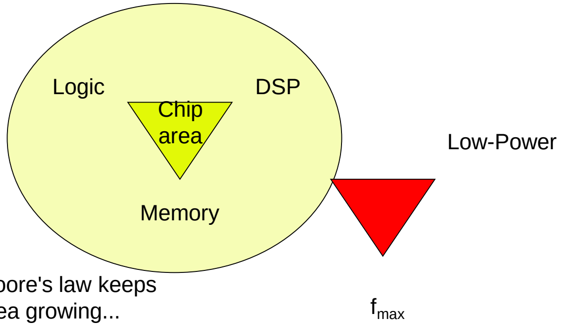




- **Xilinx Virtex 5**
- **Spartan 3**



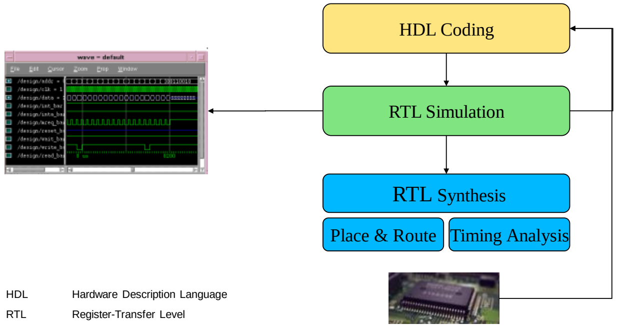
FPGA Device Range



Moore's law keeps area growing...

...but makes timing and power closure harder and harder.

FPGA Design Flow



HDL	Hardware Description Language
RTL	Register-Transfer Level

VHDL

- Invented for description of digital circuits (US DoD, 1980s)
- Based on Ada programming language
- Powerful, but complex modeling language.
- Subset is synthesizeable (IEEE Std 1076.6-2004) (mostly RTL)
- Powerful tools available for all FPGAs.
  ⇨ Simulator
  ⇨ Synthesizer
- Writing/Debugging/Optimizing VHDL is a tedious task!
  ⇨ For high-speed designs, there is no better way (yet).

Do you know, what you get?

FPGA Performance Figures

Performance Figures

- FPGA marketing likes to impress with big numbers!

The GMACs

- Performance figure from the DSP world (1 MAC/cycle)
- DSP @ 500MHz ⇨ 0.5GMAC/s
  You can achieve this with a loop
- FPGA (64 MAC blocks @ 500 MHz) ⇨ 32GMAC/s WOW!
  1) Watch the bitwidth!
  2) Can your design run at 500MHz, too?
  3) Can you deliver data at 500MHz*64*xbits?
  4) Does your design cover all FPGA resources?

Performance Figures

Total Memory Bits

- FPGAs have distributed memories of different sizes
  Every LUT can serve as mini-memory
  1. Know your desired size
  2. Then ask: How many blocks of size x.

Same for **accumulated memory bandwidth**

Logic resources: Compare with similar designs

- **FPGAs are not filled > 80%**
- **FPGAs run typically at ~150MHz**

Designing your own CPU

Softcores

SoftCores

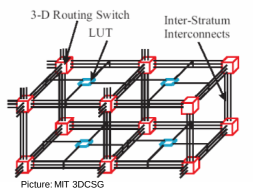
- **Softcores provide CPU-functionality implemented on FPGA**
- Optimize your CPU!
- Don't worry about CPU long-term availability.
- Cheap and slow (100-200MHz)
- But: Embedded Linux on Softcore ☺!
- Needs Tight integration with software toolchain

Where will FPGAs got to?

FPGA Outlook

3D Interconnect

- 2D Interconnect consumes most of an FPGAs **area**.
- The longer, the harder to route.
  Routing (and power) is the limiting factor.
- 3D provides "shortcuts",
  enables higher logic and routing density.
- **Manufacturable?**

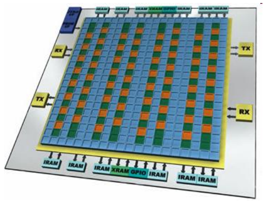


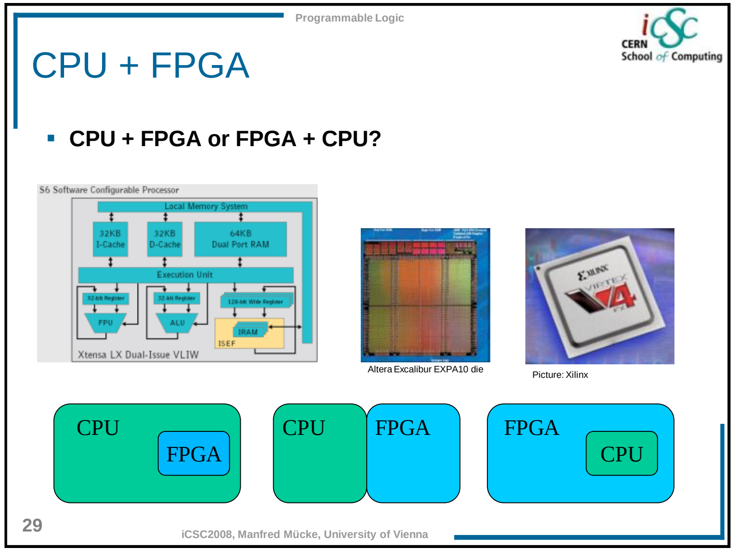
Picture: MIT 3DCSG

Mathstar

- Idea: Provide **coarser functional blocks**.
- Replace a set of LUT + interconnect with dedicated blocks (5/16bit)
- Configurable Blocks:
  - ALU
  - 16x16 MAC
  - Register
- Configurable Interconnect

  + runs at 1GHz
  + reduced routing overhead
  ➩ RISC Manycore



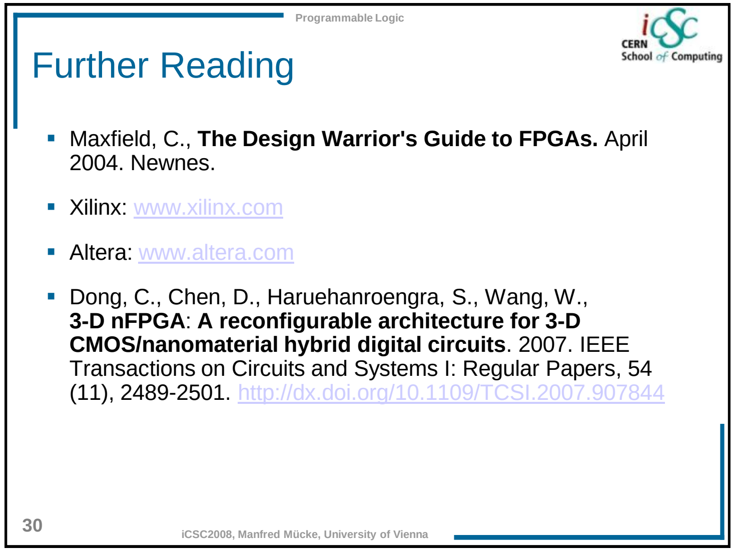


CPU + FPGA

- **CPU + FPGA or FPGA + CPU?**

Altera Excalibur EXPA10 die

Picture: Xilinx



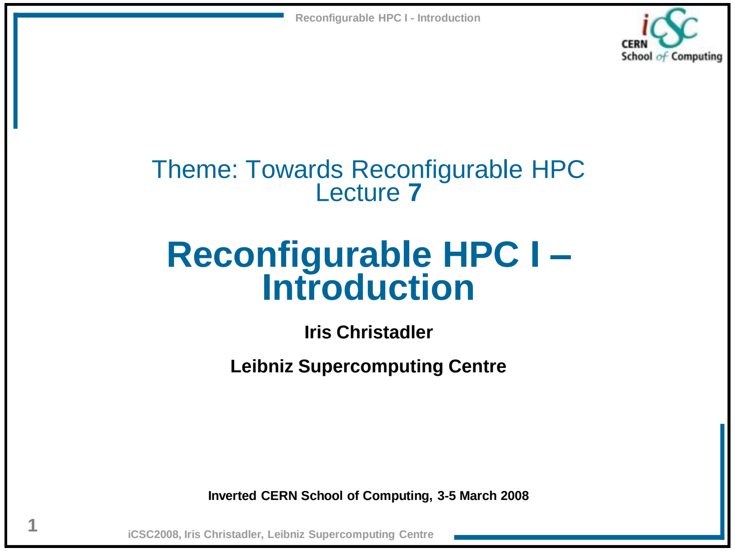
Further Reading

- Maxfield, C., **The Design Warrior's Guide to FPGAs.** April 2004. Newnes.
- Xilinx: www.xilinx.com
- Altera: www.altera.com
- Dong, C., Chen, D., Haruehanroengra, S., Wang, W., **3-D nFPGA**: **A reconfigurable architecture for 3-D CMOS/nanomaterial hybrid digital circuits**. 2007. IEEE Transactions on Circuits and Systems I: Regular Papers, 54 (11), 2489-2501. http://dx.doi.org/10.1109/TCSI.2007.907844

LECTURE 7

Reconfigurable HPC I - Introduction

Tuesday 4 March 2008			
10:05 11:00	Lecture 7	This lecture introduces Reconfigurable High-Performance Computing. The reason behind the limited numbers of Reconfigurable Computing (RC) systems in HPC is discussed. It shows the fields where RC is mostly used and the lessons that can be learned. Important topics: • Introduction to RHPC • Hybrid supercomputers • Areas in which RC is already used • Areas in which RC might be beneficial Participants should have attended Lecture 4, Platforms II - Special Purpose Accelerators, and Lecture 6, Platforms III Programmable Logic, or have equivalent knowledge. **Audience - Pre-requisite** Participants should have attended Lecture 4, Platforms II - Special Purpose Accelerators, and Lecture 6, Platforms III Programmable Logic, or have equivalent knowledge.	**Iris Christadler**



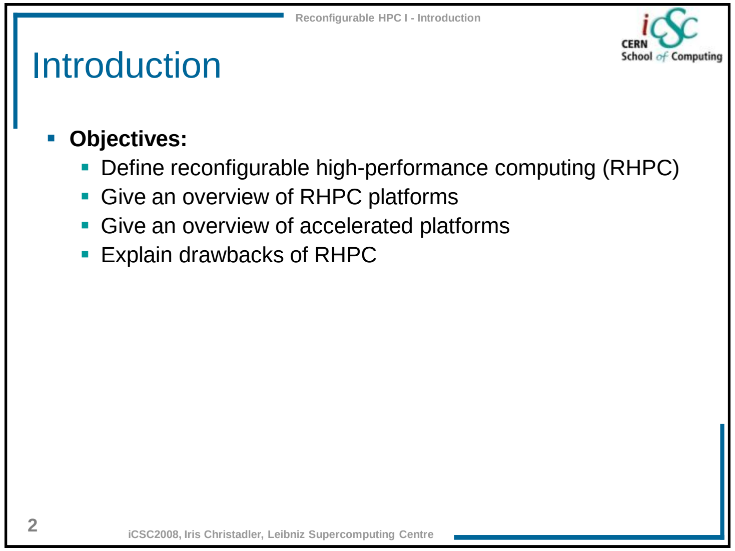
Theme: Towards Reconfigurable HPC
Lecture **7**

Reconfigurable HPC I – Introduction

Iris Christadler

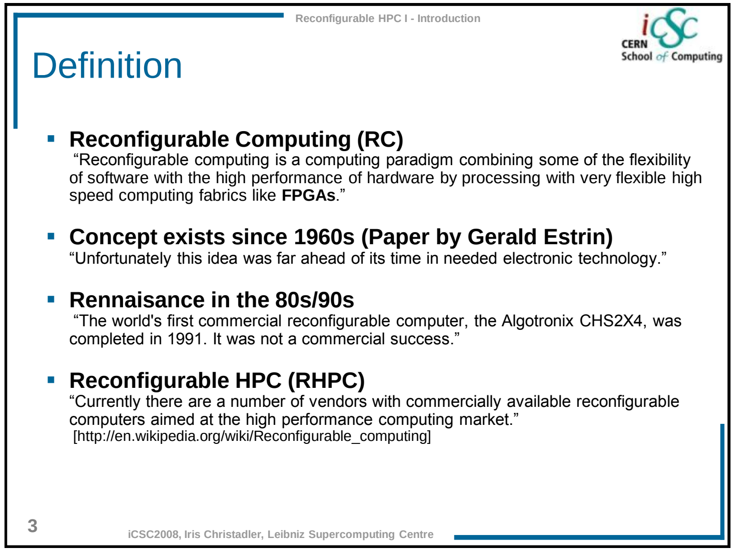
Leibniz Supercomputing Centre

Inverted CERN School of Computing, 3-5 March 2008



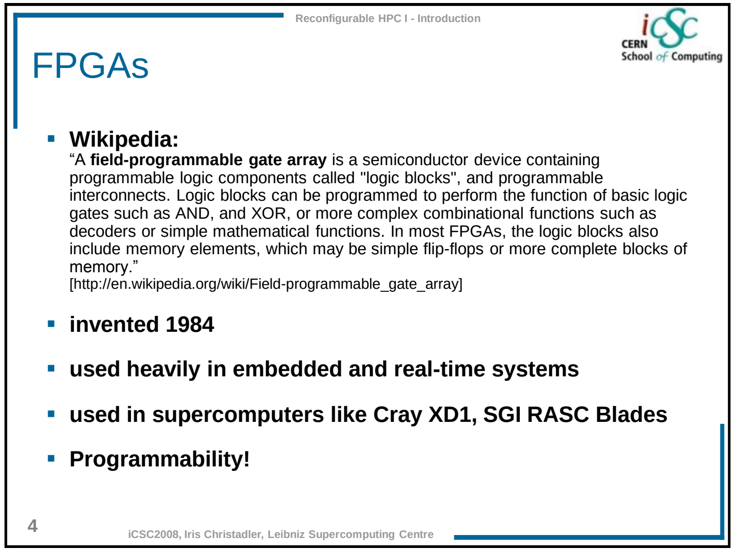
Introduction

- **Objectives:**
  - Define reconfigurable high-performance computing (RHPC)
  - Give an overview of RHPC platforms
  - Give an overview of accelerated platforms
  - Explain drawbacks of RHPC



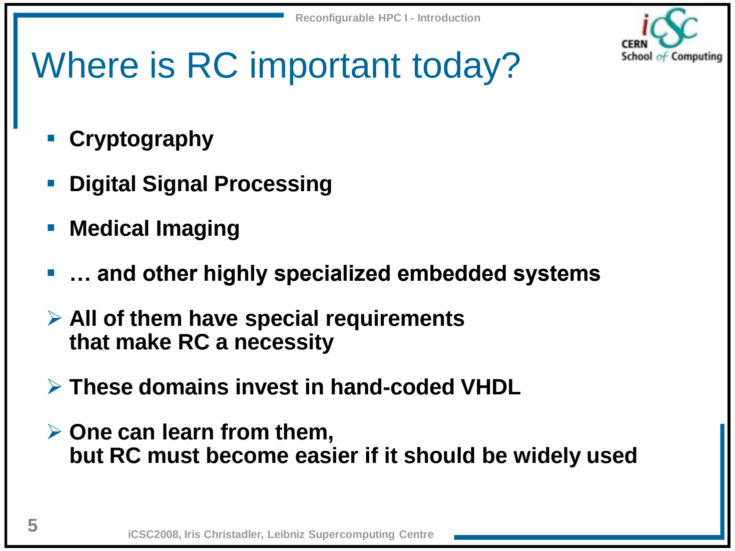
Definition

- **Reconfigurable Computing (RC)**
  "Reconfigurable computing is a computing paradigm combining some of the flexibility of software with the high performance of hardware by processing with very flexible high speed computing fabrics like **FPGAs**."
- **Concept exists since 1960s (Paper by Gerald Estrin)**
  "Unfortunately this idea was far ahead of its time in needed electronic technology."
- **Rennaisance in the 80s/90s**
  "The world's first commercial reconfigurable computer, the Algotronix CHS2X4, was completed in 1991. It was not a commercial success."
- **Reconfigurable HPC (RHPC)**
  "Currently there are a number of vendors with commercially available reconfigurable computers aimed at the high performance computing market."
  [http://en.wikipedia.org/wiki/Reconfigurable_computing]



FPGAs

- **Wikipedia:**
  "A **field-programmable gate array** is a semiconductor device containing programmable logic components called "logic blocks", and programmable interconnects. Logic blocks can be programmed to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or simple mathematical functions. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory."
  [http://en.wikipedia.org/wiki/Field-programmable_gate_array]
- **invented 1984**
- **used heavily in embedded and real-time systems**
- **used in supercomputers like Cray XD1, SGI RASC Blades**
- **Programmability!**



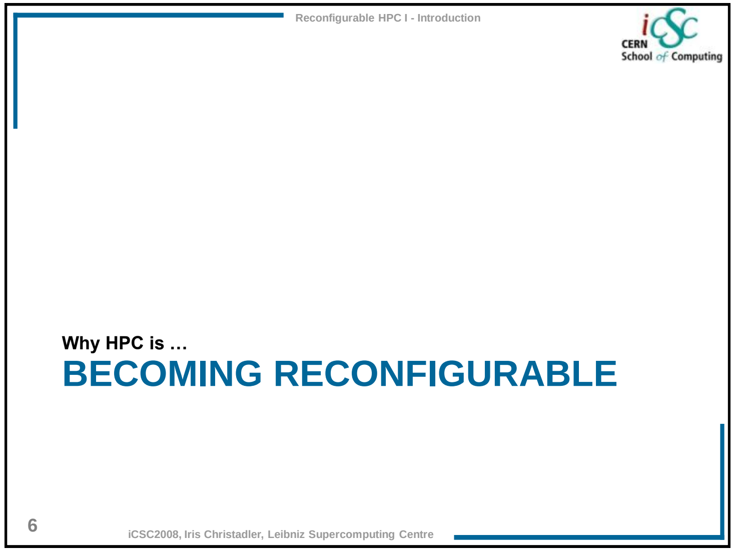
Where is RC important today?

- **Cryptography**
- **Digital Signal Processing**
- **Medical Imaging**
- **… and other highly specialized embedded systems**

➢ **All of them have special requirements that make RC a necessity**

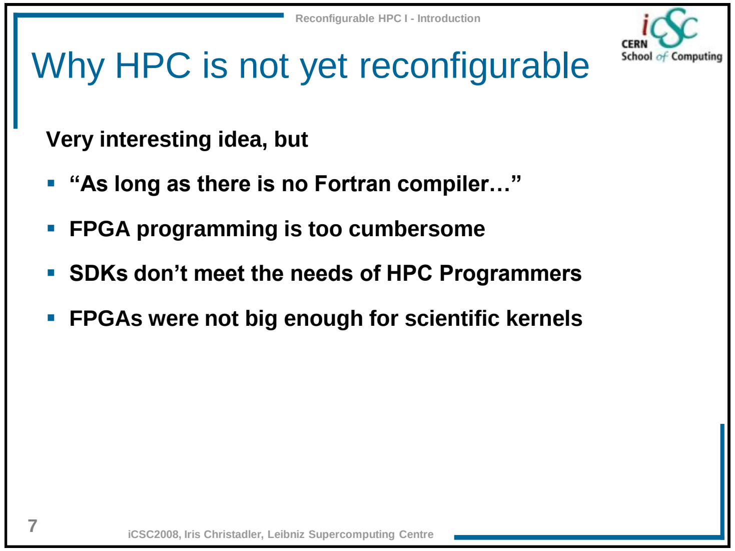
➢ **These domains invest in hand-coded VHDL**

➢ **One can learn from them,**
but RC must become easier if it should be widely used



Why HPC is …

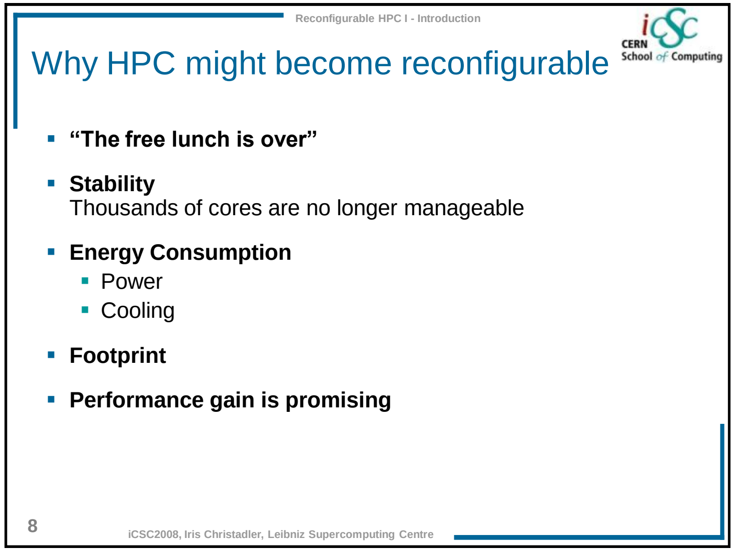
BECOMING RECONFIGURABLE



Why HPC is not yet reconfigurable

Very interesting idea, but

- **“As long as there is no Fortran compiler…”**
- **FPGA programming is too cumbersome**
- **SDKs don’t meet the needs of HPC Programmers**
- **FPGAs were not big enough for scientific kernels**



Why HPC might become reconfigurable

- **“The free lunch is over”**
- **Stability**
  Thousands of cores are no longer manageable
- **Energy Consumption**
  - Power
  - Cooling
- **Footprint**
- **Performance gain is promising**

THE GOOD, THE BAD AND THE UGLY
(NOT NECESSARILY IN THIS ORDER ☺)

Reconfigurable High Performance Computing

EXAMPLES OF RHPC SYSTEMS

Cray XD1

- **Per Chassis:**
  - 12 AMD Opterons (single or dual-core)
  - 6 Xilinix Virtex II Pro (Virtex-4) FPGAs
  - 3.2 GB/s interconnect
- **Released Oct. 2004**
- **RapidArray Interconnect**
- **Development kit:**
  - Specialized libraries
  - Xilinx Tools (HDL)
  - Mitrion-C
- **"Affordable power"** (advertisement claim)

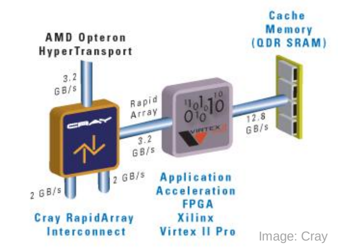

AMD Opteron HyperTransport
3.2 GB/s
Rapid Array
3.2 GB/s
12.8 GB/s
Cache Memory (QDR SRAM)
2 GB/s
2 GB/s
Cray RapidArray Interconnect
Application Acceleration FPGA Xilinx Virtex II Pro
CRAY
VIRTEX

Image: Cray

Cray XD1 Installations

- **FZJ, Juelich, Germany**
  - Not as much development as planned
  - Mainly use of codes from other institutes instead
  - "FPGA programming needs electrical engineers"
  - Pricing models of many FPGA development tools are not suitable for research institutes
- **The George Washington University, USA**
  - RC-Tutorial at Supercomputing Conference
  - Are very happy with their machine(s)
  - Part of CHREC
- **Many more sites:**
  CINECA (Italy), MHPCC (Maui), ASA (Alabama), Zuse Institute Berlin (Germany), …

SGI RASC Blades

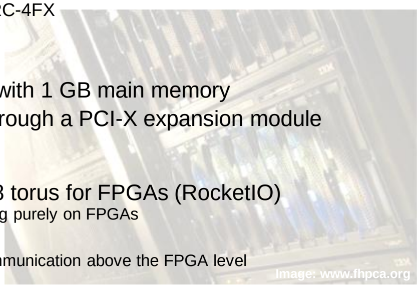
- **Reconfigurable Application-Specific Computing (RASC)**
  - Allows hybrid programming
  - Numalink Interconnect
- **RC100 Blade:**
  - Accelerates SGI Altix Itanium series
  - Dual Xilinx Virtex 4 LX200 FPGA
- **RC200 Blade:**
  - Available for SGI Altix Xeon series
  - Multiple Altera Stratix III FPGAs
- **Development kit:**
  - RASCLib, RASC API
  - Mitrion-C, Handel-C, Xilinx Synthesis Technology

Image: SGI

SGI RASC Installations

- **SGI built the world's largest FPGA supercomputer:**
  (Nov. 2007)
  They ran a broadly used bioinformatics application more than 900 times faster than the same application would run on a traditional cluster. They used the Mitrion Accelerated BLAST-n "off-the-shelf" code.
  SGI's reconfigurable supercomputer featured 70 FPGAs, more than any single system built to date. SGI's FPGA supercomputer accelerated the performance of a complex BLAST-n query by more than 900 times, completing in less than 33 minutes what took a 68-node Opteron-based cluster approximately three weeks to finish.
  [http://www.sgi.com/company_info/newsroom/press_releases/2007/november/fpga.html]
- **Many more:**
  The George Washington University, ZIH Dresden (Germany), …

Maxwell



- **FPGA supercomputer**
  - 32 IBM Intel Xeon Blades
  - 64 Xilinx Virtex-4 FPGAs mounted in two card types
    - Nallatech H101
    - Alpha Data ADM-XRC-4FX
- **Each Blade**
  - 2.8 GHz Intel Xeon with 1 GB main memory
  - hosts two FPGAs through a PCI-X expansion module
- **Two networks**
  - two-dimensional 8x8 torus for FPGAs (RocketIO)
    allows parallel programming purely on FPGAs
  - GigE for the Xeons
    supports inter-process communication above the FPGA level

Image: www.fhpca.org

Maxwell: Porting 3 Demo Codes

- **Financial Engineering**
  - Monte Carlo simulation of stock option pricing
  - Classic Black-Scholes model
  - runs 320 times faster
- **Medical Imaging**
  - 3 and 4D facial image reconstruction codes
  - batch process video images on FPGAs
  - runs 2.5 times faster (sustained)
- **Oil & Gas**
  - 3D controlled source electromagnetic (CSEM) code
  - pretty typical physic simulation code
  - runs 5.5 times faster (but scales only up to 8 nodes)

Maxwell: Findings

- **FPGAs can be used as main processors**
- **Porting will yield good results for HPC Applications with a compact & well-defined kernel**
- **Complexity & compilation overhead makes development slow**
- **"Cost of an effective port is still too high"**
- **Better tools are necessary**

Breaking Petaflops…

EXAMPLES OF ACCELERATED SYSTEMS

TSUBAME (ClearSpeed)

- **Tokyo Institute of Technology, Japan**
  fastest computer in Japan
- **single accelerated system in the top500**
  currently ranked 16, highest rank was 9 (11/2006)
- **TSUBAME Grid Cluster Sun Fire x4600 Cluster**
  - 11664 Opteron cores (2.4/2.6 GHz)
  - 360 ClearSpeed Advance Accelerator Boards
    - 96 GF theoretical peak per board
    - 50 GF sustained double-prec. DGEMM
  - Peak Performance: 102 TF
  - Linpack Performance: 56 TF

ClearSpeed: Technical Details

- **Offer two versions:**
  - Advance X620 (PCI version)
  - Advance e620 (PCIe version)
  - Two CSX600 chips on a board
- **CSX600 chip:**
  - IEEE 754 standard conform
  - Clocked at 250 MHz
  - 1GB DDR2 SDRAM (500 MHz)
  - 96 compute engines:
    - Each has a **64 bit floating point** adder and multiplier
    - Each has 6KB of high-speed local storage

ClearSpeed

- **Advertisement claims:**
  - "World's highest performance processor"
    (96 GF per board)
  - "World's highest performance per watt"
    (<25 W/Board, 2 GF/Watt)
- **Programming environment:**
  - Compiler with extension (Cn)
  - Assembler, debugger, execution profiler
  - Pre-written performance libraries:
    Level 3 BLAS, LAPACK, FFT
  - Ported versions of some codes:
    Amber, Molpro, …

RIKEN MD-GRAPE 3

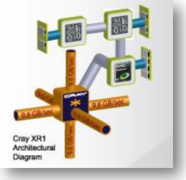
- **RIKEN Yokohama Institute, Japan**
- **Special-purpose system**
  for molecular dynamics simulation
  (protein structure prediction)
- **Consists of:**
  - 201 unit of
    24 custom MDGrape-3 chips
    (4808 total)
  - plus several Xeon servers as hosts
- **Performance:**
  - 1 Petaflop (June 2006)
  - but: not capable of running Linpack

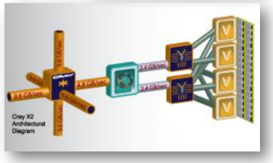
IBMs Roadrunner (Cell)

- **First petaflop system worldwide**
  Peak Performance will be 1.33 PF
- **A hybrid Opteron-Cell system**
  Cells used as accelerates
- **Cluster of:**
  - 6,912 AMD dual-core Opterons (1.8 GHz)
  - 12,960 IBM Cell eDP accelerators (~ 100 GF peak each)
- **Power Consumption:**
  - 3.9 MW Power
  - 0.35 GF/Watt (peak)
- **due in Q3/2008**

Cray XT5$_h$

- **"A milestone on the path to Adaptive Supercomputing"**
- **An integrated hybrid supercomputer**
  - XT5 Blade: scalar processing (Opterons)
    - Dual- or quad-core Opterons
  - X2 Blade: vector processing
    - more than 100 GF peak performance
    - support Unified Parallel C **(UPC)**
    - support Co-Array Fortran **(CAF)**
  - XR1 Blade: FPGA acceleration
    - 1 Opteron
    - 2 DRC Computer's RPUs
    - HyperTransport





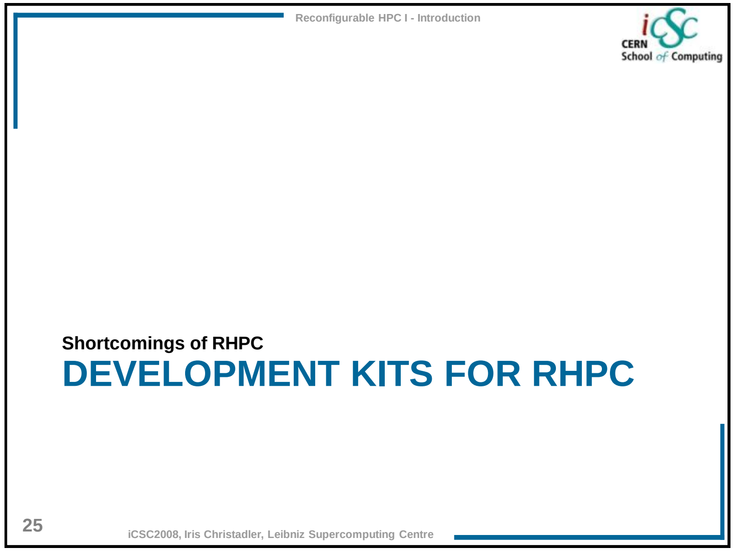
Images: Cray

Cray XT5$_h$ Installation: HECToR

- **High-End Computing Terascale Resources (HECToR)**
- **Edinburgh Parallel Computing Centre (EPCC)**
- **XT4 system**
  is already installed
- **X2 vector "Black Widow" system**
  will be delivered in September 2008
- **Upgrade to XT5$_h$ in 2009**

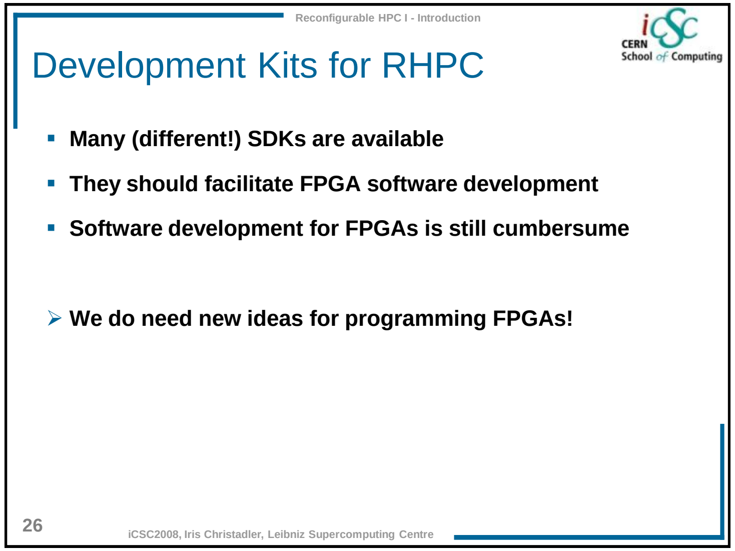
Hint I: Former EPCC computers have been available through DEISA and the DEISA Extreme Computing Initiative

Hint II: EPCC is also one of the HPC-Europa sites



Shortcomings of RHPC

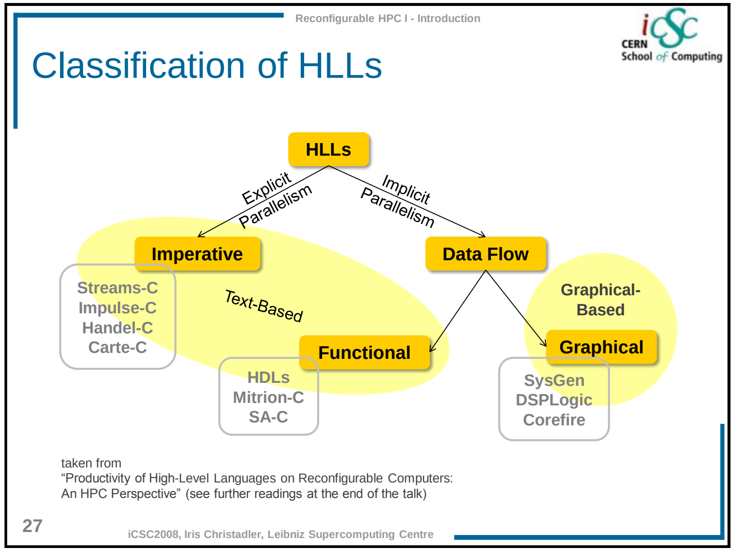
DEVELOPMENT KITS FOR RHPC



Development Kits for RHPC

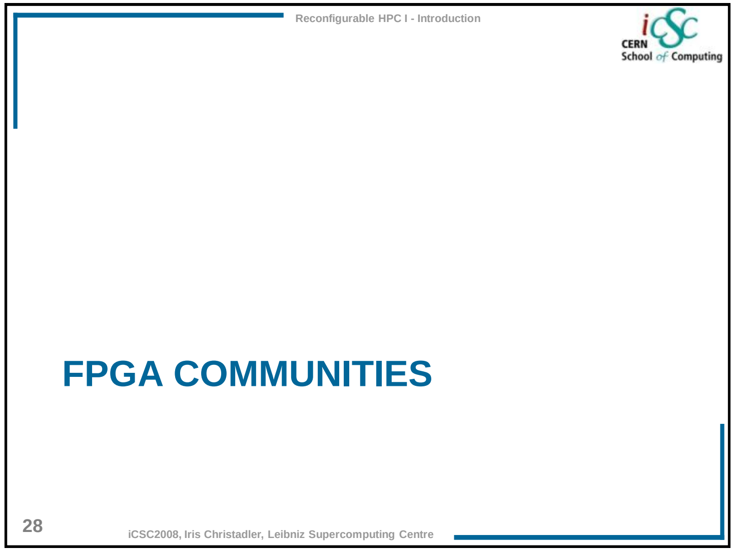
- **Many (different!) SDKs are available**
- **They should facilitate FPGA software development**
- **Software development for FPGAs is still cumbersume**

➢ **We do need new ideas for programming FPGAs!**

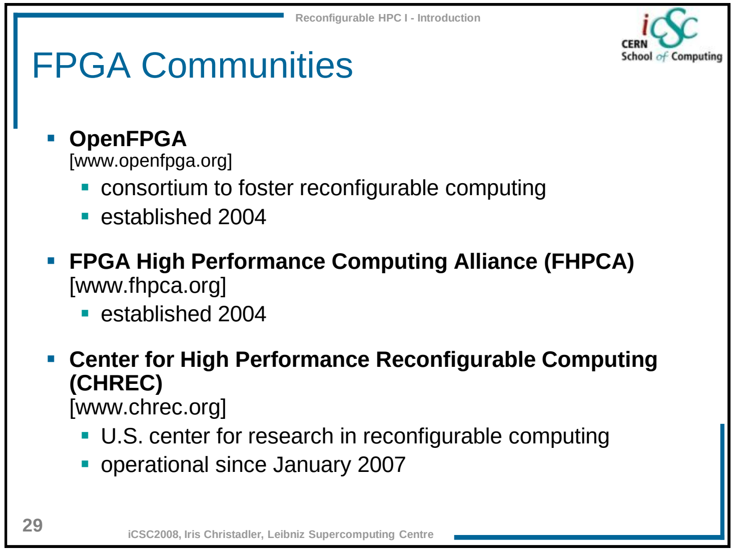


Classification of HLLs

taken from
“Productivity of High-Level Languages on Reconfigurable Computers:
An HPC Perspective” (see further readings at the end of the talk)

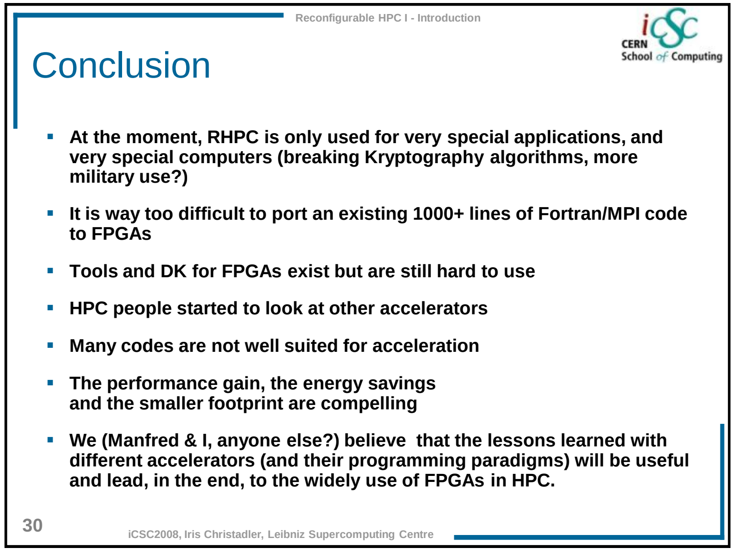


FPGA COMMUNITIES



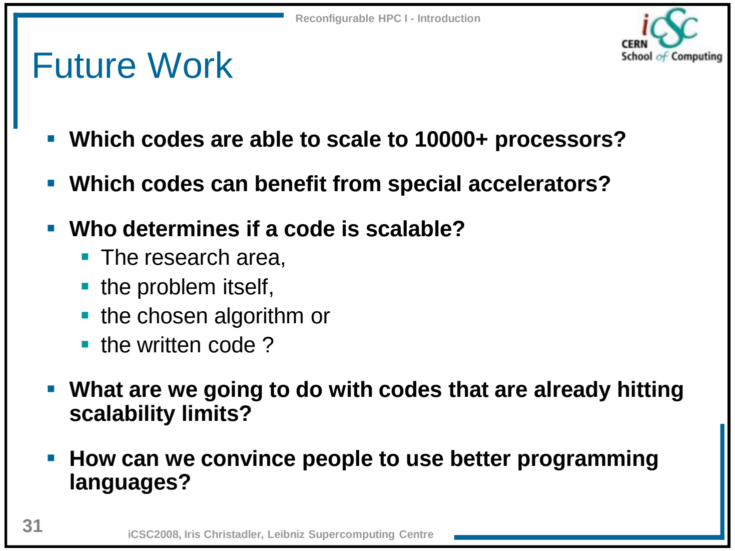
FPGA Communities

- **OpenFPGA**
  [www.openfpga.org]
  - consortium to foster reconfigurable computing
  - established 2004
- **FPGA High Performance Computing Alliance (FHPCA)**
  [www.fhpca.org]
  - established 2004
- **Center for High Performance Reconfigurable Computing (CHREC)**
  [www.chrec.org]
  - U.S. center for research in reconfigurable computing
  - operational since January 2007



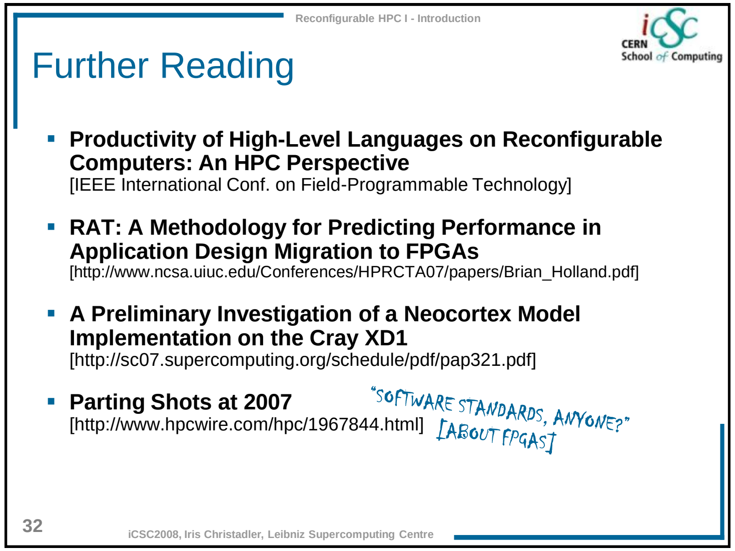
Conclusion

- **At the moment, RHPC is only used for very special applications, and very special computers (breaking Kryptography algorithms, more military use?)**
- **It is way too difficult to port an existing 1000+ lines of Fortran/MPI code to FPGAs**
- **Tools and DK for FPGAs exist but are still hard to use**
- **HPC people started to look at other accelerators**
- **Many codes are not well suited for acceleration**
- **The performance gain, the energy savings and the smaller footprint are compelling**
- **We (Manfred & I, anyone else?) believe that the lessons learned with different accelerators (and their programming paradigms) will be useful and lead, in the end, to the widely use of FPGAs in HPC.**



Future Work

- **Which codes are able to scale to 10000+ processors?**
- **Which codes can benefit from special accelerators?**
- **Who determines if a code is scalable?**
  - The research area,
  - the problem itself,
  - the chosen algorithm or
  - the written code ?
- **What are we going to do with codes that are already hitting scalability limits?**
- **How can we convince people to use better programming languages?**



Further Reading

- **Productivity of High-Level Languages on Reconfigurable Computers: An HPC Perspective**
  [IEEE International Conf. on Field-Programmable Technology]
- **RAT: A Methodology for Predicting Performance in Application Design Migration to FPGAs**
  [http://www.ncsa.uiuc.edu/Conferences/HPRCTA07/papers/Brian_Holland.pdf]
- **A Preliminary Investigation of a Neocortex Model Implementation on the Cray XD1**
  [http://sc07.supercomputing.org/schedule/pdf/pap321.pdf]
- **Parting Shots at 2007**
  [http://www.hpcwire.com/hpc/1967844.html] "SOFTWARE STANDARDS, ANYONE?" [ABOUT FPGAS]

CERN School of Computing

LECTURE 8

Reconfigurable HPC II - HW Design Methodology, Theory & Tools

Tuesday 4 March 2008

11:30
12:25

Lecture 8

Manfred Muecke

This lecture will first focus on existing tools for making use of FPGAs as number crunchers and will give examples of existing solutions. It will then discuss limitations and how they could be overcome.

Important topics:

- Old attempts and current tools
- About levels of HW abstraction
- What is higher-level synthesis?
- Examples from HPC and HEP

Audience
This lecture is intended for students seeking deeper understanding of hardware design description issues in general and when using FPGAs as number crunchers – understanding of programming languages and basic compiler technology will be helpful.

Pre-requisite
This lecture builds upon the preceding lectures "6. Platforms III - Programmable Logic" and "7. Reconfigurable HPC I - Introduction".
While not necessarily being a required prerequisite for lecture 9 and 10, it motivates why going beyond existing tools is important.

Theme: Towards Reconfigurable HPC
Lecture **8**

Reconfigurable HPC II - HW Design Methodology, Theory & Tools

Manfred Mücke

University of Vienna
Research Lab Computational Technologies and Applications

Inverted CERN School of Computing, 3-5 March 2008

Outline

- **Objectives**
  - Understanding Compilation targets
  - Understanding CPU vs. FPGA trade-offs
  - Understanding basic FPGA design issues
  - Understanding categories of synthesis tools
- **Contents**
  - Compilation targets
  - Hardware synthesis
  - Some available languages and tools

Choose Your Target

- Preferred target for HPC applications: **many tightly coupled cores**

⇨ Applications (x+MPI) are written for distributed machines

Enter FPGAs:

- In EE, FPGAs combine diverse functionality
- (several) EEs design for a single FPGA
- Some tools consider CPU+FPGA (HW/SW Codesign)

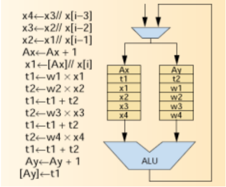
⇨ We consider **single-FPGA tools only!!**

A not so simple question

What fits into an FPGA?

Temporal Decomposition

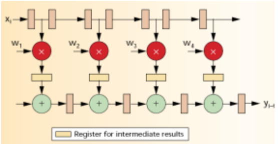
- CPUs provide a defined set of functions (their instruction set)
- A program on a classic CPU uses one function per time step.
- It solves a problem by sequential application of given functionality
  ⇨Temporal decomposition
- Memory limits complexity
- ISA defines efficiency



```
x4←x3// x[i−3]
x3←x2// x[i−2]
x2←x1// x[i−1]
Ax←Ax + 1
 x1←[Ax]// x[i]
 t1←w1 × x1
 t2←w2 × x2
 t1←t1 + t2
 t2←w3 × x3
 t1←t1 + t2
 t2←w4 × x4
 t1←t1 + t2
 Ay←Ay + 1
[Ay]←t1
```

Spatial Decomposition

- FPGAs implement a fully parallel function set, custom-tailored for your application.
- All functionality operates *in parallel*.
- All required functionality needs to fit in the given area (save run-time configuration)
  ⇨ Spatial Decomposition
- Complexity is limited by silicon area
- Efficiency is limited by FPGAs basic building blocks (and their use)



Now, HOW MUCH?

- You can fit arbitrary complex functionality in an FPGA by implementing a CPU (softcore) and running code on it. (slowest Nios II requires ~3% of an EP2C35)
- Functionality in LEs requires area, but brings speed.
- Only good tools achieve good implementations, and best balance!
- Consider data and control flow separately

Algorithm	Speed Increase (vs. Nios II CPU)	System f_{MAX} (MHz)	System Resource Increase *(1)*
Autocorrelation	41.0x	115	124%
Bit Allocation	42.3x	110	152%
Convolution Encoder	13.3x	95	133%
Fast Fourier Transform (FFT)	15.0x	85	208%
High Pass Filter	42.9x	110	181%
Matrix Rotate	73.6x	95	106%
RGB to CMYK	41.5x	120	84%
RGB to YIQ	39.9x	110	158%

Again, Your Target (On-Chip)?

Two basic approaches:

- **Start with all-SW**
  design small accelerators for specific functionality
  (extract data flow)
  repeat till you run out of area or reach speed
- **Start with all-HW**
  extract more complex FSMs
  generalize functional units
  schedule operations
  repeat till your design fits

HPC starts here

A complex issue

Hardware Synthesis

Logic Synthesis

- Digital Systems ⇨ Logic Synthesis
- **Logic synthesis = Inferring an implementation from a more abstract description**
- Already a*b is far from straightforward:

FPGA	CPU
How many bits?	Fixed data paths
Power budget?	Fixed
How fast?	Given clock
How much area?	Fixed

redesign often, make perfect match	Design once, use many years

Short History of Synthesis

- **Logic minimization**
  Same abstraction level ⇨ minimize resources
- **Register-Transfer Level (RTL) Synthesis**
  Registers (timing) + combinational logic
  Freedom between registers
- **Behavioural synthesis**
  Untimed description
  Freedom everywhere
- **High(er)-Level Synthesis**
  more decent term
- **Algorithmic Synthesis**
  "Care more about specifying your problem, than the hardware"

Algorithmic Synthesis - Constraints

How to start:

1. Pick your input language
2. Model your application
3. Take a compiler/synthesizer
4. Give model and tell the compiler,
   which out of $10^{100000..}$ versions you want!

⇨ Constraints inlined or separate
high- or low-level
quantitative or qualitative

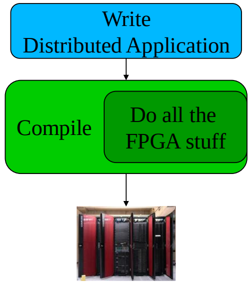
Better modelling language ⇨ less constraints needed

It's a long way...

FPGA Design Flow

Design Flow for HPC Users

- Do you want to know the details about FPGA design flows?
  ⇨ NO
- ... but you need to know for the following slides ☺



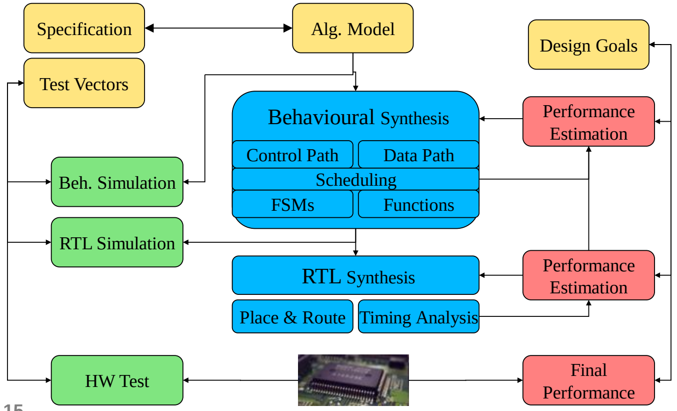
Write Distributed Application

Compile

Do all the FPGA stuff

unfortunately, it is
a bit more difficult...

Algorithmic HW Design Flow



Specification

Alg. Model

Design Goals

Test Vectors

Behavioural Synthesis

Control Path

Data Path

Scheduling

FSMs

Functions

Performance Estimation

Beh. Simulation

RTL Simulation

RTL Synthesis

Place & Route

Timing Analysis

Performance Estimation

HW Test

Final Performance

A Design-Flow Wishlist

- **Modeling**
  Expressing your problem in a machine-readable way
  Can you read it, too?
- **Verification**
  Is this really, what you wanted?
- **Simulation**
  Testvectors
  SW development
  Sufficiently fast?
- **Synthesis**
  Hardware implementation
  Achieving performance goals?

Executable Software Specification goes Hardware

Algorithmic Synthesis Tools

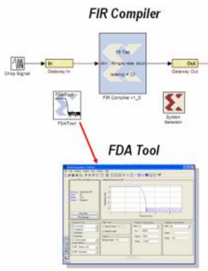
(Selection of) Existing Design Tools

- **Block-based tools/generators**
  - Vendor IP Blocks (LPM), Simulink
- **Annotated/reduced/extended C**
  - Handel C, Impulse-C, Mitrion-C
- **C-based Design Explorer**
  - Catapult C, C2H, PICO Express
- **Modeling Languages**
  - SystemC, Esterel
- **Binary Synthesis**

Block-Based

- **Xilinx System Generator**
- Blockset for Simulink
  - DSP (FIR filters, FFTs, ..)
  - Error correction (Viterbi, Reed-Solomon, ..)
  - Memories (FIFO, RAM, ROM,..)
  - Arithmetic and digital logic
- Automatic generation of VHDL or Verilog from Simulink
- Hardware co-simulation (“FPGA-in-the-loc

⇨ Highly efficient *if your functionality is available*

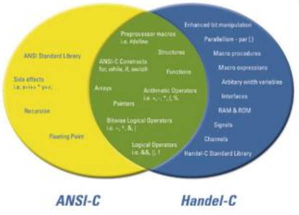


Annotated C

Handel-C by Celoxica

Celoxica

- C-based HDL
  Handel-C = ANSI-C
  – side-effects, floating-point ,
  recursion
  + signals, channels, parallelism, bit
  RAM, HW library
- Originally developed at Oxford University Computing Laboratory (~1996)



C-Based

Impulse C by Impulse Accelerated Technologies

- Originally developed as Streams-C at LANL
- C subset + library extension
- Partition your application in processes.
  Connect tasks with streams.
  (Interface logic inferred by compiler)
- Tasks can be executed either on CPU or FPGA.
- designed for dataflow-oriented, streaming applications

C-Based

Mitrion-C by Mitrionics

- "Implicitly parallel C-family programming language"
- Infers an optimized CPU for your application.
  This "Mitrion Virtual Processor" is described in VHDL,
  to be implemented on FPGA.
- Extracts heavily used functionality from your code and moves it into custom CPU instructions
- "Acceleration typically 10x to 30x
  (compared to execution on a sequential CPU)

C-Based Design Explorer

C2H Compiler by Altera

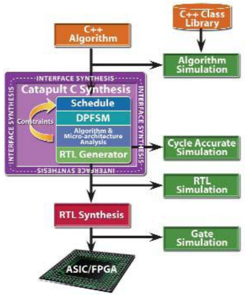
1. Profile your (ANSI-C) software
2. Highlight the desired functions within the Nios II IDE and "right-click to accelerate".
3. Review the compiler report file to determine simple C code optimizations.
4. Optimize and iterate until desired performance is met.

- Highly integrated!
  Compiler guides user.

C-Based Design Explorer

Catapult C Synthesis by MentorGraphics

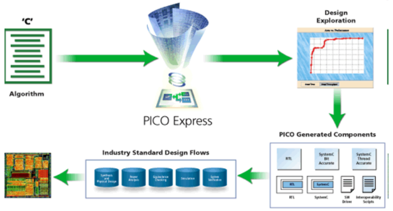
- Accepts C++ as input
  Creates a range of design variants
  User picks by specifying constraints
- SystemC simulation models
- Excels in graphical feedback
- "The price [..] currently ranges from $89,000 to $275,000"



C-Based Design Explorer

PICO Express by Synfora

- Hardware synthesis from C,
  "creating flow-controlled networks of hardware accelerators"
- Design exploration



Matlab-based Design Explorer

AccelDSP by Xilinx

- "If the Xilinx blockset is not sufficient"
- Synthesizeable blocks from Matlab code
  Architectural exploration of high-level DSP algorithms
  Floating- to fixed-point conversion
  C++ simulation model generation
- Originally developed by AccelChip, acquired in 2006

Modelling Language

SystemC

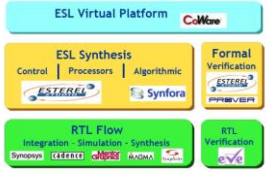
- = C++ libraries and macros enabling system-level modeling and simulation of digital systems (IEEE Std. 1666™-2005).
- = description language + simulation kernel
- Description at different levels of abstraction (RTL, C, TLM)
- Very fast simulation at higher abstraction levels
- SoC architectural exploration (busses, memory, responsiveness)
- TLM (protocol modeling) is very succesful
- Hardware synthesis from SystemC is in its infancy

Esterel-based Synthesis

Esterel Studio

- "The leading design and verification suite
  for control-intensive hardware IP"
- Esterel = synchronous programming language
  developed at Ecole des Mines & INRIA (1980s).
  Well-suited for describing reactive systems.
  + captures concurrency
  + formal verification possible
  - data-driven designs
- VHDL/Verilog/SystemC from
  Esterel specification

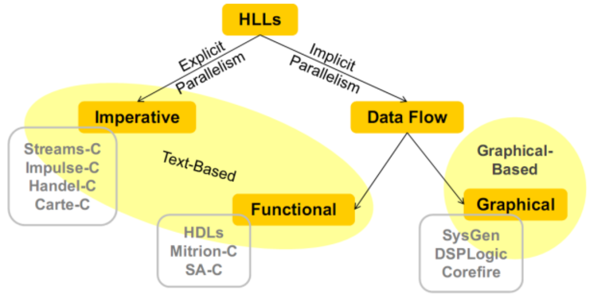
⇨ Semantically sound foundation
(Lustre is used for Airbus FCS**)**



Binary Synthesis

- Synthesis from executing binaries
- Replaces CPU by "CPU-VM"
  Extracts, implements and calls HW accelerators on the fly
  Compare to Java JIT Compiler approaches
  + Connects to all existing tools
  + Transparent to the user
  - Limited scope
  - Reverse Compiler Optimizations
- Requires closely coupled hybrid system

A Classification



Taken from: " Productivity of high-level languages on reconfigurable computers: An HPC perspective"

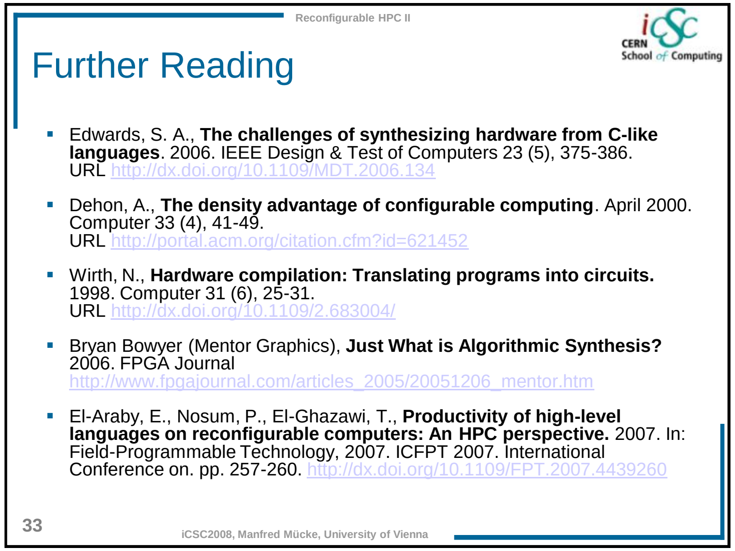
Is this the end of the story?

Outlook

Open Questions

- Is C a good choice for hardware design entry?
  Technically: NO
  Economically: Maybe
  Legacy Code: Certainly

  Alternatives ⇨ Next Lecture
- Will FPGAs stay around?
  Generally: For sure
  In HPC: we believe yes ⇨ Hybrid Systems ⇨ Last lecture
- Is FPGA-centric design helpful for HPC? ⇨NO
  Unified design approach?
  Special-purpose accelerator + libraries?



Further Reading

- Edwards, S. A., **The challenges of synthesizing hardware from C-like languages**. 2006. IEEE Design & Test of Computers 23 (5), 375-386. URL http://dx.doi.org/10.1109/MDT.2006.134
- Dehon, A., **The density advantage of configurable computing**. April 2000. Computer 33 (4), 41-49. URL http://portal.acm.org/citation.cfm?id=621452
- Wirth, N., **Hardware compilation: Translating programs into circuits.** 1998. Computer 31 (6), 25-31. URL http://dx.doi.org/10.1109/2.683004/
- Bryan Bowyer (Mentor Graphics), **Just What is Algorithmic Synthesis?** 2006. FPGA Journal http://www.fpgajournal.com/articles_2005/20051206_mentor.htm
- El-Araby, E., Nosum, P., El-Ghazawi, T., **Productivity of high-level languages on reconfigurable computers: An HPC perspective.** 2007. In: Field-Programmable Technology, 2007. ICFPT 2007. International Conference on. pp. 257-260. http://dx.doi.org/10.1109/FPT.2007.4439260

LECTURE 9

Advanced and Emerging Parallel Programming Paradigms

Tuesday 4 March 2008			
14:00 14:55	Lecture 9	This lecture will present some parallel programming paradigms and will explain why they map so well on reconfigurable hardware. It will then focus on hardware-independent programming and motivate why this is important and how it can be achieved. Current developments will be discussed. Important topics: • Explicit and implicit parallelism • On granularity of parallelism and matching hardware architectures • On cross-compiling of HPC applications (prospects and issues) • Parallel programming languages in the making **Audience** This lecture is more theoretic than preceding lectures and is thought for students seeking a more general understanding on how a programming paradigm affects implementation and performance of languages and tools for reconfigurable HPC. **Pre-requisite** As this lecture is based on issues and conclusions collected from all preceding lectures, having followed as many as possible is certainly helpful. The most helpful prerequisites are possibly • lecture "6. Platforms III - Programmable Logic" and • lecture "8. Reconfigurable HPC II - HW Design Methodology, Theory & Tools"	**Manfred Muecke**

Theme: Towards Reconfigurable HPC
Lecture **9**

Advanced and Emerging Parallel Programming Paradigms

Manfred Mücke

Research Lab Computational Technologies and Applications
University of Vienna

Inverted CERN School of Computing, 3-5 March 2008

Objectives

Objectives

- Stressing importance of parallel problem specification
- Outlining common steps in parallel programming
- Presenting (not teaching) different paradigms
- Presenting some parallel languages

Contents

- Why parallel programming?
- Concurrency and parallelism
- Parallel programming paradigms
- New Languages in the making

Don't we have enough problems?

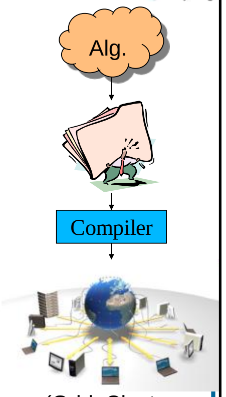
Why Parallel Programming?

Why Parallel Programming?

- Contemporary computing platforms become increasingly parallel at different levels:
  - Grid, BOINC, Clusters
  - Manycores, multicores, GPUs
  - MMX, SSE, AltiVec
  - FPGAs
- Compilers can only exploit parallelism within the limits of the language used.

⇨ It is on the shoulders of programmers to write efficient software (and to rewrite it constantly)

⇨ Parallel languages allow much simpler and more reliable extraction of parallelism than sequential languages.



(Grid, Cluster, Multicore, GPU, FPGA, Hybrid)

Why Parallel Programming

⇨ Because it appears the only solution to exploit the diverse range of computing platforms we keep inventing.

"A major challenge for modern HPC systems is their lack of programmability."

M.Weiland, EPCC Edinburgh [1]

"Parallel programming languages are, I think, the most important issue in computing today"

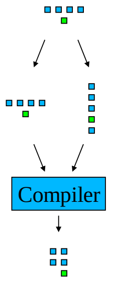
Burton Smith, Microsoft

But it might take a while....

A word on parallelizing compilers

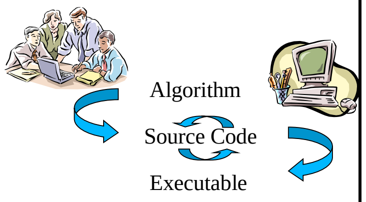
- Most real-world problems feature some concurrency.
- Sequential programming arbitrarily serializes concurrent tasks.
- Parallelizing compilers need to reverse-engineer code to separate real from programmer-induced sequential code.
- This job gets harder, the larger the scope (ILP possible, applications ☹).

⇨ Parallelizing compilers have a hard and increasingly impossible job *when faced with purely sequential code*!



Men vs. Machine

- We tend to optimize existing code.
- Only helpful if original code structure matches hardware.
- Few people nowadays know the platform, their software will run on (say in 5 years).
- Good performance across platforms is only possible, if concurrency (not explicit parallelism) is expressed in source code ... and if we have suitable compilers.



Two friends

Concurrency and Parallelism

Concurrency and Parallelism

Concurrency and parallelism are often used synonymously.

- **Concurrency:** The independence of parts of an algorithm. Our world is inherently concurrent!
- **Parallelism** (also parallel execution): Two or more parts of a program are executed at the same moment in time.

Concurrency is a necessary **prerequisite** for parallel execution

but

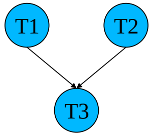
Parallel execution is only one **possible consequence** of concurrency.

Expressing Concurrency

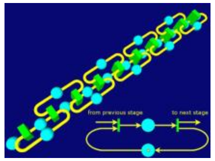
- **Mathematical notation (e.g. no common subterms)**

x = ab + cd + ef

- **Taskgraphs**



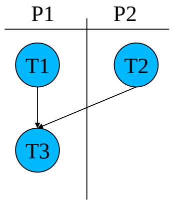
- **Petri Nets**



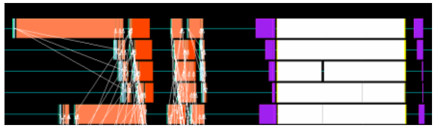
- **...**

Expressing Parallelism

- **Annotated source code**
- **Mapped task graphs**



- **Gantt Charts**



- **...**

The Ideal World

Our world is not sequential!

The more concurrency a program contains,

- the more parallel instructions/threads/tasks can be extracted *reliably* by (suitable) compilers and
- the better the automatic mapping on distributed hardware architectures.

Concurrency ⇨ Algorithm

Source

Parallel Exec.⇨ HW

Our World

- Classical CPUs are sequential.
- There is an enormous **sequential programming knowledge** out there.
- Parallel Programming is requiring **new skills and new tools**.
- **"Saving existing investments":** Everyone is reluctant to make the big move. Small transitional steps are preferred.

Let's face the challenges ahead!

Capitalizing on Concurrency

From Concurrency to Parallelism

From Concurrency to Parallel Execution

- This is not a one-stop show

- Inherent Concurrency
  ↳Decomposition
  ↳Mapping
  ↳ Add Communication
  ↳ Add Synchronization

  ↳ Enjoy Parallel Execution

Explicit vs. Implicit

- **Explicit specifications** allow the programmer to guide the implementation, but pollute the code with (low-level) details.

```
float a, b, c;
x = (a + b) + c
```

- **Implicit specifications** give more freedom to the compiler, but make hand-tuning difficult

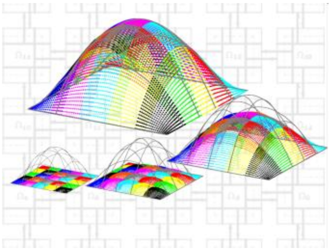
```
smallfloat a, b;
biggerfloat c;
x = a + b + c;
```

- Explicit specifications define well the **level of abstraction** provided.

Decomposition	×✓
Mapping	×✓
Communication	×✓
Synchronization	×✓

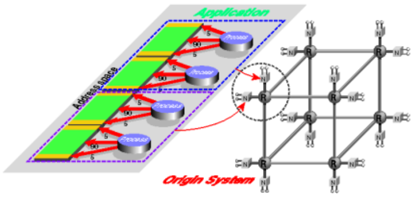
Decomposition

- Divide problem (program/data) into separate tasks/threads/operations to be executed/evaluated in parallel on distinct devices.



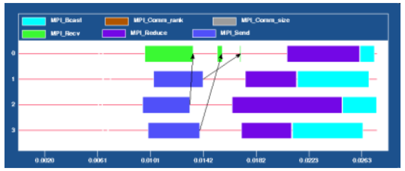
Mapping

- Assign identified tasks to specific hardware.
- Observe
  - Communication patterns
  - Network topology
  - Special functionality (FPU, ..)



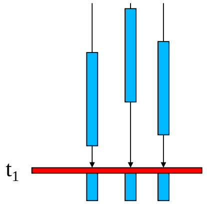
Communication

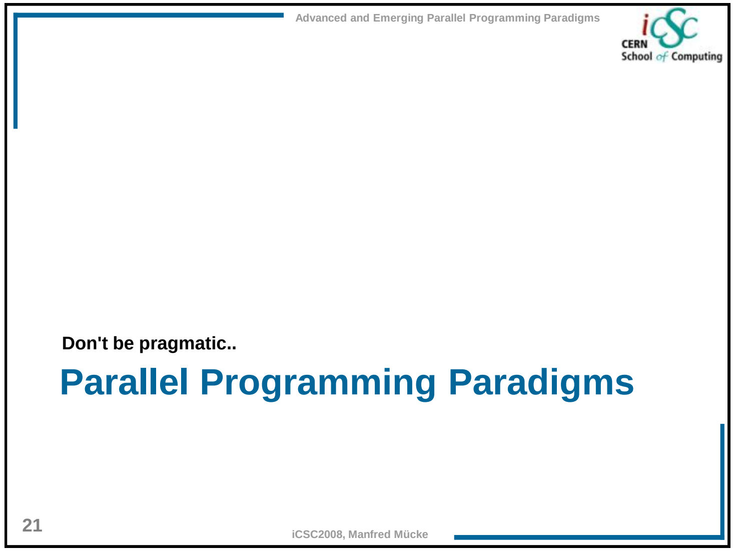
- Implement communication action if nonlocal data is required.
- Hardware-dependent
- ☠ Deadlock ☠
  (waiting for data which never arrives)



Synchronization

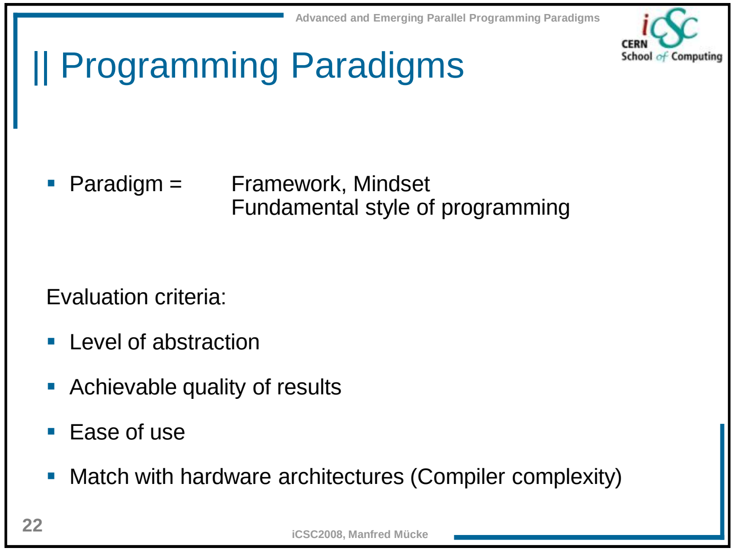
- Make sure each task within a set has reached a certain point.
- Waiting time accumulates fast!
- ☠ Deadlock ☠
  (waiting for absent task)





Don't be pragmatic..

Parallel Programming Paradigms

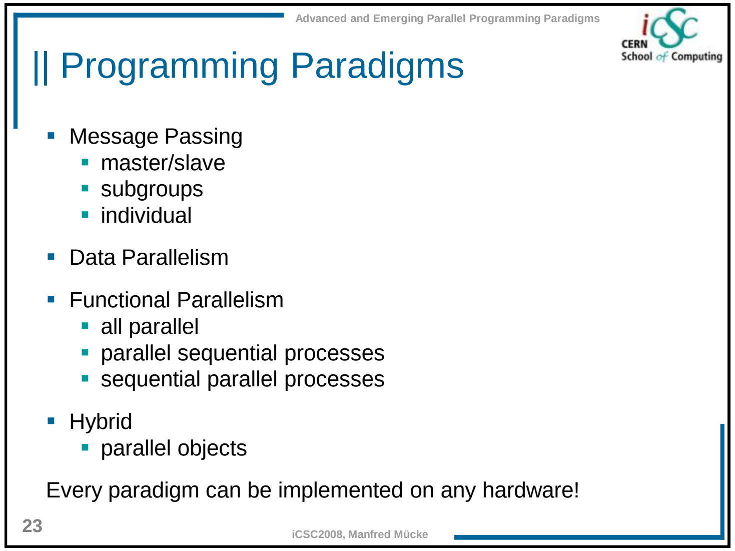


|| Programming Paradigms

- Paradigm = Framework, Mindset
  Fundamental style of programming

Evaluation criteria:

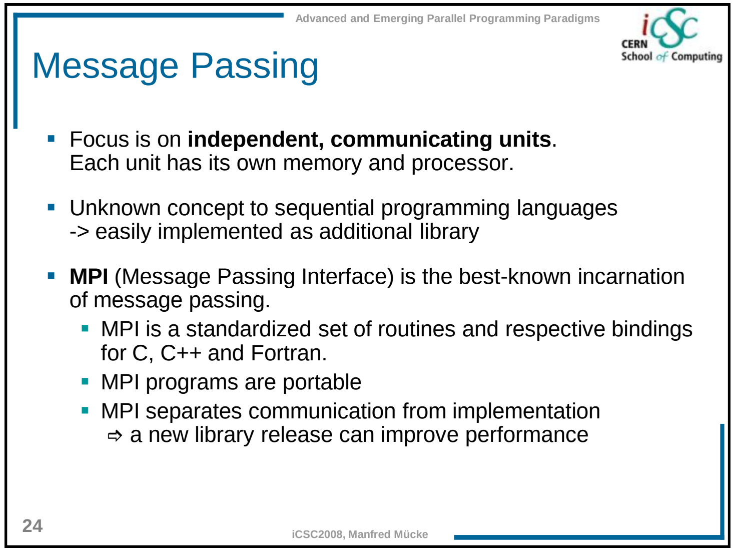
- Level of abstraction
- Achievable quality of results
- Ease of use
- Match with hardware architectures (Compiler complexity)



|| Programming Paradigms

- Message Passing
  - master/slave
  - subgroups
  - individual
- Data Parallelism
- Functional Parallelism
  - all parallel
  - parallel sequential processes
  - sequential parallel processes
- Hybrid
  - parallel objects

Every paradigm can be implemented on any hardware!



Message Passing

- Focus is on **independent, communicating units**.
  Each unit has its own memory and processor.
- Unknown concept to sequential programming languages
  -> easily implemented as additional library
- **MPI** (Message Passing Interface) is the best-known incarnation of message passing.
  - MPI is a standardized set of routines and respective bindings for C, C++ and Fortran.
  - MPI programs are portable
  - MPI separates communication from implementation
    ⇨ a new library release can improve performance

Message Passing Variations

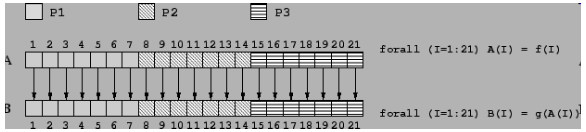
- **Uniform Master/Slave**
  Master distributes data and synchronizes all slaves.
  Direct Implementation: MPI collective communication primitives (Broadcast, Reduce, ...)
  - + Code scales easily (just change n)
  - + Just one code for all machines
  - - No load balancing possible
- **Nonuniform Master/Slave**
  - + Load balancing possible
  - - Explicit communication with each slave

Message Passing Variations

- **Subgroups**
  Several masters can coexist, directing slave subgroups.
  MPI: Communicators
  - + Different tasks can be accomplished in parallel
  - - Coordination of Masters and respective subgroups
- **Individual**
  N coexisting, independent tasks
  - + Arbitrary communication complexity
  - - Only send/receive
  - - Deadlock

Data Parallelism

- The same operation is applied to multiple data sets.
  Data sets implicitly define task distribution
- Extends existing data structures
  Simple implementation as annotations, e.g.
  High-Performance Fortran
  OpenMP
-
  - + Very efficient for matching problems
  - + Easy to debug (communication is implicit)
  - - Load balancing difficult



```
forall (I=1:21) A(I) = f(I)
forall (I=1:21) B(I) = g(A(I))
```

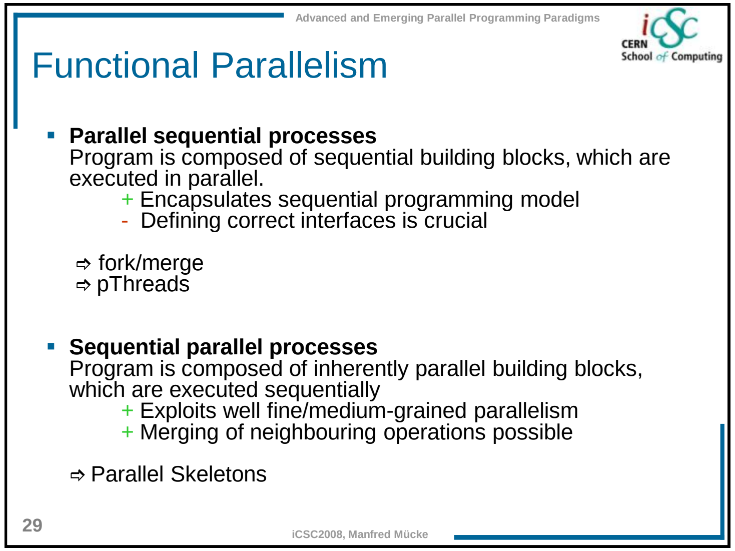
Functional Parallelism

- Parallelism is derived from **program flow**.
  Independent loops/functions/ can be evaluated in parallel.
- How to **identify independent code**?

Imperative languages	--
Single-assignment languages	+
Functional languages	+
Mathematical Notation	++

⇨ Intimate link with matching programmming language

- **All parallel**
  As much parallelism as possible is expressed
  Native concept of functional programming languages



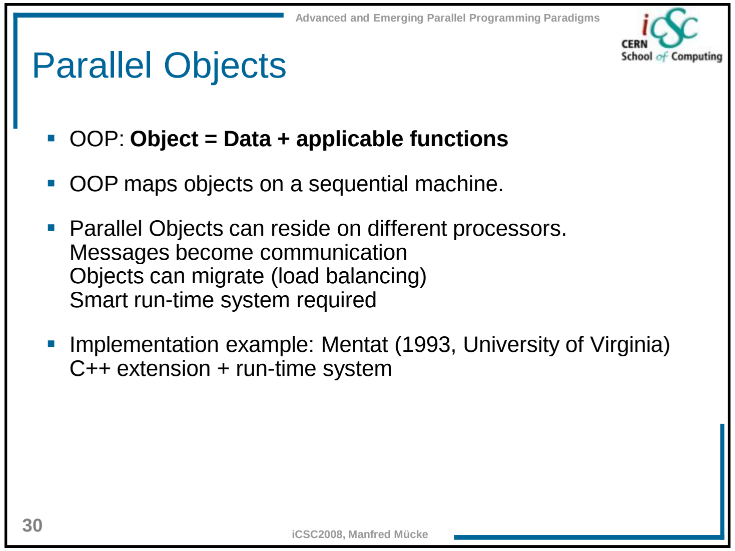
Functional Parallelism

- **Parallel sequential processes**
  Program is composed of sequential building blocks, which are executed in parallel.
  + Encapsulates sequential programming model
  - Defining correct interfaces is crucial

  ⇨ fork/merge
  ⇨ pThreads

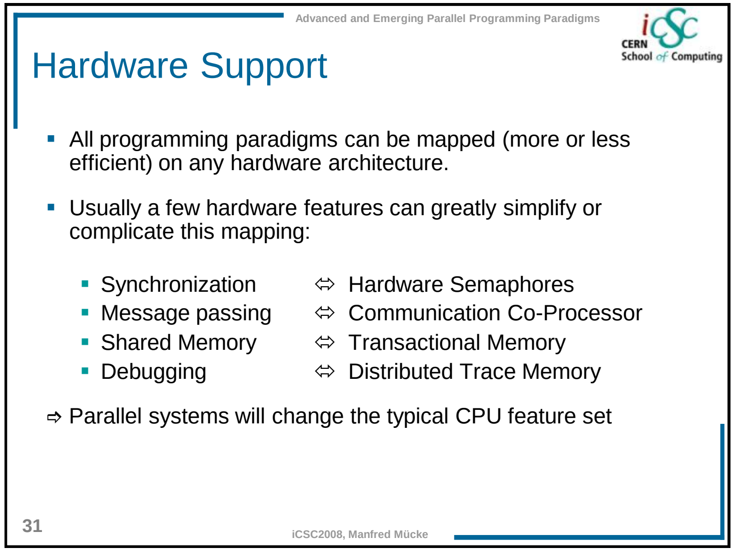
- **Sequential parallel processes**
  Program is composed of inherently parallel building blocks, which are executed sequentially
  + Exploits well fine/medium-grained parallelism
  + Merging of neighbouring operations possible

  ⇨ Parallel Skeletons



Parallel Objects

- OOP: **Object = Data + applicable functions**
- OOP maps objects on a sequential machine.
- Parallel Objects can reside on different processors.
  Messages become communication
  Objects can migrate (load balancing)
  Smart run-time system required
- Implementation example: Mentat (1993, University of Virginia)
  C++ extension + run-time system

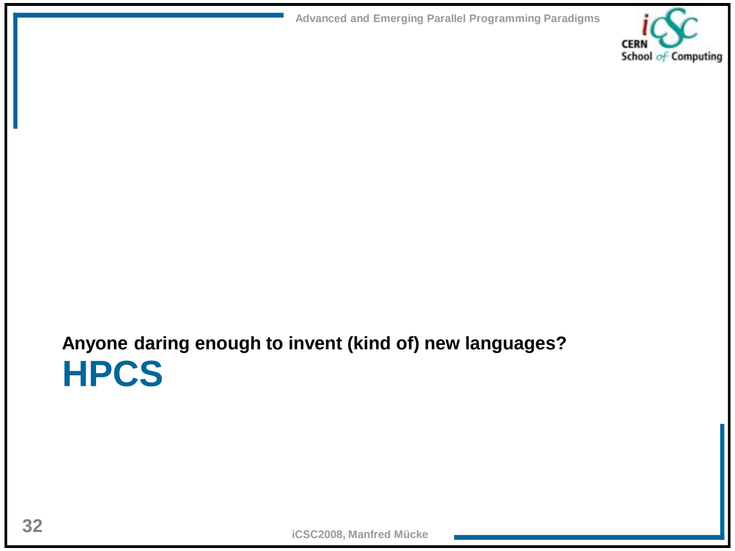


Hardware Support

- All programming paradigms can be mapped (more or less efficient) on any hardware architecture.
- Usually a few hardware features can greatly simplify or complicate this mapping:

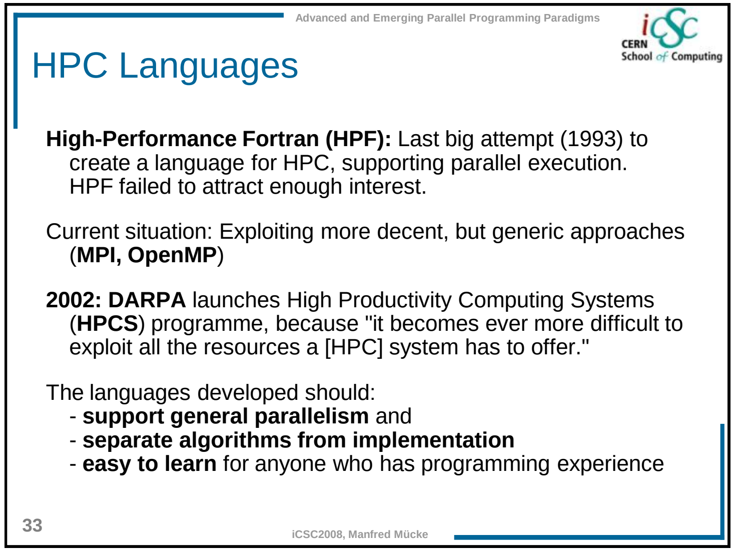
Synchronization	⇔ Hardware Semaphores
Message passing	⇔ Communication Co-Processor
Shared Memory	⇔ Transactional Memory
Debugging	⇔ Distributed Trace Memory

⇨ Parallel systems will change the typical CPU feature set



Anyone daring enough to invent (kind of) new languages?

HPCS



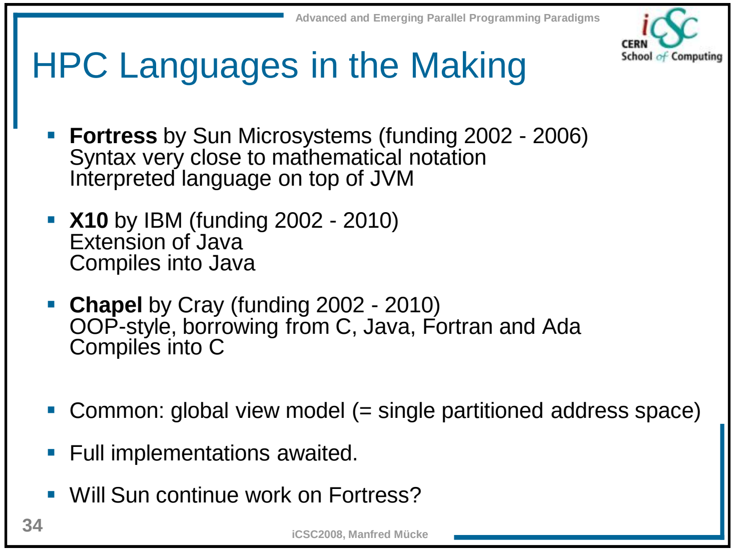
HPC Languages

High-Performance Fortran (HPF): Last big attempt (1993) to create a language for HPC, supporting parallel execution. HPF failed to attract enough interest.

Current situation: Exploiting more decent, but generic approaches (**MPI, OpenMP**)

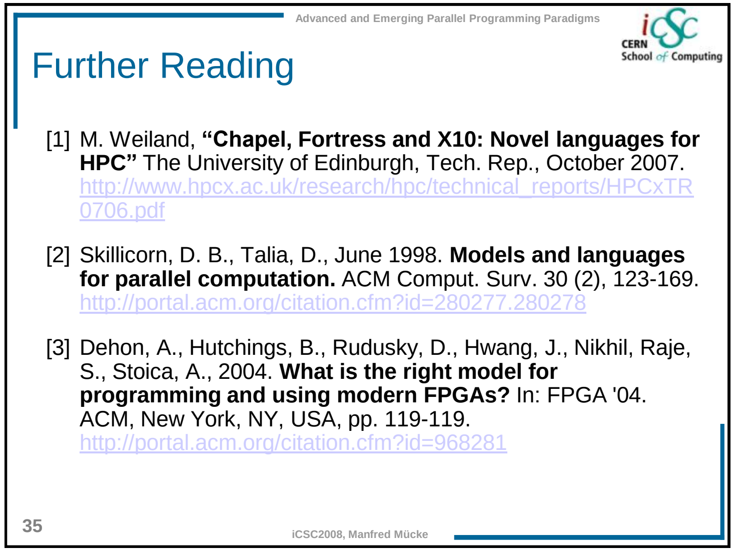
2002: DARPA launches High Productivity Computing Systems (**HPCS**) programme, because "it becomes ever more difficult to exploit all the resources a [HPC] system has to offer."

The languages developed should:
- **support general parallelism** and
- **separate algorithms from implementation**
- **easy to learn** for anyone who has programming experience



HPC Languages in the Making

- **Fortress** by Sun Microsystems (funding 2002 - 2006)
  Syntax very close to mathematical notation
  Interpreted language on top of JVM
- **X10** by IBM (funding 2002 - 2010)
  Extension of Java
  Compiles into Java
- **Chapel** by Cray (funding 2002 - 2010)
  OOP-style, borrowing from C, Java, Fortran and Ada
  Compiles into C
- Common: global view model (= single partitioned address space)
- Full implementations awaited.
- Will Sun continue work on Fortress?

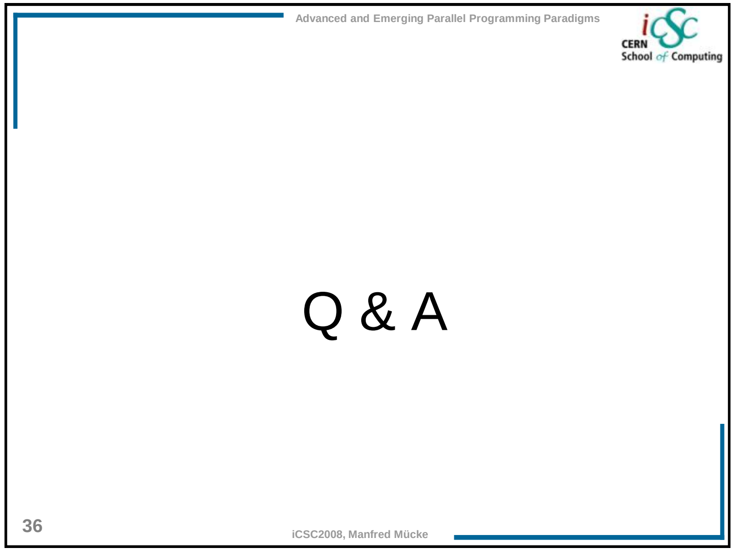


Further Reading

[1] M. Weiland, **“Chapel, Fortress and X10: Novel languages for HPC”** The University of Edinburgh, Tech. Rep., October 2007. http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0706.pdf

[2] Skillicorn, D. B., Talia, D., June 1998. **Models and languages for parallel computation.** ACM Comput. Surv. 30 (2), 123-169. http://portal.acm.org/citation.cfm?id=280277.280278

[3] Dehon, A., Hutchings, B., Rudusky, D., Hwang, J., Nikhil, Raje, S., Stoica, A., 2004. **What is the right model for programming and using modern FPGAs?** In: FPGA '04. ACM, New York, NY, USA, pp. 119-119. http://portal.acm.org/citation.cfm?id=968281

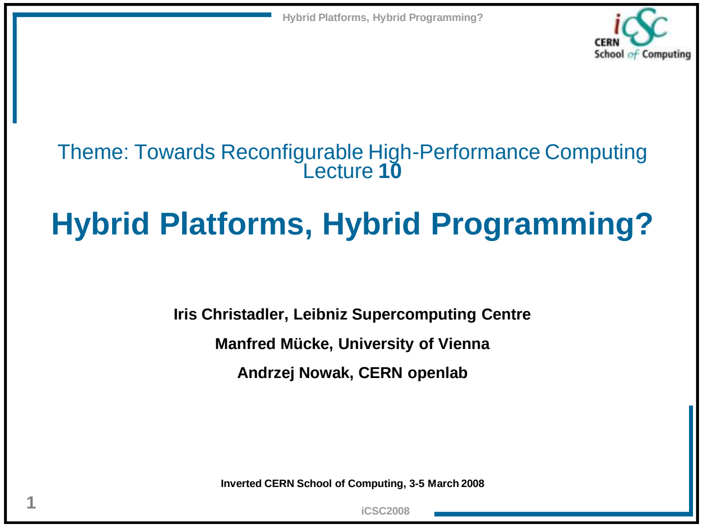


Q & A

LECTURE 10

Summary: Hybrid Platforms, Hybrid Programming?

Tuesday 4 March 2008			
15:00 16:00	Lecture 10	This lecture will address future prospects of hybrid platforms. It will specify what is necessary to make Reconfigurable HPC a success. Important topics: • Existing hybrid platforms • Existing hybrid programming models • Future programming models • Necessary tools **Audience and Pre-requisite** This lecture is both a summary and an outlook. It addresses participants who attended Lectures 7 and 8 about "Reconfigurable HPC" and Lecture 9, Advanced and Emerging Parallel Programming Paradigms.	**Iris Christadler**



Theme: Towards Reconfigurable High-Performance Computing
Lecture **10**

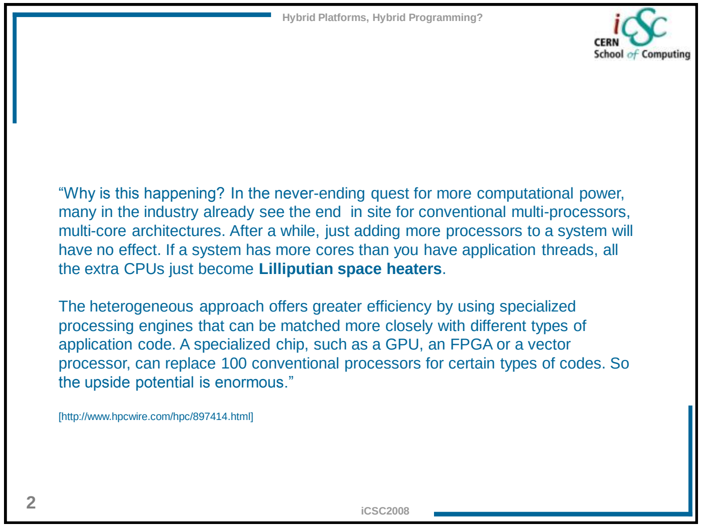
Hybrid Platforms, Hybrid Programming?

Iris Christadler, Leibniz Supercomputing Centre

Manfred Mücke, University of Vienna

Andrzej Nowak, CERN openlab

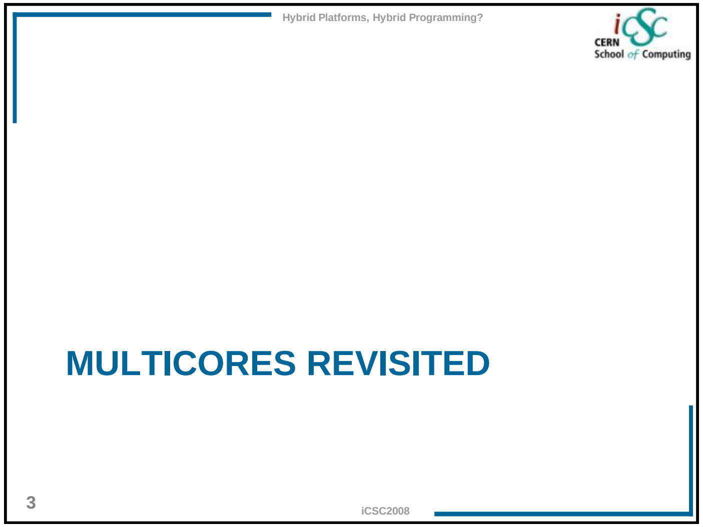
Inverted CERN School of Computing, 3-5 March 2008



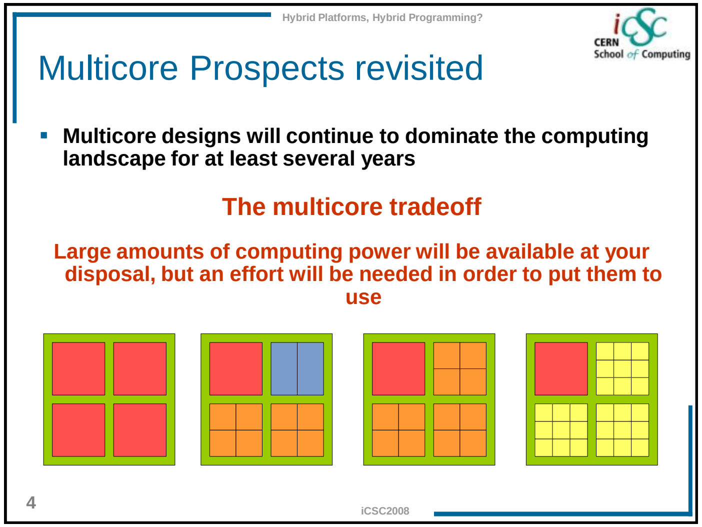
"Why is this happening? In the never-ending quest for more computational power, many in the industry already see the end in site for conventional multi-processors, multi-core architectures. After a while, just adding more processors to a system will have no effect. If a system has more cores than you have application threads, all the extra CPUs just become **Lilliputian space heaters**.

The heterogeneous approach offers greater efficiency by using specialized processing engines that can be matched more closely with different types of application code. A specialized chip, such as a GPU, an FPGA or a vector processor, can replace 100 conventional processors for certain types of codes. So the upside potential is enormous."

[http://www.hpcwire.com/hpc/897414.html]



MULTICORES REVISITED

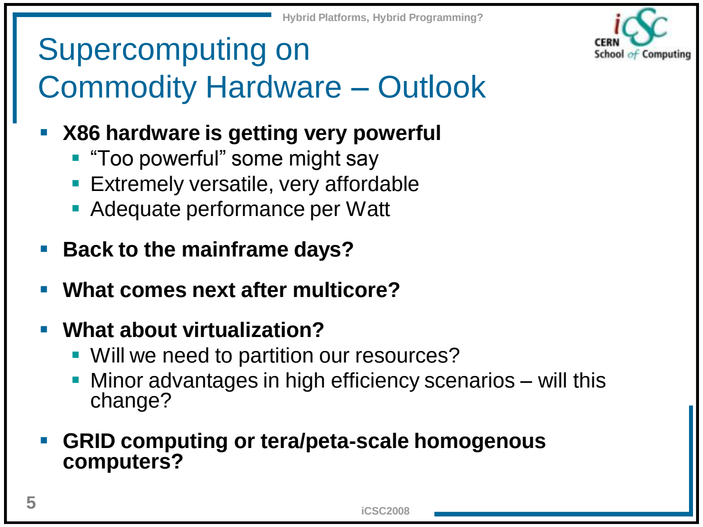


Multicore Prospects revisited

- **Multicore designs will continue to dominate the computing landscape for at least several years**

The multicore tradeoff

Large amounts of computing power will be available at your disposal, but an effort will be needed in order to put them to use



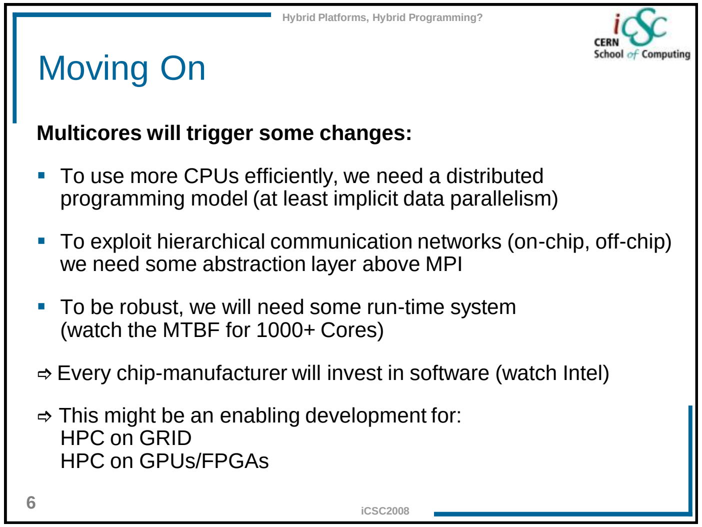
Hybrid Platforms, Hybrid Programming?

CERN School of Computing

Supercomputing on Commodity Hardware – Outlook

- **X86 hardware is getting very powerful**
  - "Too powerful" some might say
  - Extremely versatile, very affordable
  - Adequate performance per Watt
- **Back to the mainframe days?**
- **What comes next after multicore?**
- **What about virtualization?**
  - Will we need to partition our resources?
  - Minor advantages in high efficiency scenarios – will this change?
- **GRID computing or tera/peta-scale homogenous computers?**

5 iCSC2008



Hybrid Platforms, Hybrid Programming?

CERN School of Computing

Moving On

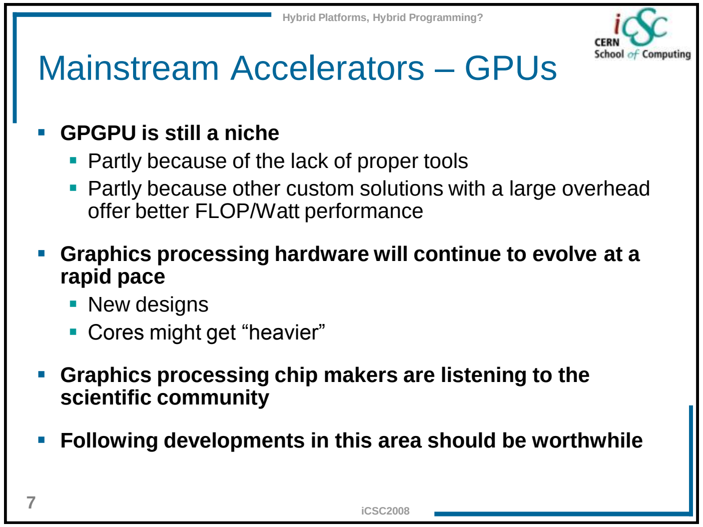
Multicores will trigger some changes:

- To use more CPUs efficiently, we need a distributed programming model (at least implicit data parallelism)
- To exploit hierarchical communication networks (on-chip, off-chip) we need some abstraction layer above MPI
- To be robust, we will need some run-time system (watch the MTBF for 1000+ Cores)

⇨ Every chip-manufacturer will invest in software (watch Intel)

⇨ This might be an enabling development for:
HPC on GRID
HPC on GPUs/FPGAs

6 iCSC2008



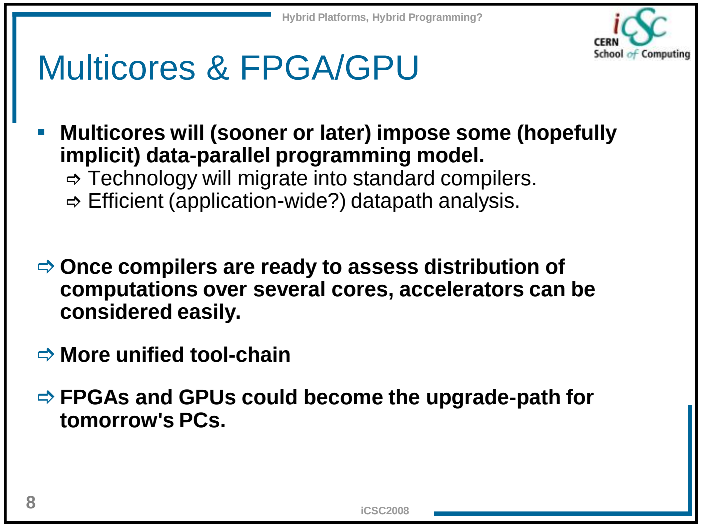
Hybrid Platforms, Hybrid Programming?

CERN School of Computing

Mainstream Accelerators – GPUs

- **GPGPU is still a niche**
  - Partly because of the lack of proper tools
  - Partly because other custom solutions with a large overhead offer better FLOP/Watt performance
- **Graphics processing hardware will continue to evolve at a rapid pace**
  - New designs
  - Cores might get "heavier"
- **Graphics processing chip makers are listening to the scientific community**
- **Following developments in this area should be worthwhile**

7 iCSC2008



Hybrid Platforms, Hybrid Programming?

CERN School of Computing

Multicores & FPGA/GPU

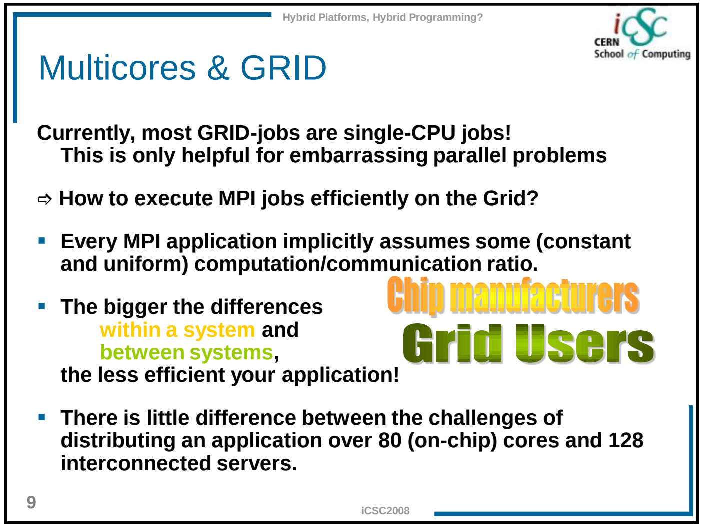
- **Multicores will (sooner or later) impose some (hopefully implicit) data-parallel programming model.**
  ⇨ Technology will migrate into standard compilers.
  ⇨ Efficient (application-wide?) datapath analysis.

⇨ **Once compilers are ready to assess distribution of computations over several cores, accelerators can be considered easily.**

⇨ **More unified tool-chain**

⇨ **FPGAs and GPUs could become the upgrade-path for tomorrow's PCs.**

8 iCSC2008



Multicores & GRID

Currently, most GRID-jobs are single-CPU jobs!
This is only helpful for embarrassing parallel problems

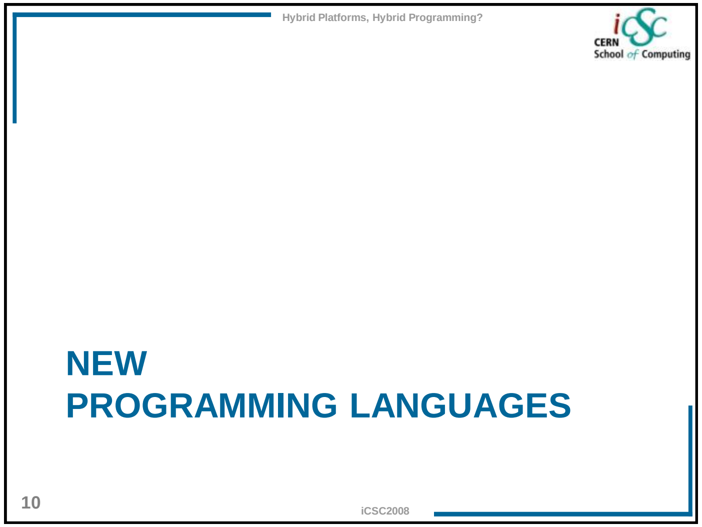
⇨ How to execute MPI jobs efficiently on the Grid?

- **Every MPI application implicitly assumes some (constant and uniform) computation/communication ratio.**
- **The bigger the differences**
  within a system and
  between systems,
  the less efficient your application!

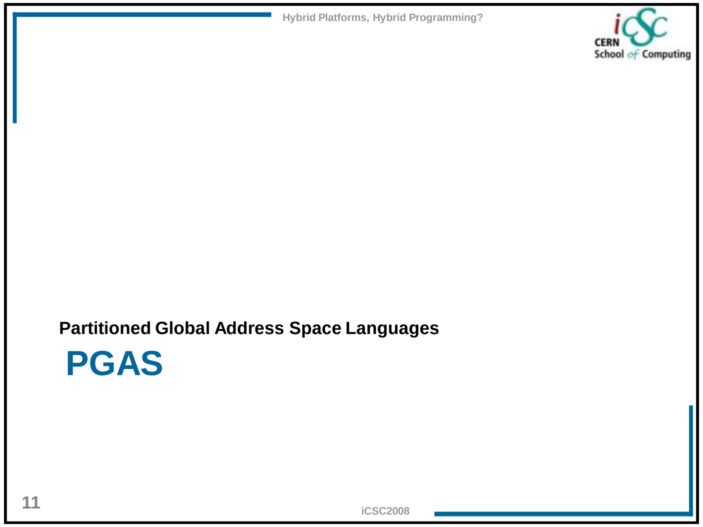
Chip manufacturers

Grid Users

- **There is little difference between the challenges of distributing an application over 80 (on-chip) cores and 128 interconnected servers.**

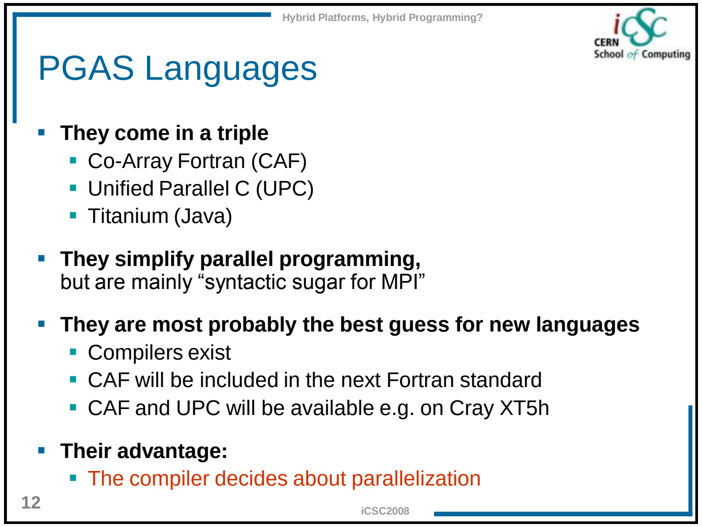


NEW PROGRAMMING LANGUAGES



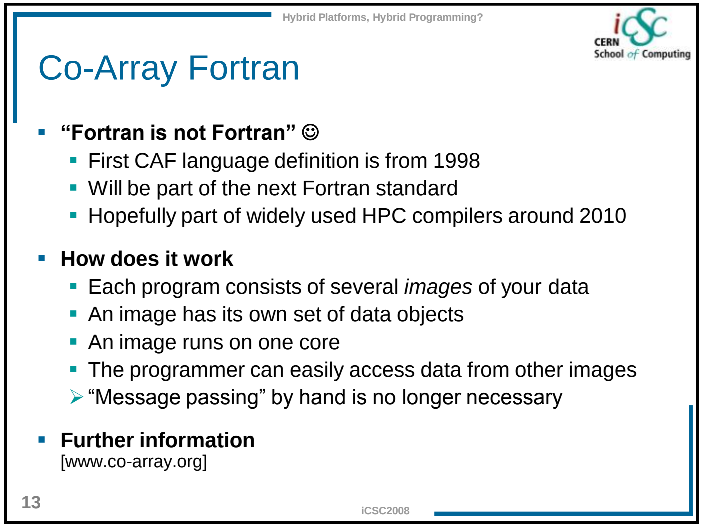
Partitioned Global Address Space Languages

PGAS



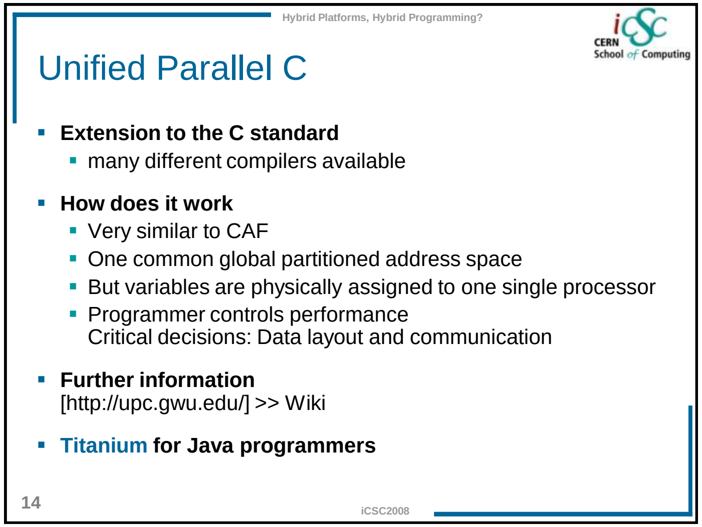
PGAS Languages

- **They come in a triple**
  - Co-Array Fortran (CAF)
  - Unified Parallel C (UPC)
  - Titanium (Java)
- **They simplify parallel programming,**
  but are mainly “syntactic sugar for MPI”
- **They are most probably the best guess for new languages**
  - Compilers exist
  - CAF will be included in the next Fortran standard
  - CAF and UPC will be available e.g. on Cray XT5h
- **Their advantage:**
  - The compiler decides about parallelization



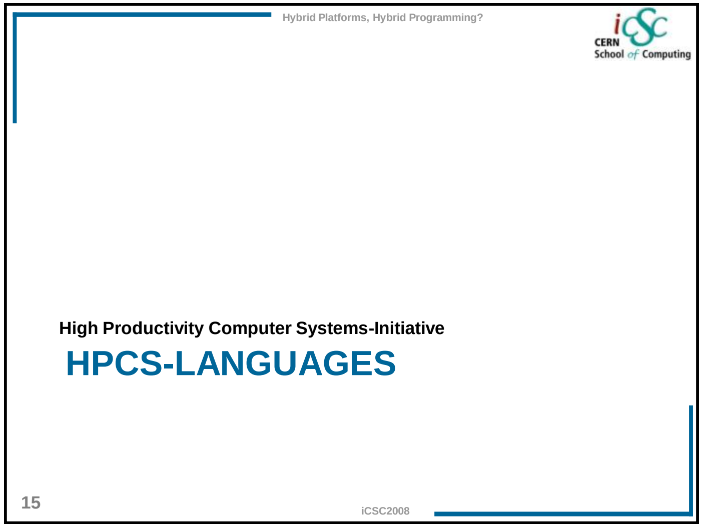
Co-Array Fortran

- **“Fortran is not Fortran” ☺**
  - First CAF language definition is from 1998
  - Will be part of the next Fortran standard
  - Hopefully part of widely used HPC compilers around 2010
- **How does it work**
  - Each program consists of several *images* of your data
  - An image has its own set of data objects
  - An image runs on one core
  - The programmer can easily access data from other images
  - ➢ “Message passing” by hand is no longer necessary
- **Further information**
  [www.co-array.org]



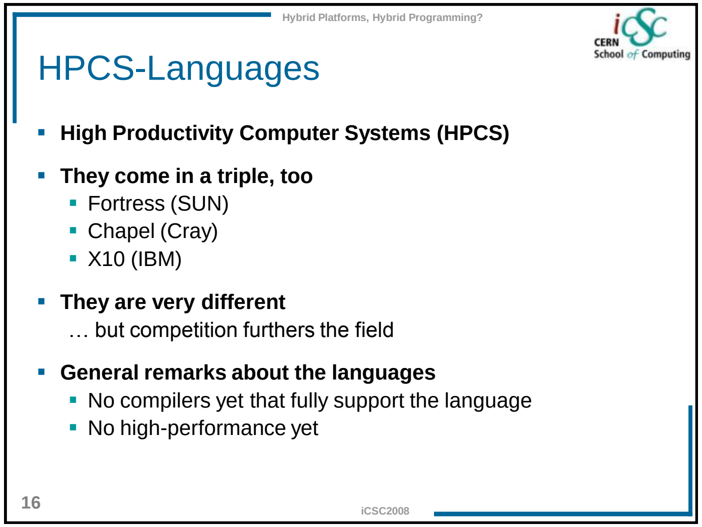
Unified Parallel C

- **Extension to the C standard**
  - many different compilers available
- **How does it work**
  - Very similar to CAF
  - One common global partitioned address space
  - But variables are physically assigned to one single processor
  - Programmer controls performance
    Critical decisions: Data layout and communication
- **Further information**
  [http://upc.gwu.edu/] >> Wiki
- **Titanium for Java programmers**



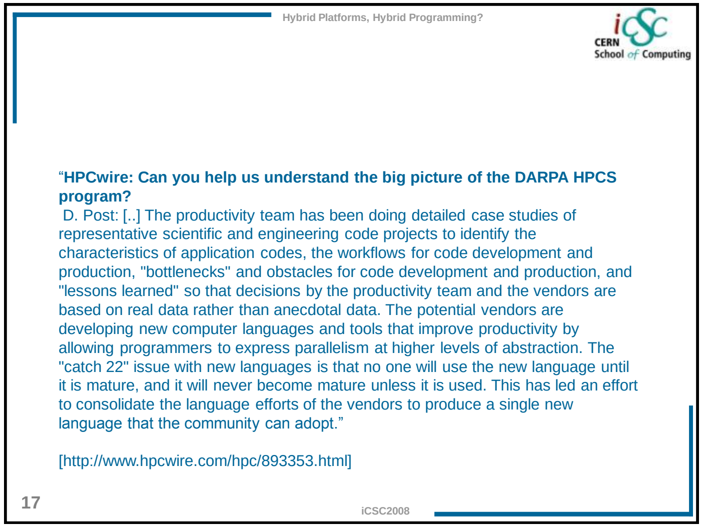
High Productivity Computer Systems-Initiative

HPCS-LANGUAGES



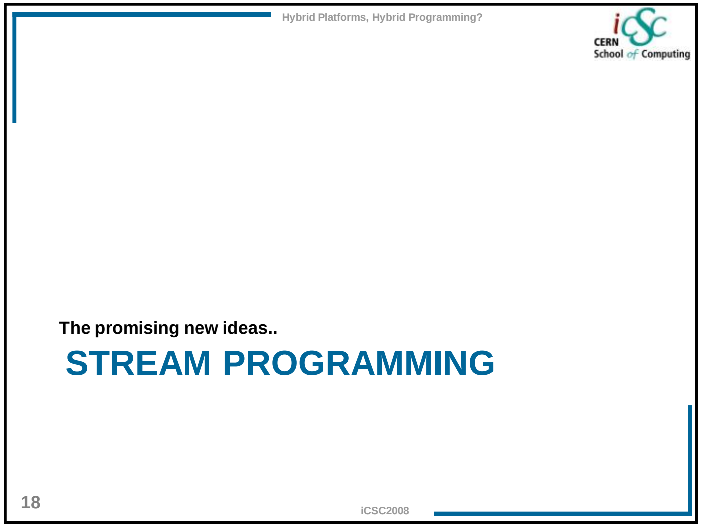
HPCS-Languages

- **High Productivity Computer Systems (HPCS)**
- **They come in a triple, too**
  - Fortress (SUN)
  - Chapel (Cray)
  - X10 (IBM)
- **They are very different**
  … but competition furthers the field
- **General remarks about the languages**
  - No compilers yet that fully support the language
  - No high-performance yet



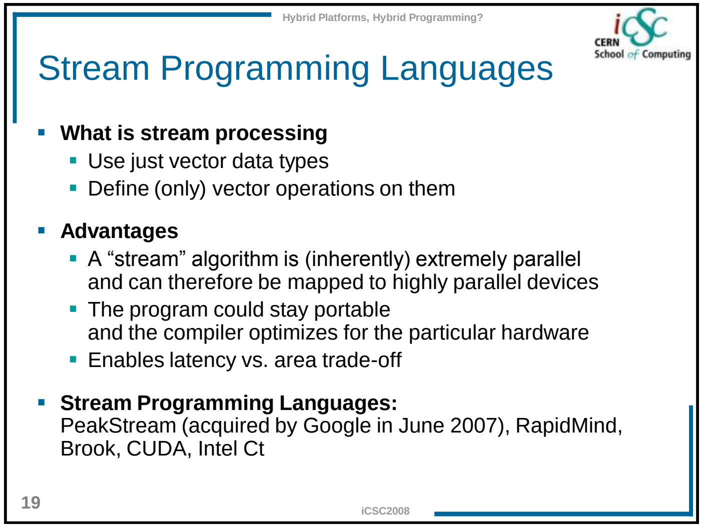
“HPCwire: Can you help us understand the big picture of the DARPA HPCS program?
D. Post: [..] The productivity team has been doing detailed case studies of representative scientific and engineering code projects to identify the characteristics of application codes, the workflows for code development and production, "bottlenecks" and obstacles for code development and production, and "lessons learned" so that decisions by the productivity team and the vendors are based on real data rather than anecdotal data. The potential vendors are developing new computer languages and tools that improve productivity by allowing programmers to express parallelism at higher levels of abstraction. The "catch 22" issue with new languages is that no one will use the new language until it is mature, and it will never become mature unless it is used. This has led an effort to consolidate the language efforts of the vendors to produce a single new language that the community can adopt.”

[http://www.hpcwire.com/hpc/893353.html]



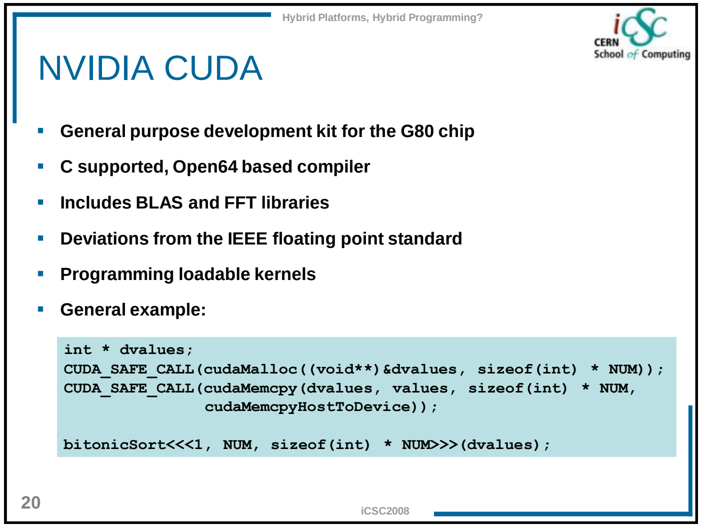
The promising new ideas..

STREAM PROGRAMMING



Stream Programming Languages

- **What is stream processing**
  - Use just vector data types
  - Define (only) vector operations on them
- **Advantages**
  - A “stream” algorithm is (inherently) extremely parallel and can therefore be mapped to highly parallel devices
  - The program could stay portable and the compiler optimizes for the particular hardware
  - Enables latency vs. area trade-off
- **Stream Programming Languages:**
  PeakStream (acquired by Google in June 2007), RapidMind, Brook, CUDA, Intel Ct



NVIDIA CUDA

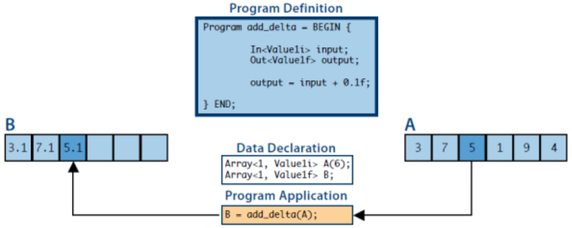
- **General purpose development kit for the G80 chip**
- **C supported, Open64 based compiler**
- **Includes BLAS and FFT libraries**
- **Deviations from the IEEE floating point standard**
- **Programming loadable kernels**
- **General example:**

```
int * dvalues;
CUDA_SAFE_CALL(cudaMalloc((void**)&dvalues, sizeof(int) * NUM));
CUDA_SAFE_CALL(cudaMemcpy(dvalues, values, sizeof(int) * NUM,
                cudaMemcpyHostToDevice));

bitonicSort<<<1, NUM, sizeof(int) * NUM>>>(dvalues);
```

RapidMind

- Multicore and GPGPU development platform
- An API library for C++ with additions
- Numerous backends – x86, GPU, CELL



```
Program add_delta = BEGIN {

        In<Value1i> input;
        Out<Value1f> output;

        output = input + 0.1f;

} END;
```

3.1	7.1	5.1			

3	7	5	1	9	4

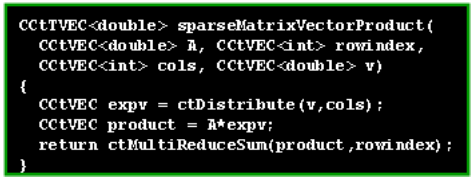
```
Array<1, Value1i> A(6);
Array<1, Value1f> B;
```

```
B = add_delta(A);
```

Graphics: Rapid Mind

Intel Ct

- An experimental data parallel programming environment
- Designed to facilitate multicore programming and increase portability
- Best with vectors, sparse matrices, trees, linked lists
- Example:



```
CCtTVEC<double> sparseMatrixVectorProduct(
  CCtVEC<double> A, CCtVEC<int> rowindex,
  CCtVEC<int> cols, CCtVEC<double> v)
{
  CCtVEC expv = ctDistribute(v,cols);
  CCtVEC product = A*expv;
  return ctMultiReduceSum(product,rowindex);
}
```

Intel STM

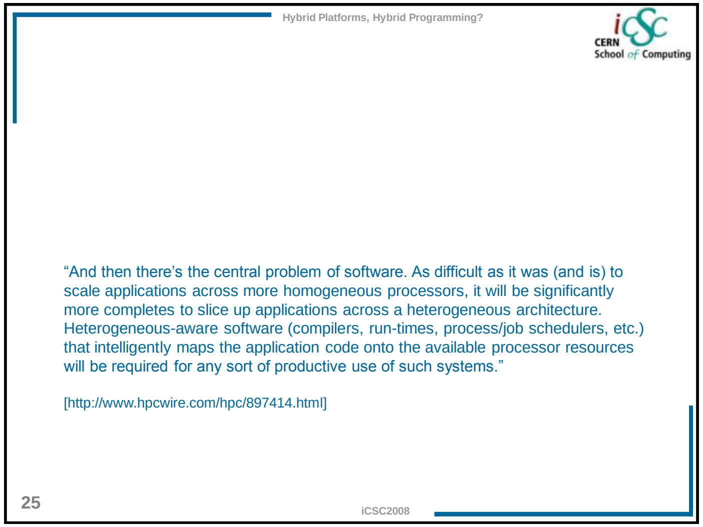
- A prototype version of the ICC C/C++ compiler
- Added transactional programming constructs
- Basic construct: __tm_atomic { *statements*; }
- Examples:

```
void foo(void){
    __tm_atomic {
    stmt1;
    stmt2;
    }
}
```

```
void func(void)
{
    try {
      __tm_atomic {
        stmt1;
        exception
        stmt2;
      } TM_HANDLER
    } catch/except {
      .........
    }
}
```

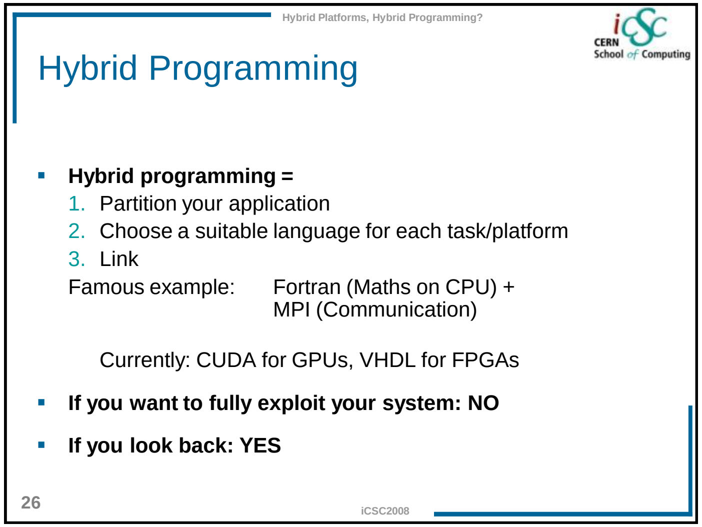
Hybrid Systems,

HYBRID PROGRAMMING?



“And then there’s the central problem of software. As difficult as it was (and is) to scale applications across more homogeneous processors, it will be significantly more completes to slice up applications across a heterogeneous architecture. Heterogeneous-aware software (compilers, run-times, process/job schedulers, etc.) that intelligently maps the application code onto the available processor resources will be required for any sort of productive use of such systems.”

[http://www.hpcwire.com/hpc/897414.html]

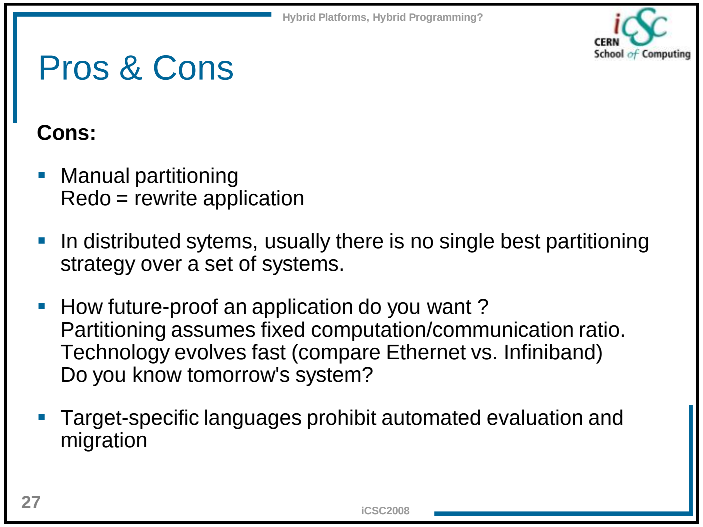


Hybrid Programming

- **Hybrid programming =**
  1. Partition your application
  2. Choose a suitable language for each task/platform
  3. Link

  Famous example: Fortran (Maths on CPU) +
  MPI (Communication)

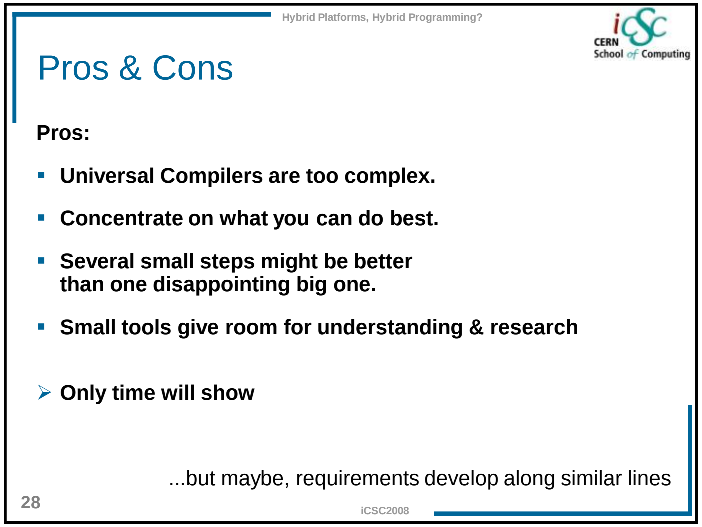
  Currently: CUDA for GPUs, VHDL for FPGAs
- **If you want to fully exploit your system: NO**
- **If you look back: YES**



Pros & Cons

Cons:

- Manual partitioning
  Redo = rewrite application
- In distributed sytems, usually there is no single best partitioning strategy over a set of systems.
- How future-proof an application do you want ?
  Partitioning assumes fixed computation/communication ratio.
  Technology evolves fast (compare Ethernet vs. Infiniband)
  Do you know tomorrow's system?
- Target-specific languages prohibit automated evaluation and migration



Pros & Cons

Pros:

- **Universal Compilers are too complex.**
- **Concentrate on what you can do best.**
- **Several small steps might be better than one disappointing big one.**
- **Small tools give room for understanding & research**

➢ **Only time will show**

...but maybe, requirements develop along similar lines

HOW TO CONNECT THE WORLDS OF HPC AND FPGAS?

Worlds Apart!?

- **HPC brings in very different requirements compared to embedded/DSP systems**
- **DSP: Speed is everything!**
  Embedded: Balance price/power/size
- **HPC: As fast as possible**
  FLOPS/$, FLOPS/W...
- **Biggest difference: Design effort / FPGA**

⇨ Enable Tools to make sufficient good trade-offs

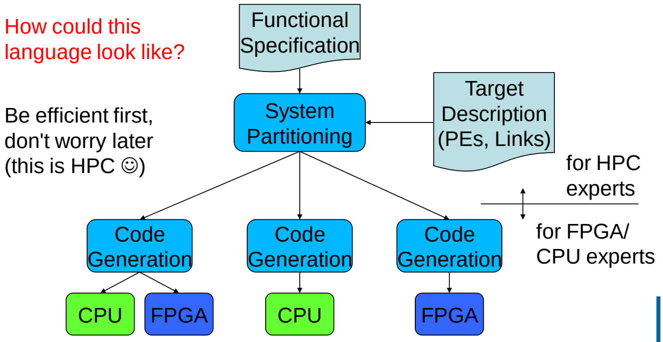
⇨ Tools rely on suitable input specifications

How To Connect…

What FPGAs can deliver to HPC:

- **Very fast implementation of static data-paths**
- **Very fast and wide on/off-chip communication (on-chip distributed memories, dedicated links)**
- **2 Approaches**
- **Use FPGA as coprocessor**
  ⇨ local, late in design process
- **Consider FPGA at task distribution**
  ⇨ global, early in design process

How To Connect…



How could this language look like?

Be efficient first, don't worry later (this is HPC ☺)

OUTLOOK

"HPCwire: You mentioned you're not really interested in FPGAs as accelerators. Why is that?"
D. Turek (vice president of Deep Computing at IBM):
"[…] I'm not convinced that the software tools and the other things you need for programming them will ever make it, fundamentally."

[http://www.hpcwire.com/hpc/893353.html]

The Application Mix of the Future

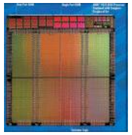
- **Which codes are able to scale to 10000+ processors?**
- **Which codes can benefit from special accelerators?**
- **Who determines if a code is scalable?**
  - The research area,
  - the problem itself,
  - the chosen algorithm or
  - the written code ?
- **What are we going to do with codes that are already hitting scalability limits?**
- **How can we convince people to use better programming languages?**

Accelerators – FPGAs

- **A bright future ahead**
- **Large investments required on the software front**
- **Using this technology usually required advanced knowledge and introduces a large overhead, but the benefits are potentially enormous**
- **More advances relating to double precision issues needed**

Tomorrow's Hardware?

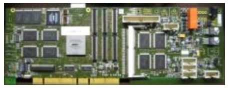
- **FPGAs and CPUs complement eachother!**
- **Why not combining them on a single chip?**
- **Xilinx (PowerPC) and Altera (ARM) tried, but failed (economically).**
- **Softcores triggered impressive tool development.**
- **With better tools, maybe a second try is due!**



Altera Excalibur EXPA10 die

Today's Resources

- Is **CERN** a good place to think about how FPGAs could be used for number crunching?
- It certainly features abundant FPGA resources (hidden in LHC DAQ systems)!
- What will they do during LHC **winter shutdown**?



Acknowledgments

Fabienne Baud-Lavigne, CERN
Francois Flueckiger, CERN
Ivica Puljak, Univ. of Split, Croatia
Sverre Jarp, CERN openlab
Dr. Matthias Brehm, LRZ, Munich, Germany
Markus Stürmer, Univ. of Erlangen, Germany

Special Topics: Fundamentals and Best Practices

CERN School of Computing

iCSC2008 Special Topics:
Fundamentals and Best Practices

Lecturers:

Jose Dana Perez - CERN
Alfio Lazzaro - University of Milan and INFN, Milan – Italy

To complement the iCSC2008's main theme, two special topics though different have been selected. Each of them may attract a specific attendance interested in the latest development of these two domains.

The two special topics:

- Overview of advanced aspects of data analysis software and techniques
- Scalable Image and Video coding

A few questions

- *Do Do you know how to speed up the execution of your data analysis program?*
- *Do you want to improve your data analysis technique for signal/background discrimination?*
- *Do you know the last applications of advanced data analysis techniques (Neural Network, Decision Tree,…) in High Energy Physics?*

All the answers at **iCSC**

Overview

Slot	Lecture	Description	Lecturer
Wednesday 5 March 2008			
09:00 - 10:00	Lecture 1	Overview of advanced aspects of data analysis software and techniques	Alfio Lazzaro
10:00-10:30		Coffee	
10:30 - 12:00	Lecture 2	Scalable Image and Video coding	Jose Dana Perez
12:00		Adjourn	

iCSC CERN School of Computing

LECTURE 1

Overview of advanced aspects of data analysis software and techniques

Wednesday 5 march 2008			
09:00 09:55	Lecture 1	In this lecture we give an overview of the advanced data analysis techniques based on multivariate techniques, which are recently used in many High Energy Physics data analysis. The topic is relevant to many Particle Physics analyses, as well as in several other fields. We will give an over view on the different techniques and their relative merits. **Audience** This lecture targets an audience with experience in data analysis, in particular interested in techniques of signal/background discrimination **Pre-requisite** This lecture can be reasonably followed without having attended to the other lecturers of this school **Keywords** • Data analysis • Parallel processing • Signal Background Separation • Maximum Likelihood • Artificial Neural Network • Decision Tree	Alfio Lazzaro

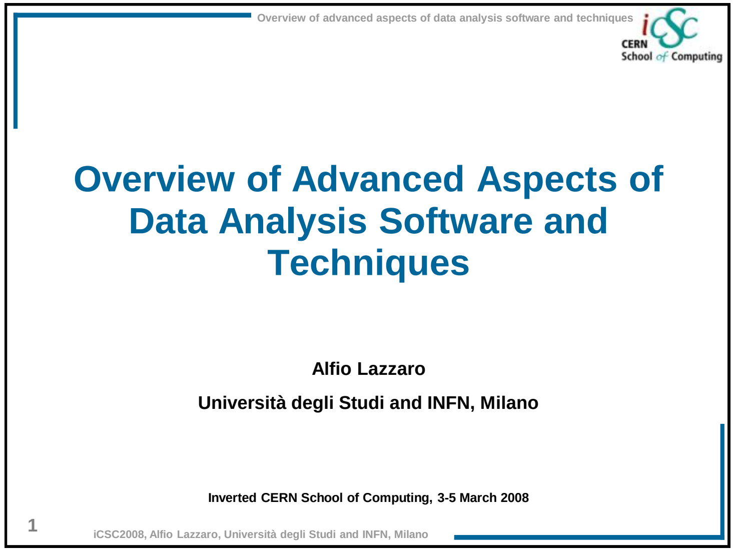
Details

In the past years, many advanced techniques in statistical data analysis have been used in High Energy Physics (such as maximum likelihood fits, Neural Networks, and Decision Trees). In the past, the most common technique was the simple cut and count analysis. This technique consists in the following steps: several cuts are applied on well studied discriminating variables, background estimation is performed using Monte Carlo simulation samples or events outside the signal region, and then the final measurement is done counting the events after cuts minus the estimated background events.

This simple technique is hampered by its low efficiency (defined as ratio between the number events after and before the cuts) and does not provide a good discrimination between signal and background events. For this reason it was replaced by more sophisticated techniques, such as the multivariate maximum likelihood for the measurements done at the BaBar experiment, running at Stanford Linear Accelerator Center (SLAC) in California.

The maximum likelihood (ML) technique permits to achieve higher efficiency, the possibility to take in account errors with better precisions, and consider correlations between the discriminating variables used in the analysis. Anyway, in future experiments, like LHC experiments at CERN, it may be crucial to have better discrimination between signal and background events to discover new phenomenas, which suffer higher background. Neural Networks and Decision Trees are good techniques to reach this goal. Another important issue to take into account lies in the fact that these techniques are in most cases very CPU-time consuming. It is possible to speed them up using concepts of High Performance Computing (HPC).

In this lecture we will give an overview of the advanced data analysis techniques mentioned above, introducing some software packages commonly used in HEP. This will be preceded by a short session at the end of the previous theme, giving briefly examples of possible HPC optimizations.

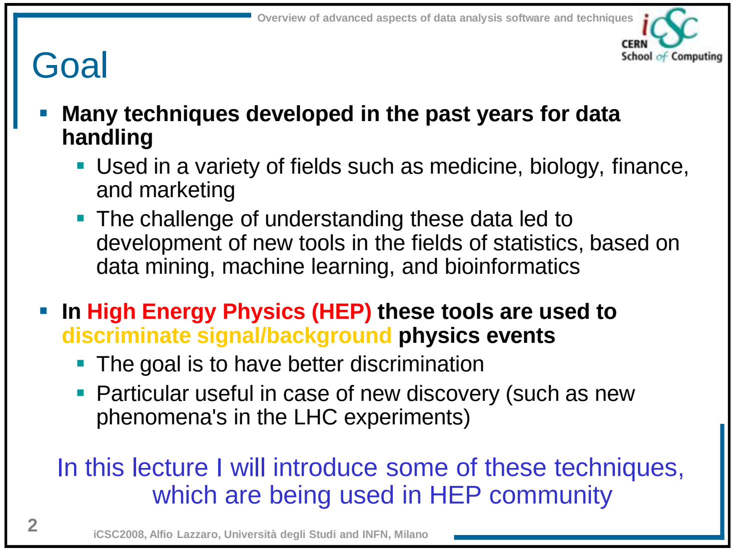


Overview of Advanced Aspects of Data Analysis Software and Techniques

Alfio Lazzaro

Università degli Studi and INFN, Milano

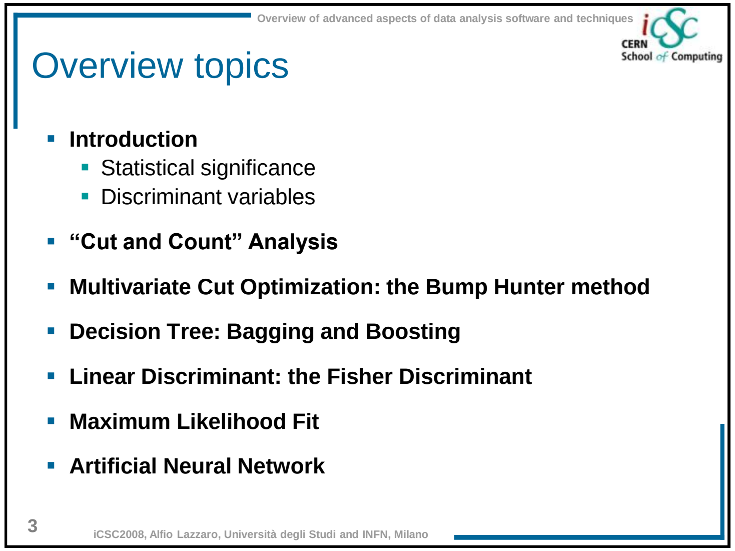
Inverted CERN School of Computing, 3-5 March 2008



Goal

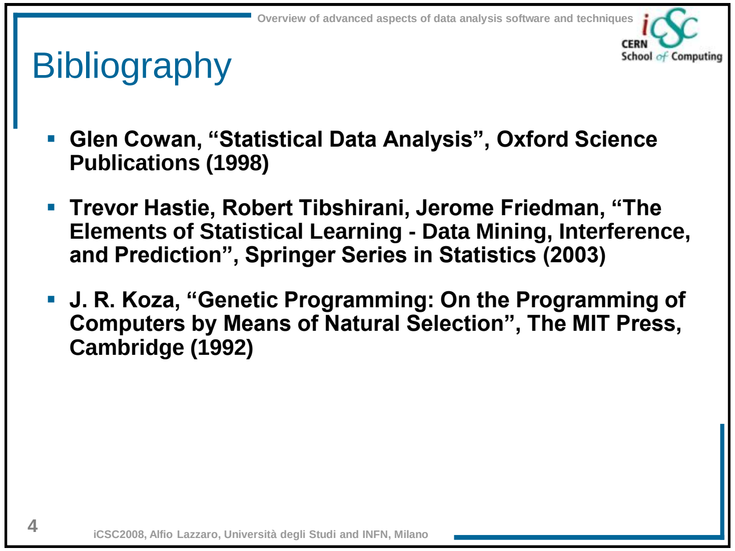
- **Many techniques developed in the past years for data handling**
  - Used in a variety of fields such as medicine, biology, finance, and marketing
  - The challenge of understanding these data led to development of new tools in the fields of statistics, based on data mining, machine learning, and bioinformatics
- **In High Energy Physics (HEP) these tools are used to discriminate signal/background physics events**
  - The goal is to have better discrimination
  - Particular useful in case of new discovery (such as new phenomena's in the LHC experiments)

In this lecture I will introduce some of these techniques, which are being used in HEP community



Overview topics

- **Introduction**
  - Statistical significance
  - Discriminant variables
- **“Cut and Count” Analysis**
- **Multivariate Cut Optimization: the Bump Hunter method**
- **Decision Tree: Bagging and Boosting**
- **Linear Discriminant: the Fisher Discriminant**
- **Maximum Likelihood Fit**
- **Artificial Neural Network**



Bibliography

- **Glen Cowan, “Statistical Data Analysis”, Oxford Science Publications (1998)**
- **Trevor Hastie, Robert Tibshirani, Jerome Friedman, “The Elements of Statistical Learning - Data Mining, Interference, and Prediction”, Springer Series in Statistics (2003)**
- **J. R. Koza, “Genetic Programming: On the Programming of Computers by Means of Natural Selection”, The MIT Press, Cambridge (1992)**

Software

- **ROOT**
  - General framework for Data Analysis
  - http://root.cern.ch
- **RooFit:**
  - Package for Multivariate Binned/Unbinned Maximum Likelihood fits
  - http://roofit.sourceforge.net/intro.html
- **TMVA -- Toolkit for Multivariate Data Analysis:**
  - General framework for Multivariate data analysis
  - http://tmva.sourceforge.net/
- **SPR -- StatPatternRecognition:**
  - General framework for Multivariate data analysis
  - http://sourceforge.net/projects/statpatrec

All code developed in C++, fully supported, well documented

Event hypotheses

- **Two hypotheses for events: signal and background (we can generalize for more event hypotheses)**
- **Goal: distinguish events belonging to the two hypotheses**
  - We measure some discrimant variables for each event, i.e. variables which allow to distinguish between hypotheses
- **We look for a technique which combines these variables for a better discrimination**
  - For a giving unknown events, general speaking we want to minimize the probability *P* that the event is due to background fluctuation, giving (1-*P*) as number of sigma of the Normal Distribution ==> maximization of the statistical significance
  - In HEP it is common to say 4σ (probability of 10^{-4} to be a background fluctuation) as evidence, 5σ (10^{-6}) as observation

Discriminant Variables

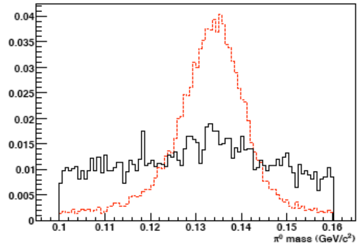
- **Discrimination Power:**
  - Good variables ==> Better discrimination
  - Bad variables ==> worse discrimination

Real world: something in the middle

- **We use Monte Carlo simulation or specific control samples to have samples of only signal and only background events**
  - We use these samples for training our discrimination method
  - Control sample should reproduce the "unknown" sample
    - Systematic uncertainties are introduced by wrong training samples
  - We need independent validation samples to check our discrimination method

Example of Discriminant Variables

- **We measure several physics variables in each event**
  - Energies, directions, momenta, masses, …
- **We combine these informations to extract our results**
- **Example: Mass of a π^0 particle**
  - Red: signal
  - Black: background



Statistical Decision Theory

- **We have *d* variables *X* for each event, and we classify the events in several hypotheses *Y* (e.g. signal and background) depending of the input variables with a joint distribution *P*(*X*,*Y*)**
- **We seek a function *f* (*X*) for predicting *Y* for given values of the input *X***
  - In general *f* (X) has a specific form with some free parameters to be determinate using the training data
- **We use expected (squared) prediction error**

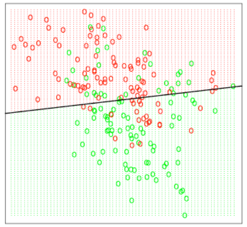
$$EPE(f) = E(Y - f(X))^2 = \int (y - f(x))^2 \, dP(x, y)$$

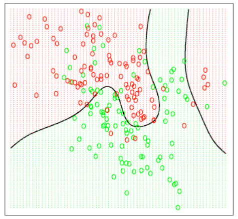
- **The *f* (*X*) free parameters are given by the minimization of EPE in each point**

Training and Validation

- **We use the training sample to find the free parameters of the function *f* (*X*)**
- **Question: How complex should the *f* (*X*) be?**

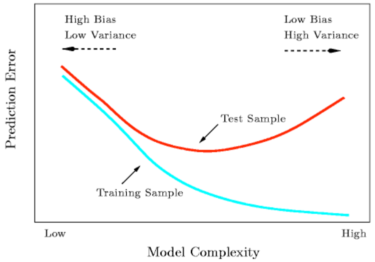
GREEN = Signal, RED = Background





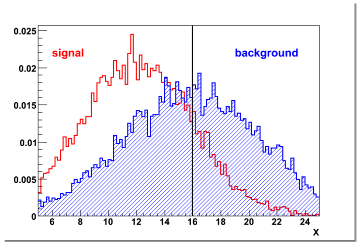
Overtraining

- **Complex model gives accurate prediction on training sample (low bias), but they can give wrong prediction on an independent validation sample (high variance)**
  - Complex model can reproduce all statistical fluctuation of the training sample: Overtraining
  - Use always a validation sample!



Single-variable “Cut and Count” Analysis

- **Consider the simple case of just 1 variable**

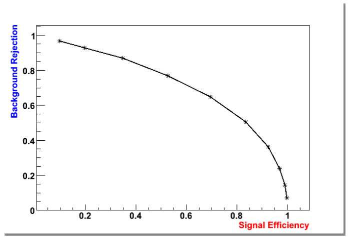


- **We apply a cut to separate signal from background (binary split)**
  - We identify the cut as a step function

$$f(x) = \begin{cases} 0, & x > cut \quad \text{background} \\ 1, & x \leq cut \quad \text{signal} \end{cases}$$

Single-variable Cut Optimization

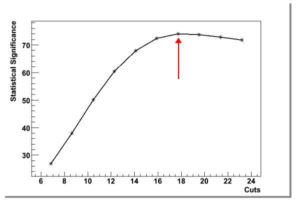
- **Depending on the value of the cut, we can define:**
  - Signal efficiency (SE): number of signal events after the cut divided by number of events before the cut
  - Background rejection (BR): number of background events which are eliminated by the cut divided number of events before the cut
- **We can plot SE versus BR for several applied cut values**
- **Very fast way to understand the effect of a cut on signal and background samples and to compare cut on different variables**
  - higher curve ==> better discrimination



Single-variable Cut Optimization

- **We can chose the value of the cut using the statistical significance maximization method**
  - We calculate the quantity $SS = \frac{S}{\sqrt{S+B}}$

    as function of the applied cuts, where *S* and *B* are the expected number of signal and background events after the cut, respectively.
  - The best cut is found for the maximum value of *SS* using a training samples
  - Independent validation samples are used to obtain the efficiency and background rejection of the cut

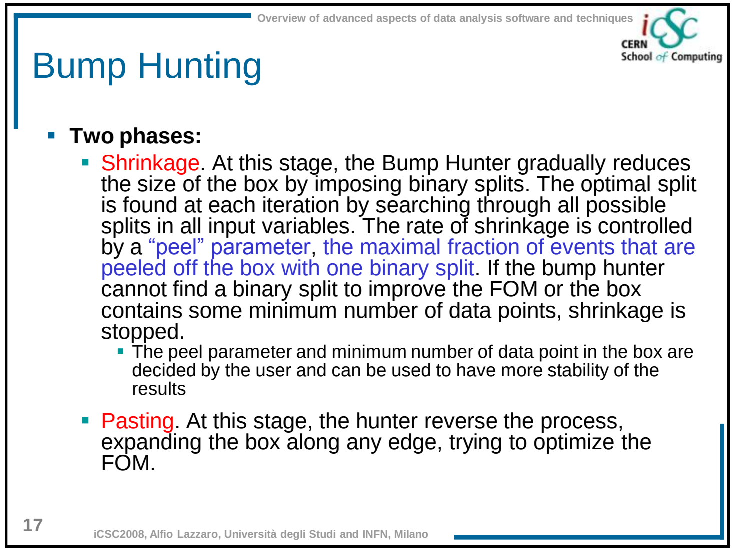


Multi-variable Cut Optimization

- **Generalization to multivariable cut optimization is not straightforward**
  - It is important to take in account correlations between the discriminant variables
  - Assuming totally uncorrelated variables, we can use the statistical significance maximization method for each variable, but in this way the order of the cuts becomes important for the optimization
- **More powerful methods are based on techniques which apply cuts in the whole variables space, searching the optimal separation between the event hypotheses**
  - Patient Rule Induction Method (PRIM)
  - Decision Tree

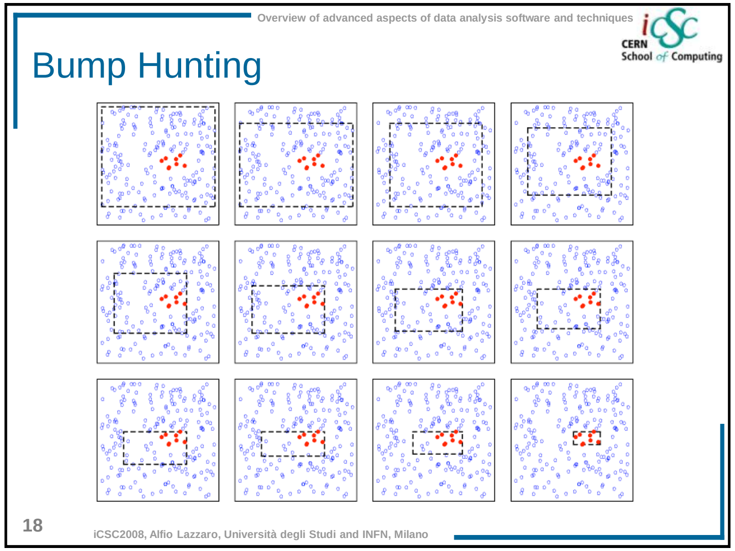
PRIM Cut Optimization

- **We consider two variables: PRIM searches for rectangular regions of the two variables plane where the statistical significance is high**
  - This algorithm is known as Bump Hunting
  - The statistical significance used in the search (but other criteria can used as well, like maximization of the ratio signal over background number of events) is known as Figure Of Merit (FOM)
- **Generalization to case with more than 2 variables is straightforward:**
  - Cuts are applied in *d*-dimensional variables space to look for a hyperrectangle where the FOM is high

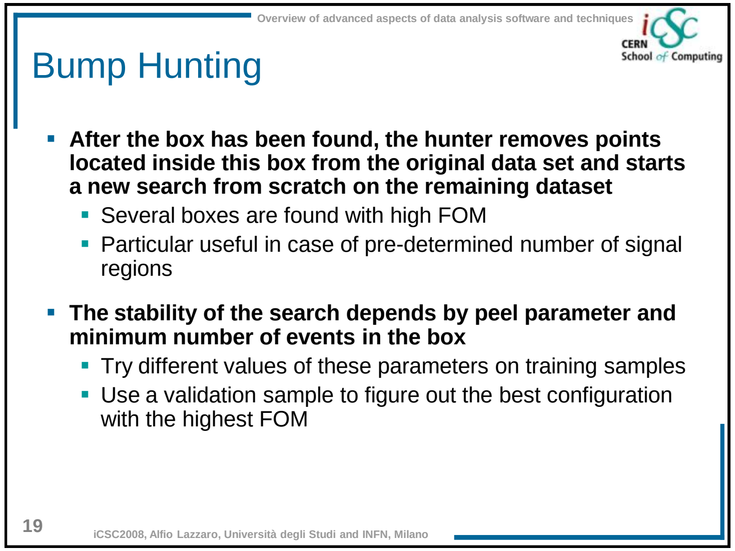


Bump Hunting

- **Two phases:**
  - Shrinkage. At this stage, the Bump Hunter gradually reduces the size of the box by imposing binary splits. The optimal split is found at each iteration by searching through all possible splits in all input variables. The rate of shrinkage is controlled by a "peel" parameter, the maximal fraction of events that are peeled off the box with one binary split. If the bump hunter cannot find a binary split to improve the FOM or the box contains some minimum number of data points, shrinkage is stopped.
    - The peel parameter and minimum number of data point in the box are decided by the user and can be used to have more stability of the results
  - Pasting. At this stage, the hunter reverse the process, expanding the box along any edge, trying to optimize the FOM.

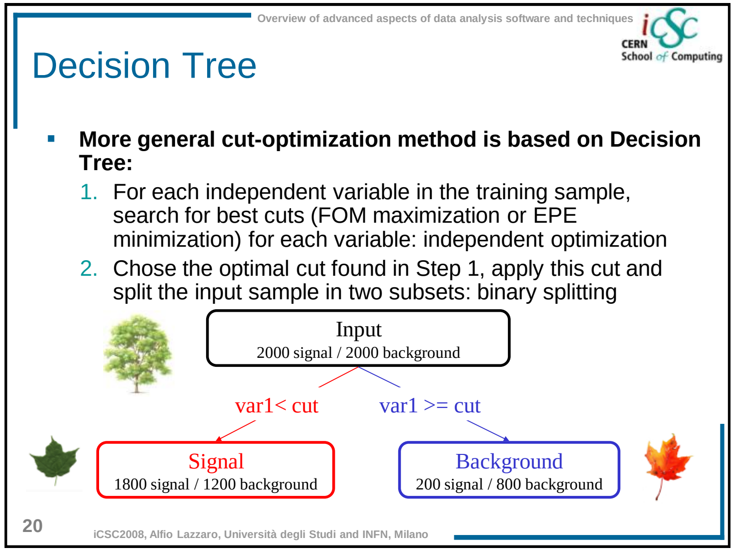


Bump Hunting



Bump Hunting

- **After the box has been found, the hunter removes points located inside this box from the original data set and starts a new search from scratch on the remaining dataset**
  - Several boxes are found with high FOM
  - Particular useful in case of pre-determined number of signal regions
- **The stability of the search depends by peel parameter and minimum number of events in the box**
  - Try different values of these parameters on training samples
  - Use a validation sample to figure out the best configuration with the highest FOM



Decision Tree

- **More general cut-optimization method is based on Decision Tree:**
  1. For each independent variable in the training sample, search for best cuts (FOM maximization or EPE minimization) for each variable: independent optimization
  2. Chose the optimal cut found in Step 1, apply this cut and split the input sample in two subsets: binary splitting

Input
2000 signal / 2000 background

var1< cut

var1 >= cut

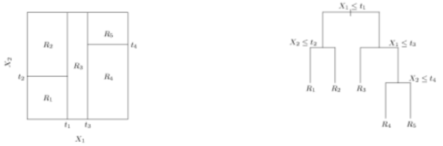
Signal
1800 signal / 1200 background

Background
200 signal / 800 background

Decision Tree

3. We repeat for each leaf from Step 1. In this way each leaf becomes a node, building a complex tree
4. Several methods can be used to stop training: generally the user specify the minimal number of events per tree node. The tree continues making new nodes until it is composed of leaves only — nodes that cannot be split without a decrease in the FOM and nodes that cannot be split because they have too few events.

- **At the end decision tree gives a partition of the input sample. Example in 2 variables:**

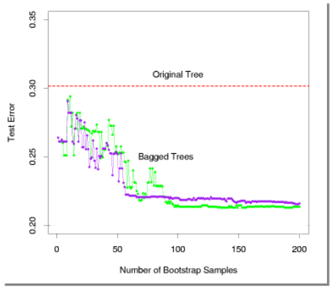


Boostrap aggregating or “Bagging”

- **From an enlarge sample of training events, we randomly choose events and we replace them in the primary training sample**
- **In this way we produce *B* different training samples**
- **We apply the optimization procedure of the decision tree for each sample, so we have *B* trees ==> Random Forrest**
  - Each tree is different
- **We examine the behavior of the fits over the *B* replications, making an average of the prediction for all samples or taking the tree with best FOM**
  - This reduce the variance of the prediction, giving better results

Boostrap aggregating or “Bagging”

- **We compare here the Test Error (EPE):**
  - for the Original Tree (Red)
  - Bagged Tree, using the best tree (Green)
  - Bagged Tree, using the average (Purple)
- **Advantages:**
  - Free from overtraining
  - Better than a single tree
- **Disadvantages:**
  - Require more computational time



Boosted Decision Tree

- **The purpose of boosting is to sequentially apply the decision tree algorithm to repeatedly modified version of the data, producing a sequence of trees ==> Random Forrest**
  - We can apply boosting in general for other discriminant methods, not only for Decision Trees
- **The prediction of all them are then combined to have a better prediction**

Boosted Decision Tree

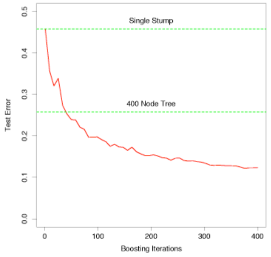
- **The most popular boosting algorithm is AdaBoost (Freund and Schapire, 1997)**
  1. For a giving data sample of *N* events, initialize each event with weight 1/*N*
  2. Run the Decision Tree over the training sample and compute the error *err*
  3. Compute the parameter $\alpha = \log((1\text{-}err)/err)$
  4. Use the parameter α for obtaining new weights for each event
  5. Repeat from Step 2 for *M* times
  6. The final answer after *M* steps is given by a combination of the answer of each tree, weighted by the correspondent α
- **We obtain a Random Forrest (combination of trees)**

Boosted Decision Tree

- **The α values, computed by the boosting algorithm for each boosting iteration (Step 3), give higher influence to the more accurate classifiers in the sequence**
- **The weights of events calculated in each boosting iteration (Step 4) are increased for the events which were misclassified in the previous iteration, whereas they are decreased for those that were classified correctly**
  - As iterations proceed, events that are difficult to be correctly identified receive ever-increase influence
  - Then each successive tree is forced to examine better training events which are missed by the previous ones in the sequence

Boosted Decision Tree

- **Very powerful technique:**
  - It allows to increase the performance of simply discriminant algorithms
  - For example, we take a simple “stump”: a two-terminal node tree, applying the boosting procedure we reduce consistently the error

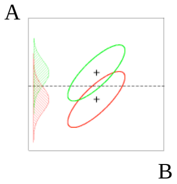


Boosted Decision Tree

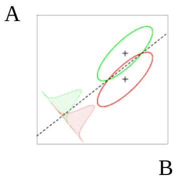
- **Note that we can apply bagging and boosting**
  - In general very CPU-time consuming
- **Possible overtraining**
  - check performance on a validation sample for the maximum number of boosting iterations!
- **Used recently by MiniBooNE Collaboration for particle identification (PID): arXiv:physics/0408124v2 (2004)**
  - First use of this this technique in HEP community
  - They found that the boosting algorithm is 20 to 80% better than that with a standard Artificial Neural Network PID technique
- **Also BaBar Collaboration now implement boosted decision tree for PID**

Linear Discrimination

- **Assuming variables which are linear uncorrelated**
- **Taking the simple case of 2 variables with 2 hypotheses**
  - Green: Signal
  - Red: Background



- **We can find a linear combination to have maximum separation (i.e. small EPE)**

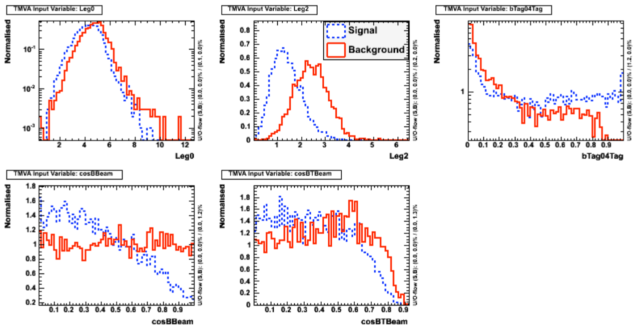


Fisher Discriminant

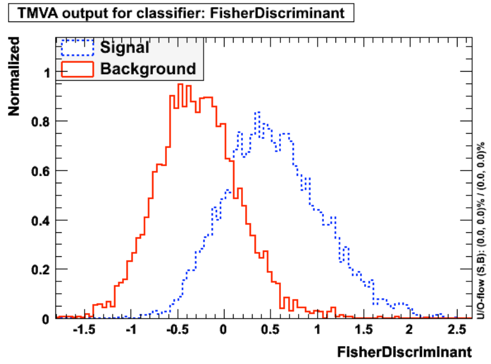
- **Linear discriminant analysis was introduced by Fisher in 1936**
  - very popular tool in analysis of HEP data
- **Event selection is performed in the transformed variable space with zero linear correlations, by distinguishing the mean values of the signal and background distributions**
- **The linear discriminant analysis determines an axis in the (correlated) hyperspace of the input variables such that, when projecting the output hypotheses (signal and background) upon this axis, they are pushed as far as possible away from each other, while events of a same class are confined in a close vicinity**

Fisher Discriminant Example

- **5 Input Variables**



Fisher Discriminant Example



$$\mathcal{F} = (0.405 \cdot L_0 - 0.858 \cdot L_2 - 0.219 \cdot |\cos\theta_{TB}| - 1.95 \cdot |\cos\theta_{BB}| + 0.756 \cdot Tag_{04}) - 0.134$$

Maximum Likelihood Method

- **Consider a variable *x***
  - We know the expression of his Probability Density Function (PDF) $f(x; \theta)$, where θ are unknown parameters
- **Suppose to perform an experiment where the measurement of *x* has been repeated *N* times**
- **The maximum likelihood method is a technique to estimate the value of parameter θ for a finite data sample, maximizing the likelihood function (Maximum Likelihood):**

$$\mathcal{L}(\theta) = \prod_{i=1}^{N} f(x_i; \theta)$$

- **The maximization is done in a numeric way**

Extended Maximum Likelihood

- **If the *N* number of observations in the sample is itself a Poisson random variable with a mean value *n*, we use add the Poissonian term in the likelihood function**

$$\mathcal{L}(n, \theta) = \frac{e^{-n}}{N!} \prod_{i=1}^{N} n f(x_i; \theta).$$

- **This function is called the Extended Likelihood function**
  - We have to maximize this function ==> Extended Maximum Likelihood

Maximum Likelihood Fit

- **In case of several variables *h* and events belonging to different hypotheses, we can write the total PDF for a given event *i* and hypothesis *j* as**

$$\mathcal{P}_j^i = \prod_{l=1}^{h} f_j^l(x_l^i)$$

(assuming uncorrelated variables)

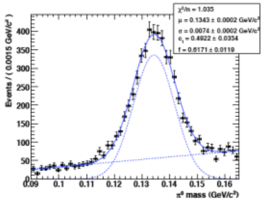
- **We want determine the number of events n_j belonging to each hypothesis *j***

$$\mathcal{L} = \frac{e^{-\sum_{j=1}^{s} n_j}}{N!} \prod_{i=1}^{N} \sum_{j=1}^{s} n_j \mathcal{P}_j^i.$$

- **We maximize this function in term of the n_j free parameters and eventual free parameters of the PDFs**

Maximum Likelihood Fit

- **We can fit a single variables with an analytical function for the PDF, finding the value of the free parameters of the PDF**
- **Unbinned Fits (using RooFit)**
  - Several models are available
  - Note that PDFs must be normalized: RooFit provides several methods for integral calculation

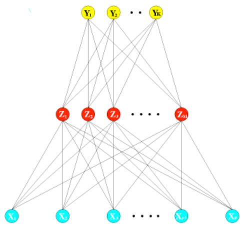


Single Variable Unbinned fit:
Gaussian (signal)
+
Linear polynomial (background)

- **Widely used in HEP BaBar and Belle Collaborations**

Artificial Neural Network

- **The central idea is to extract linear combinations of the input variables as derived features, and then model the target as a nonlinear function of these features**
  - Neural Network is represented by a network diagram



- **The most widely used neural net is called “vanilla”, also called the single hidden layer back-propagation network**
  - We several input variables X, one output Y, and several hidden layers with several nodes (neurons) linked sequentially

Artificial Neural Network

- **Derived features Z_m are created from linear combinations of the inputs, and then the target functions Y_k are modeled as function of linear combinations of the Z_m**

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X),\ m = 1, \dots, M$$

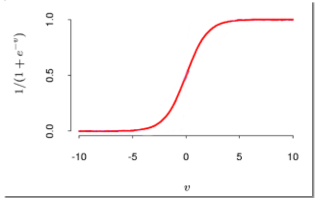
$$f_k(X) = \beta_{0k} + \beta_k^T Z,\ k = 1, \dots, K$$

where $Z = (Z_1, Z_2, \dots, Z_M)$

$T = (T_1, T_2, \dots, T_K)$

- ***f*** **(*x*) can be a different function**
- **The activation function σ(*v*) is usually chose to be the sigmoid function**

$$1/(1+e^{-v})$$



Neural Network Training

- **We should determine the unknown parameters (weights)**

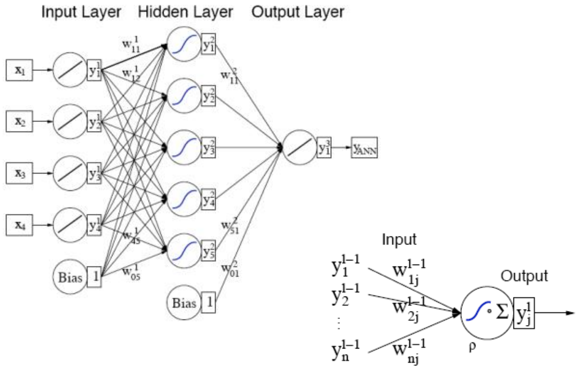
$$\{\alpha_{0m}, \alpha_m;\ m = 1, 2, \dots, M\}\ \ M(p+1)\ \text{weights},$$
$$\{\beta_{0k}, \beta_k;\ k = 1, 2, \dots, K\}\ \ K(M+1)\ \text{weights}.$$

trying to minimize the error function

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2.$$

- **Usually we add a bias input to remove the intercepts α_{0m} and β_{0k}**

Neural Network Training

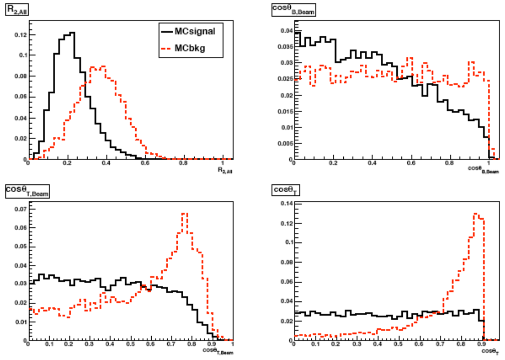


Neural Network Training

- **Weights are calculated using a numerical method, in two stages: forward and backward propagation**
  - In the first stage the weights are fixed and we compute the output of the neural network
  - In the second stage we use the output of the neural network to calculate the new weights using a gradient methods in order to minimize the error function
  - The two steps represent a cycle of the training: each cycle try to reduce the error
  - Training is stopped after a determined number of cycles, in order to avoid overtraining (using validation sample as check)
- **Several minima can exist, depending on the choice of the starting weights: it is important to randomize the initial weights, choosing the solution which gives the lowest error**

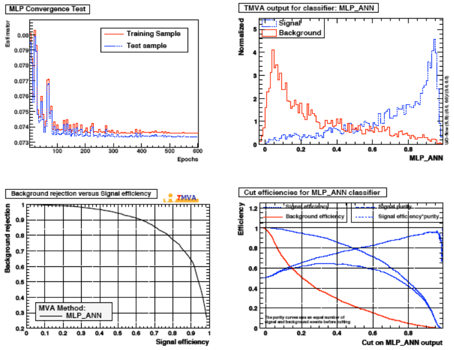
Neural Network Example

- **4 Input Variables**

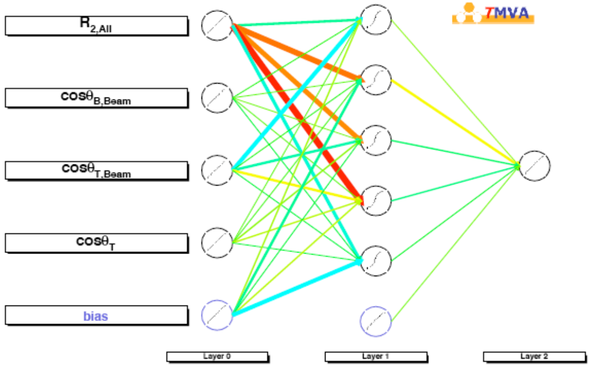


Neural Network Example

- **600 training cycles (epochs), 1 hidden layer with 5 hidden nodes**



Neural Network Example



Neural Networks Conclusion

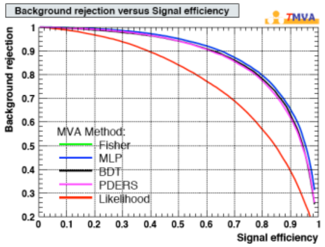
- **Widely used in many application in HEP experiments and data analysis:**
  - Event reconstruction
  - Background suppression
  - Particle Identification
- **Try different configurations with different number of hidden layers and neurons**
  - Chose the best one with the minimum error
- **Use a validation sample to avoid overtraining**
  - Training can be very CPU-time consuming

Which method?

- **In general there is not a priori theorem which says what is the best method for discrimination of events**
- **Depends of your specific problems**
  - Sophisticated methods can be very CPU-time consuming, giving a small improvements in the overall results
  - Easy methods can give fast answer, less accurate, but still valid
- **Note that each method can have a systematic (for example if you use Monte Carlo samples in the training and validation)**
  - Usually these systematic errors are not easily understandable

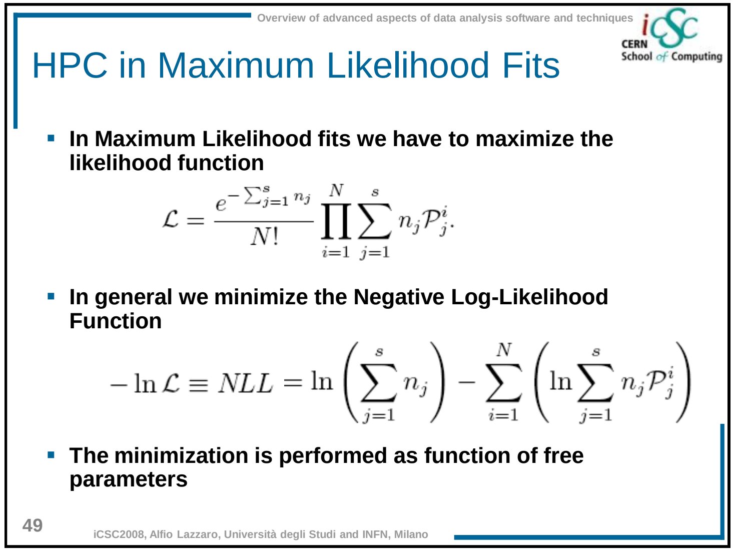
Suggestions

- **Try to have a good compromise between complexity of the models and discrimination power**
  - Try several methods, choose one method with better performance and try to improve his performance using different configurations (number of nodes in a tree, number of hidden layer in a neural network,…)
- **Remember to use validation sample to check your training**
  - In general sophisticated methods can give overtraining performance
  - Use a simple method (like “Cut & Count” analysis) as comparison for your results



Computational Considerations

- **In general all methods are based on optimization problems: find a maximum (for example in case of Statistical Significance Maximization or Maximum Likelihood) or a minimum (Expected Prediction Error) of a function**
- **This is done by numerical algorithms**
  - Most commonly used are based on Gradient Descent Methods, which require the calculation of several derivates of the function
- **This procedure can be very slow, depending on the number of free parameters to be determined, the number of input events, and the complexity of the model**



HPC in Maximum Likelihood Fits

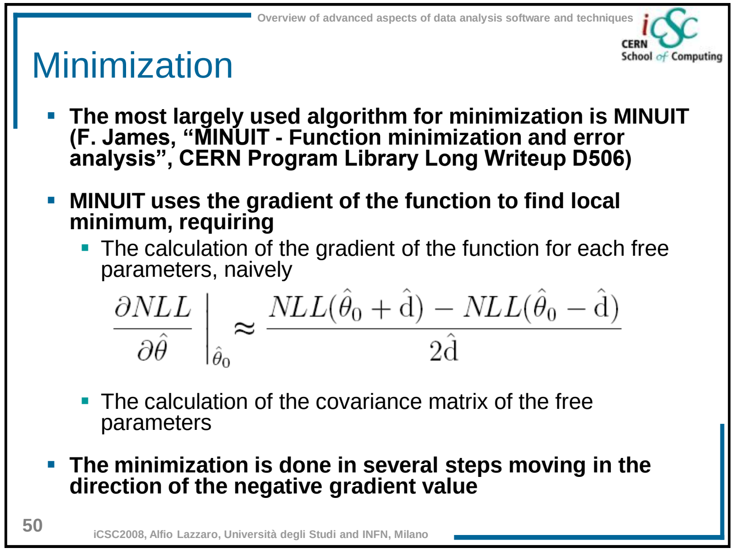
- **In Maximum Likelihood fits we have to maximize the likelihood function**

$$\mathcal{L} = \frac{e^{-\sum_{j=1}^{s} n_j}}{N!} \prod_{i=1}^{N} \sum_{j=1}^{s} n_j \mathcal{P}_j^i.$$

- **In general we minimize the Negative Log-Likelihood Function**

$$-\ln \mathcal{L} \equiv NLL = \ln\left(\sum_{j=1}^{s} n_j\right) - \sum_{i=1}^{N}\left(\ln \sum_{j=1}^{s} n_j \mathcal{P}_j^i\right)$$

- **The minimization is performed as function of free parameters**

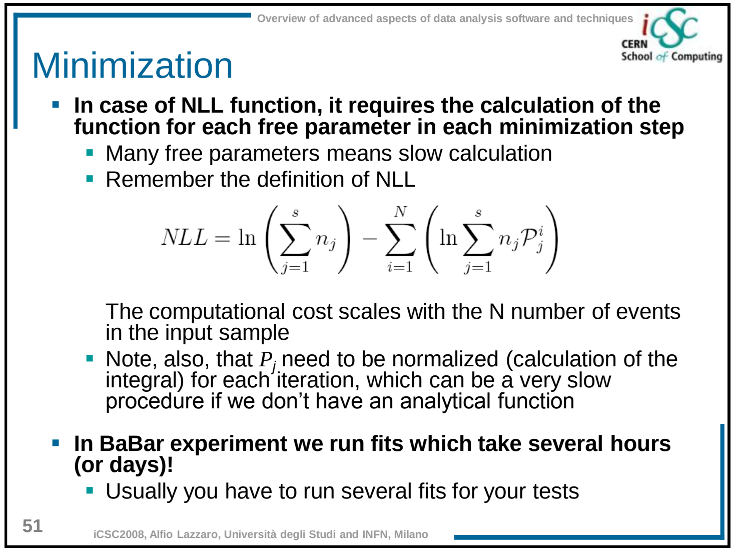


Minimization

- **The most largely used algorithm for minimization is MINUIT (F. James, "MINUIT - Function minimization and error analysis", CERN Program Library Long Writeup D506)**
- **MINUIT uses the gradient of the function to find local minimum, requiring**
  - The calculation of the gradient of the function for each free parameters, naively

$$\left.\frac{\partial NLL}{\partial \hat{\theta}}\right|_{\hat{\theta}_0} \approx \frac{NLL(\hat{\theta}_0 + \hat{\mathrm{d}}) - NLL(\hat{\theta}_0 - \hat{\mathrm{d}})}{2\hat{\mathrm{d}}}$$

  - The calculation of the covariance matrix of the free parameters
- **The minimization is done in several steps moving in the direction of the negative gradient value**

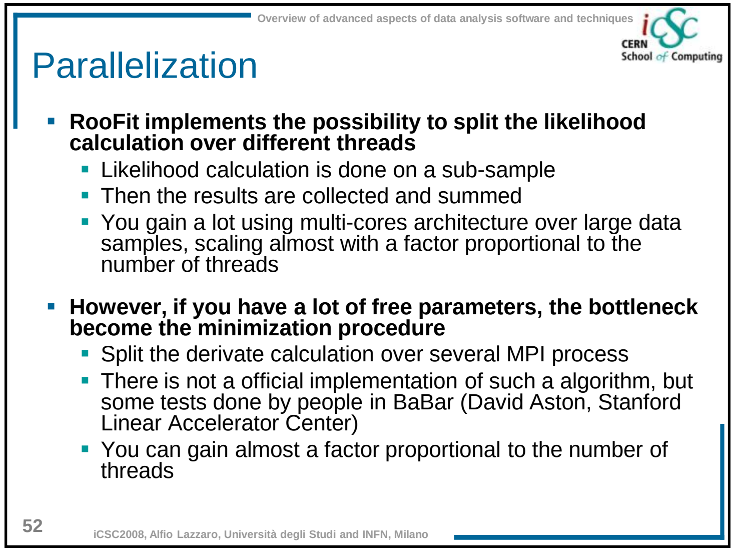


Minimization

- **In case of NLL function, it requires the calculation of the function for each free parameter in each minimization step**
  - Many free parameters means slow calculation
  - Remember the definition of NLL

$$NLL = \ln\left(\sum_{j=1}^{s} n_j\right) - \sum_{i=1}^{N}\left(\ln \sum_{j=1}^{s} n_j \mathcal{P}_j^i\right)$$

  The computational cost scales with the N number of events in the input sample
  - Note, also, that P_j need to be normalized (calculation of the integral) for each iteration, which can be a very slow procedure if we don't have an analytical function
- **In BaBar experiment we run fits which take several hours (or days)!**
  - Usually you have to run several fits for your tests

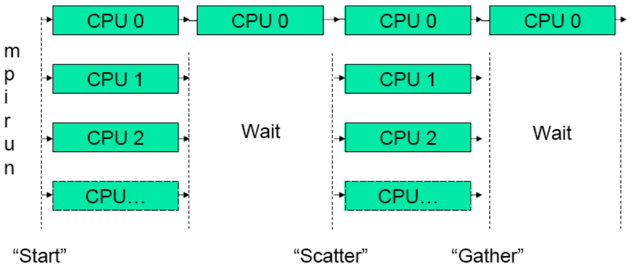


Parallelization

- **RooFit implements the possibility to split the likelihood calculation over different threads**
  - Likelihood calculation is done on a sub-sample
  - Then the results are collected and summed
  - You gain a lot using multi-cores architecture over large data samples, scaling almost with a factor proportional to the number of threads
- **However, if you have a lot of free parameters, the bottleneck become the minimization procedure**
  - Split the derivate calculation over several MPI process
  - There is not a official implementation of such a algorithm, but some tests done by people in BaBar (David Aston, Stanford Linear Accelerator Center)
  - You can gain almost a factor proportional to the number of threads

Parallelization

“Scatter-Gather” running



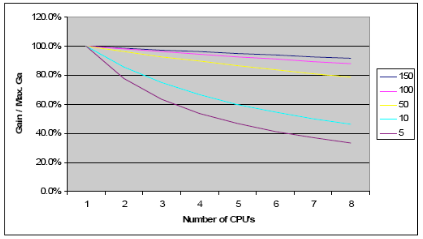
From Brain Meadows talk at RooFit Mini Workshop @ SLAC (December 2007): http://www.slac.stanford.edu/BFROOT/www/doc/Workshops/2007/BaBar_RooFit/Agenda.html

Parallelization

It works well in case of large number of parameters

Gain ~ NCPU*(NPAR + 2) / (NPAR + 2*NCPU)

Max. Gain = NCPU



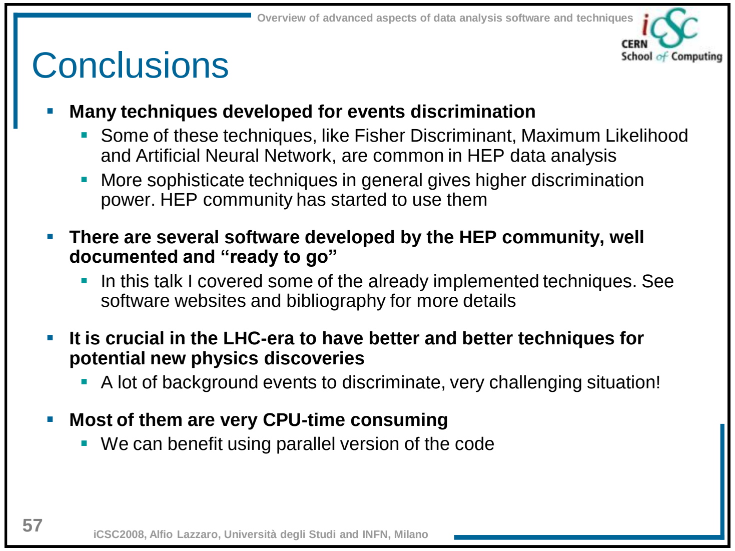
From Brain Meadows talk at RooFit Mini Workshop @ SLAC (December 2007): http://www.slac.stanford.edu/BFROOT/www/doc/Workshops/2007/BaBar_RooFit/Agenda.html

Parallelization

- **Hybrid of the likelihood calculation and minimization process are not implemented yet ==> higher gain in case of multi-cores/MPI case**
- **Anyway, in some case the bottleneck is the integral calculation (in Dalitz plot analysis we have integral in several variables, which are very slow to compute)**
  - There is not parallel implementation of the normalization integral calculation
  - My current work
- **I think it is time to have faster Maximum Likelihood fits!**
  - Welcome HPC techniques!

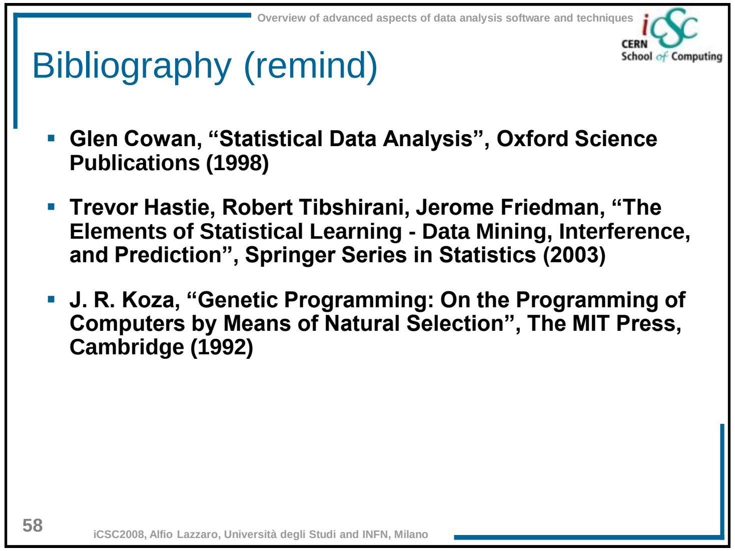
Other example of HPC use

- **Selection of events applying different cuts: PROOF project, implemented in ROOT:**
  http://root.cern.ch/twiki/bin/view/ROOT/PROOF
  - allowing transparent analysis of large sets of ROOT files in parallel on compute clusters or multi-core computers, splitting the data sample
- **Bagging and Boosting can be very CPU-time consuming**
  - Several variables on several events
  - Trees are almost independent, they can split in a parallel architecture
- **Neural Networks can implemented on parallel architectures**
- **In HEP community there is not mention of Trees and Neural Networks (as far as I know) using HPC**
  - Please let me know if you know something that I missed



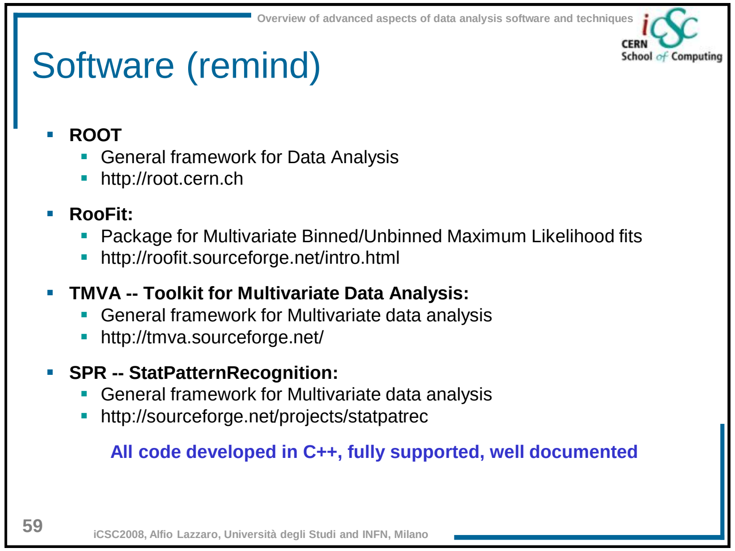
Conclusions

- **Many techniques developed for events discrimination**
  - Some of these techniques, like Fisher Discriminant, Maximum Likelihood and Artificial Neural Network, are common in HEP data analysis
  - More sophisticate techniques in general gives higher discrimination power. HEP community has started to use them
- **There are several software developed by the HEP community, well documented and "ready to go"**
  - In this talk I covered some of the already implemented techniques. See software websites and bibliography for more details
- **It is crucial in the LHC-era to have better and better techniques for potential new physics discoveries**
  - A lot of background events to discriminate, very challenging situation!
- **Most of them are very CPU-time consuming**
  - We can benefit using parallel version of the code



Bibliography (remind)

- **Glen Cowan, "Statistical Data Analysis", Oxford Science Publications (1998)**
- **Trevor Hastie, Robert Tibshirani, Jerome Friedman, "The Elements of Statistical Learning - Data Mining, Interference, and Prediction", Springer Series in Statistics (2003)**
- **J. R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection", The MIT Press, Cambridge (1992)**



Software (remind)

- **ROOT**
  - General framework for Data Analysis
  - http://root.cern.ch
- **RooFit:**
  - Package for Multivariate Binned/Unbinned Maximum Likelihood fits
  - http://roofit.sourceforge.net/intro.html
- **TMVA -- Toolkit for Multivariate Data Analysis:**
  - General framework for Multivariate data analysis
  - http://tmva.sourceforge.net/
- **SPR -- StatPatternRecognition:**
  - General framework for Multivariate data analysis
  - http://sourceforge.net/projects/statpatrec

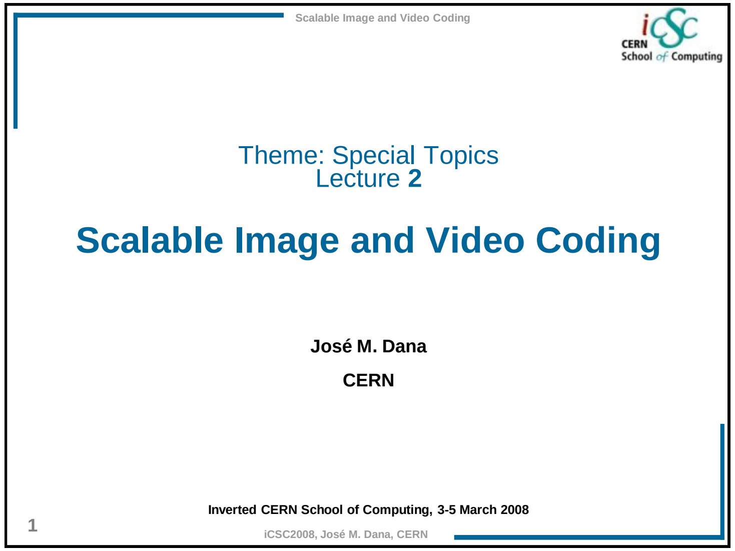
All code developed in C++, fully supported, well documented

iCSC
CERN
School of Computing

LECTURE 2

Scalable Image and Video coding

		Wednesday 5 march 2008	
10:30 12:00	Lecture2	The aim of this lecture is to describe the basis of image and video coding and compression, with a special emphasis on the latest developments. We will see how to encode and compress this particular type of data using lossy algorithms that take advantage of the limitations of the human visual system. We will focus on scalable image and video coding, which is a cutting-edge area of research, an area were few fully recognized standards have emerged yet. Sometimes, specialized developers need to design systems which require an image or video (de)coder. Understanding the internals of some coding systems may help them in to select the most appropriate approach (streaming systems, pattern recognition systems, etc.) and algorithm (JPEG, JPEG2000, MPEG-2, MPEG-4, WMV, etc.). We will present techniques used in well-known algorithms and the audience will have the opportunity to learn the fundamentals through practical examples. **Audience** The lecture targets all participants with interest in image, video coding and compression. **Pre-requisite** No pre-requisite is necessary.	**Jose Dana Perez**



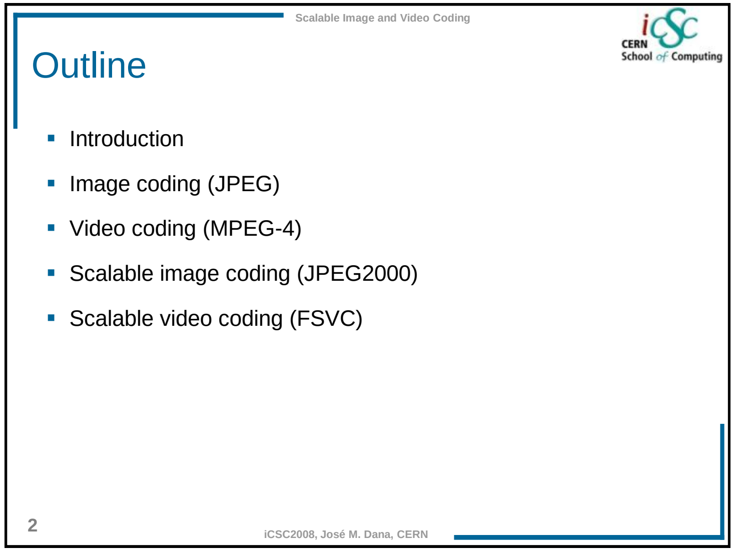
Theme: Special Topics
Lecture **2**

Scalable Image and Video Coding

José M. Dana

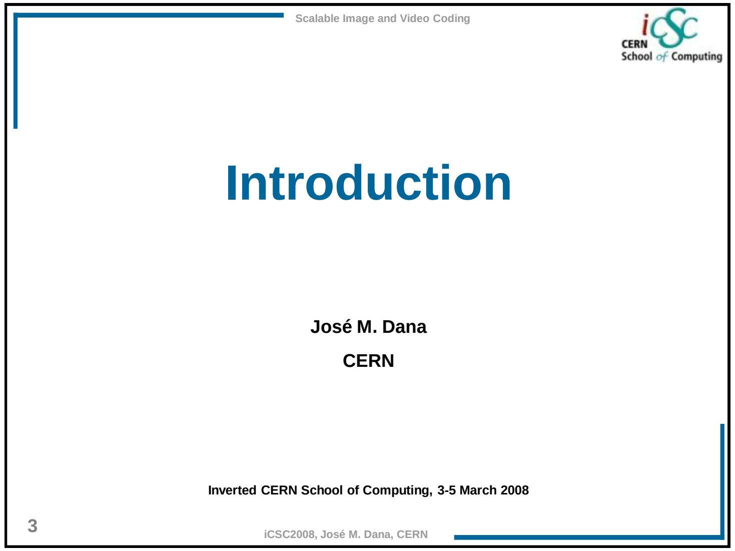
CERN

Inverted CERN School of Computing, 3-5 March 2008



Outline

- Introduction
- Image coding (JPEG)
- Video coding (MPEG-4)
- Scalable image coding (JPEG2000)
- Scalable video coding (FSVC)

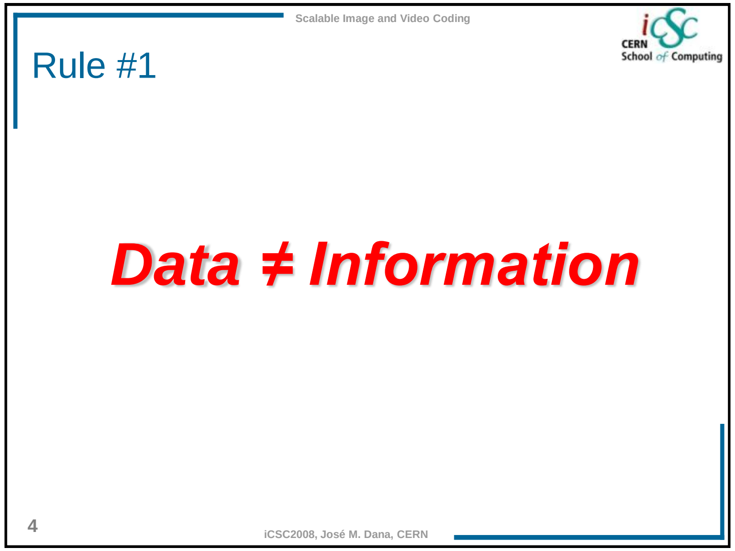


Introduction

José M. Dana

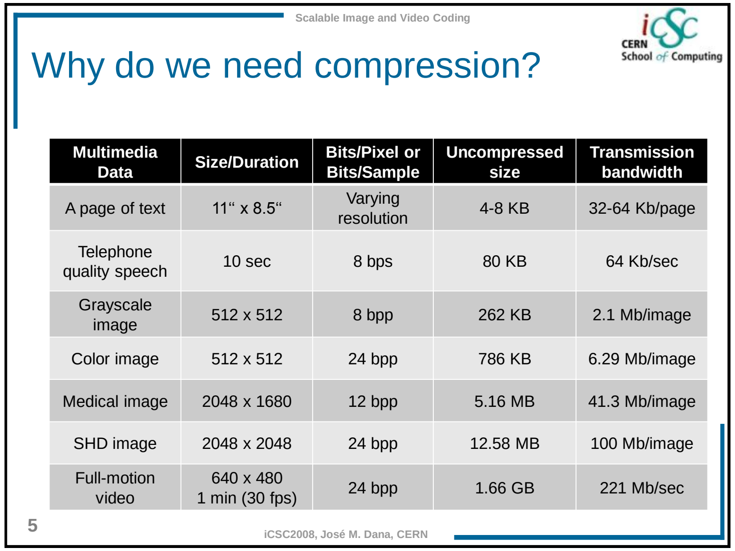
CERN

Inverted CERN School of Computing, 3-5 March 2008



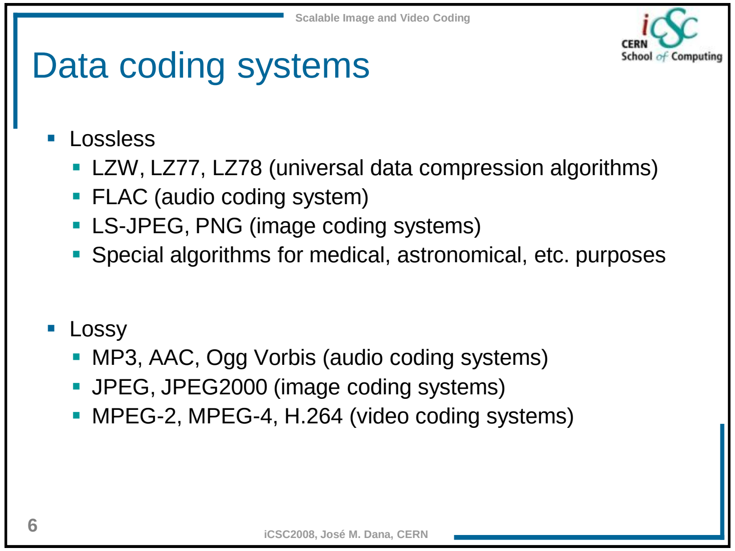
Rule #1

Data ≠ Information



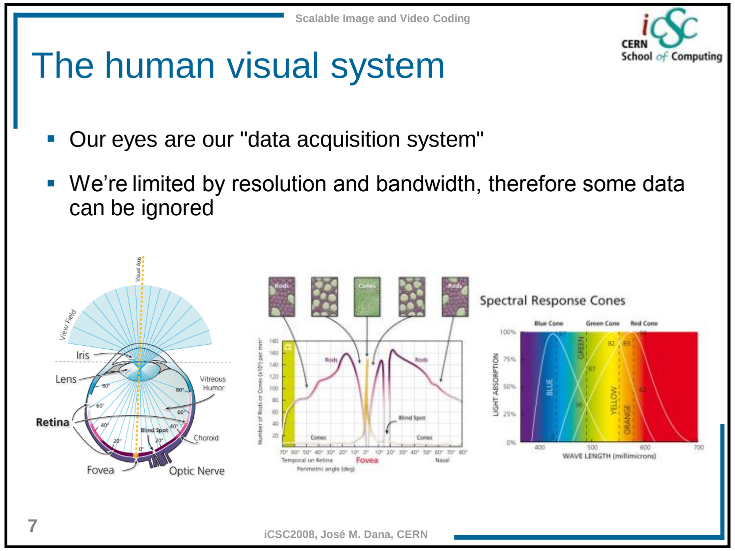
Why do we need compression?

Multimedia Data	Size/Duration	Bits/Pixel or Bits/Sample	Uncompressed size	Transmission bandwidth
A page of text	11“ x 8.5“	Varying resolution	4-8 KB	32-64 Kb/page
Telephone quality speech	10 sec	8 bps	80 KB	64 Kb/sec
Grayscale image	512 x 512	8 bpp	262 KB	2.1 Mb/image
Color image	512 x 512	24 bpp	786 KB	6.29 Mb/image
Medical image	2048 x 1680	12 bpp	5.16 MB	41.3 Mb/image
SHD image	2048 x 2048	24 bpp	12.58 MB	100 Mb/image
Full-motion video	640 x 480 1 min (30 fps)	24 bpp	1.66 GB	221 Mb/sec



Data coding systems

- Lossless
  - LZW, LZ77, LZ78 (universal data compression algorithms)
  - FLAC (audio coding system)
  - LS-JPEG, PNG (image coding systems)
  - Special algorithms for medical, astronomical, etc. purposes
- Lossy
  - MP3, AAC, Ogg Vorbis (audio coding systems)
  - JPEG, JPEG2000 (image coding systems)
  - MPEG-2, MPEG-4, H.264 (video coding systems)



The human visual system

- Our eyes are our "data acquisition system"
- We’re limited by resolution and bandwidth, therefore some data can be ignored

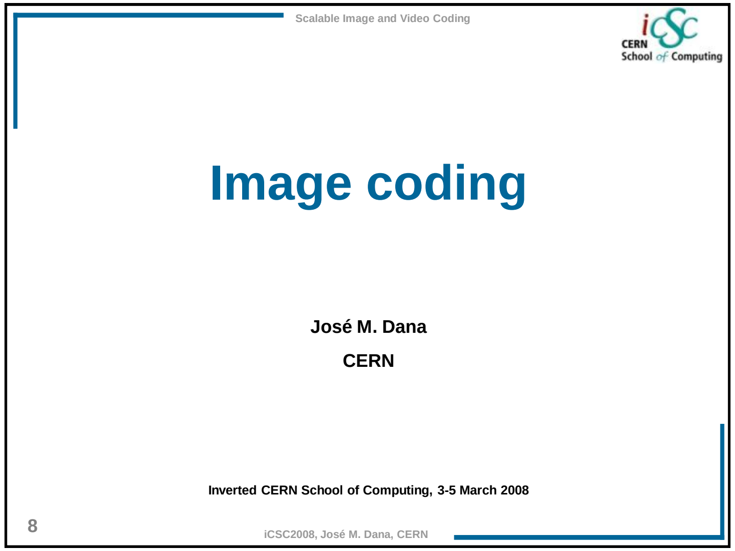


Image coding

José M. Dana

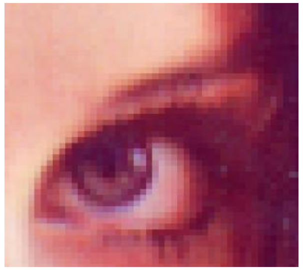
CERN

Inverted CERN School of Computing, 3-5 March 2008

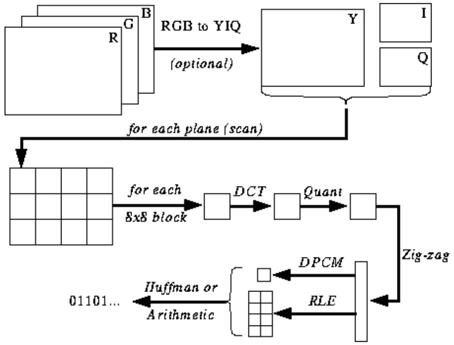
Some definitions

- **Digital Image:** *An image stored in binary form and divided into a matrix of pixels, each consists of one or more bits of information that represent either the brightness, or brightness and color, of the image at that point.*
- **Pixel:** *A contraction of the words picture element. The smallest unit of information in an image or raster map. Referred to as a cell in an image or grid.*

Spatial redundancy



JPEG coding

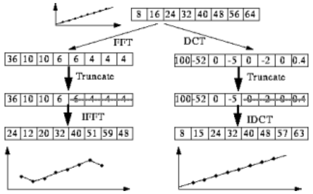


JPEG - Colorspaces

- Valid color spaces
  - Grayscale
  - YIQ (Y=Luminance; I,Q=Chrominance)
  - YCbCr (Y=Luminance, Cb=Blue/Yellow axis, Cr=Red/Green axis)
  - CMYK (Cyan-Magenta-Yellow-Key)
- Not valid color spaces (transformation needed)
  - RGB (Red-Green-Blue)
  - RGBA (Red-Green-Blue-Alpha)
  - YUV (Y=Luminance; U,V=Chrominance)
  - Etc.

Energy compaction capability

- When applying the DCT to a signal, a higher ratio of the energy is concentrated in a small number of coefficients relative to the FFT or other similar transforms



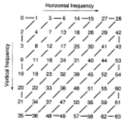
FFT vs. DCT

JPEG - Quantization and "zig-zag"

$$\begin{bmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{bmatrix} \xrightarrow{\text{DCT}} \begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & 1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & 1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} = \begin{bmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

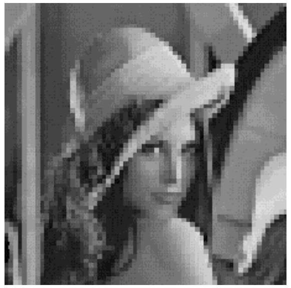
$$\begin{bmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \longrightarrow$$



→ DPCM/RLE
&
HUFFMAN/ARITMETIC

Block artifacts

Original image



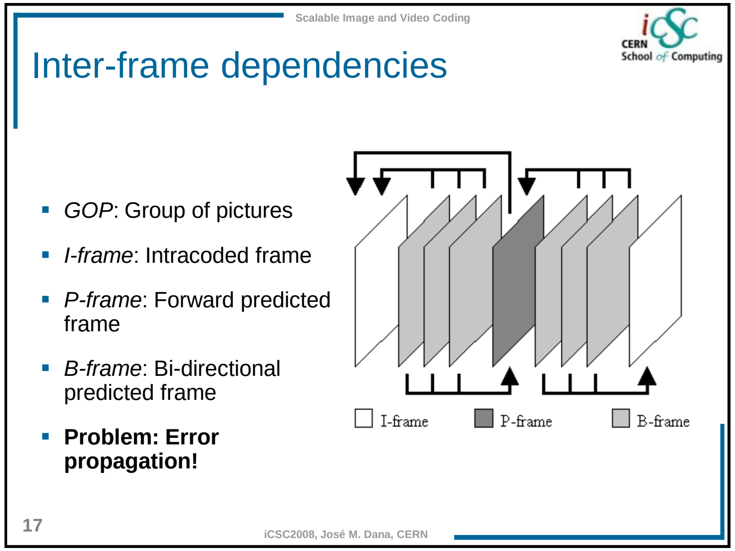
DC component

Video coding

José M. Dana

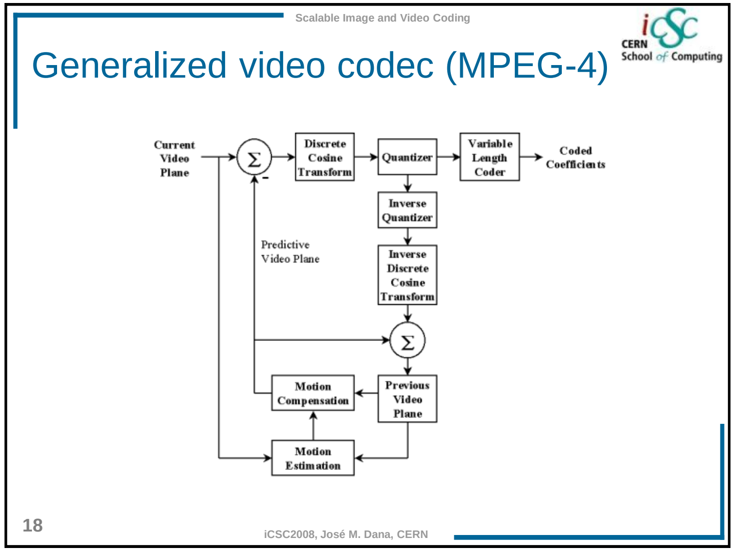
CERN

Inverted CERN School of Computing, 3-5 March 2008

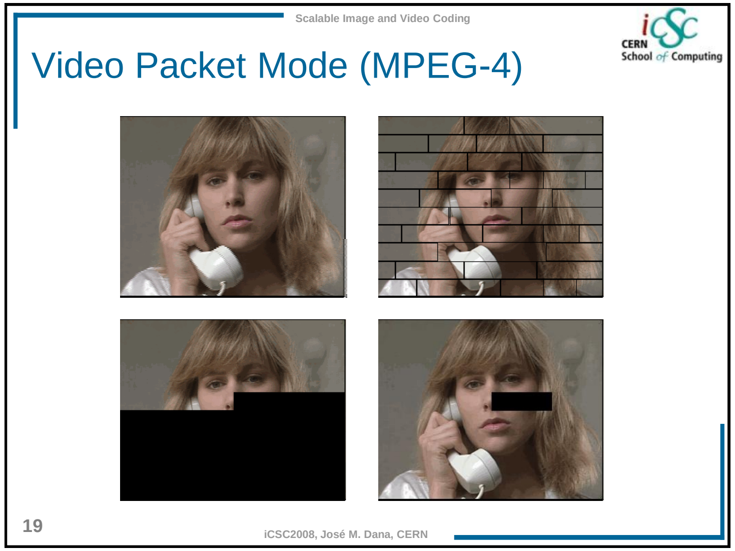


Inter-frame dependencies

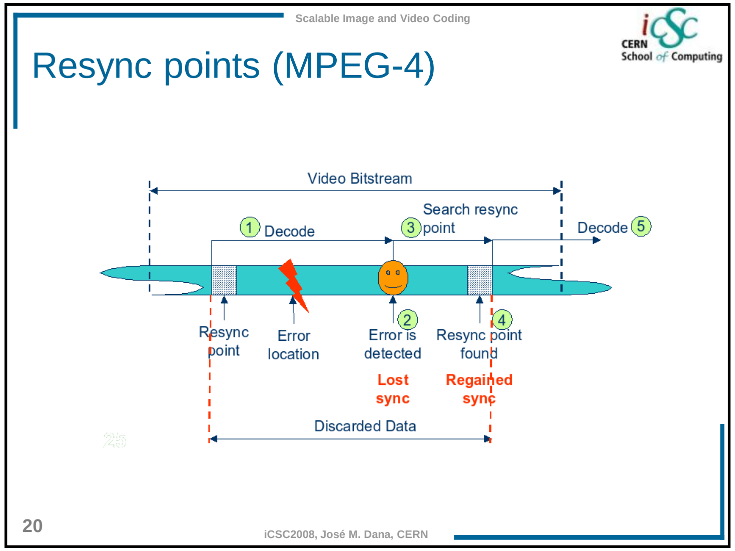
- *GOP*: Group of pictures
- *I-frame*: Intracoded frame
- *P-frame*: Forward predicted frame
- *B-frame*: Bi-directional predicted frame
- **Problem: Error propagation!**



Generalized video codec (MPEG-4)

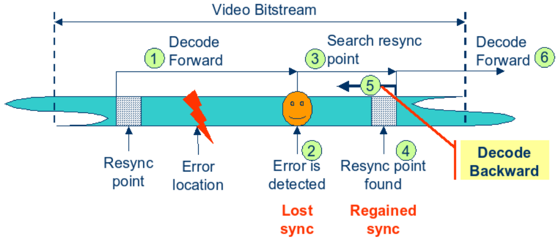


Video Packet Mode (MPEG-4)



Resync points (MPEG-4)

Reversible variable length codes (MPEG-4)



Scalable image coding

José M. Dana

CERN

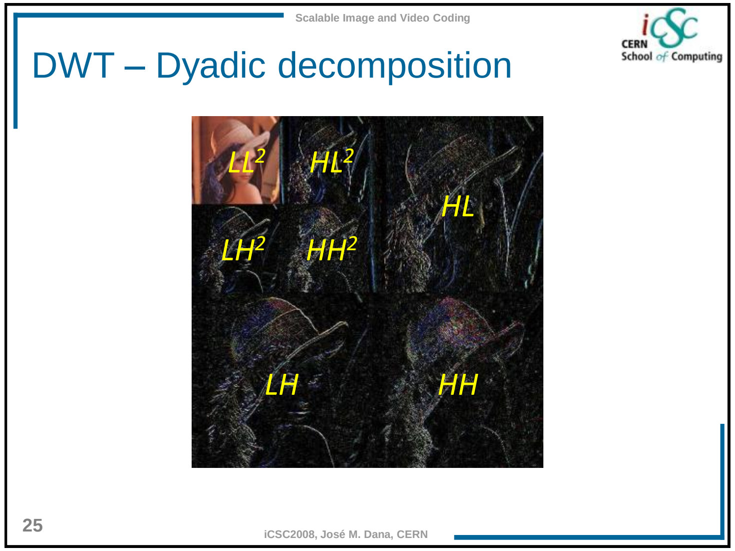
Inverted CERN School of Computing, 3-5 March 2008

What does scalability mean?

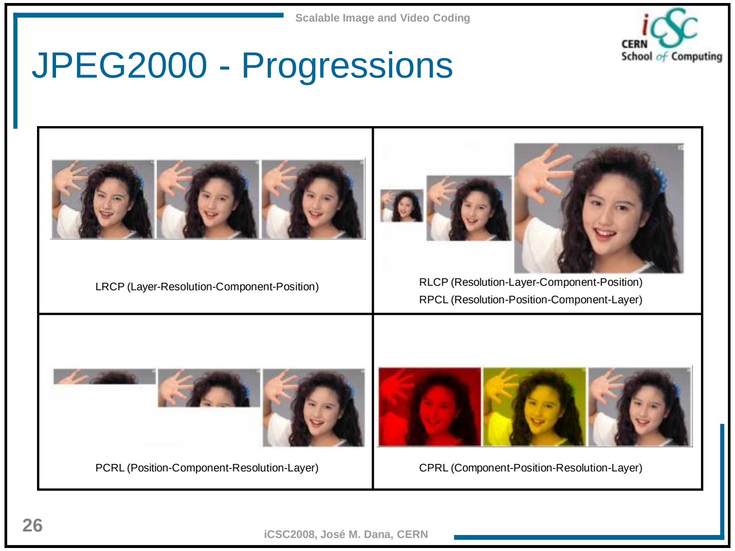
- *Encode Once, Display/Stream Anywhere*
- Current digital video applications require at least three types of scalability features:
  - Quality scalability
  - Spatial resolution scalability
  - Temporal (frame rate) scalability

The Discrete Wavelet Transform

- The DCT previously carries out a division into squared blocks (8x8 pixels in JPEG) while the 2D-DWT works in its totality
- The decomposition into subbands gives a higher flexibility in terms of scalability in resolution and distortion
- The DWT returns a multiresolution representation in a joint spatial-spectral domain
- Better error resilience

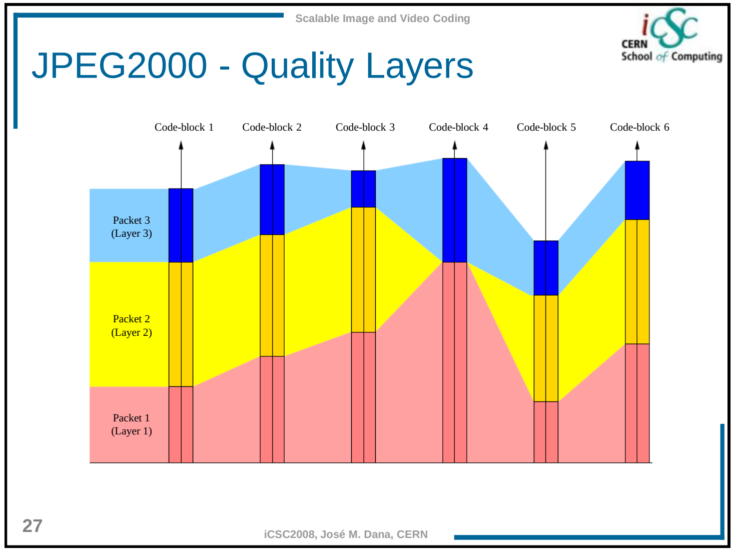


DWT – Dyadic decomposition

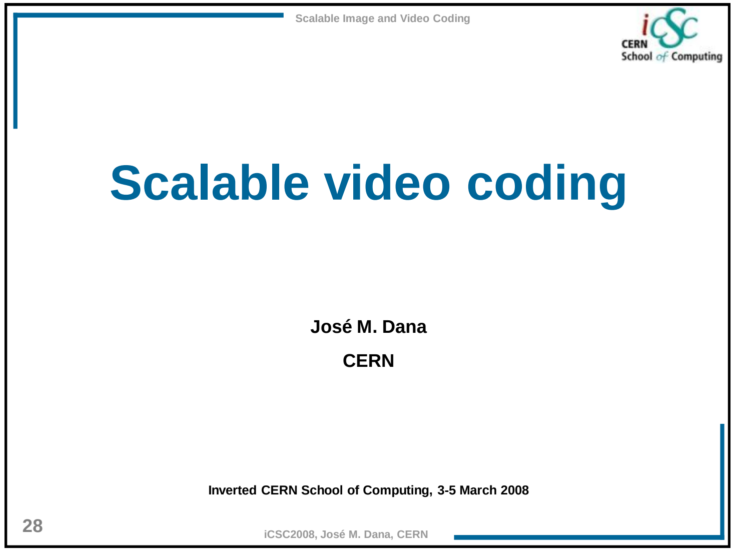


JPEG2000 - Progressions

LRCP (Layer-Resolution-Component-Position)	RLCP (Resolution-Layer-Component-Position) RPCL (Resolution-Position-Component-Layer)
PCRL (Position-Component-Resolution-Layer)	CPRL (Component-Position-Resolution-Layer)



JPEG2000 - Quality Layers



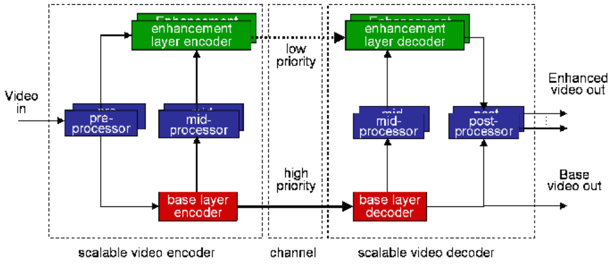
Scalable video coding

José M. Dana

CERN

Inverted CERN School of Computing, 3-5 March 2008

Generalized scalable video codec

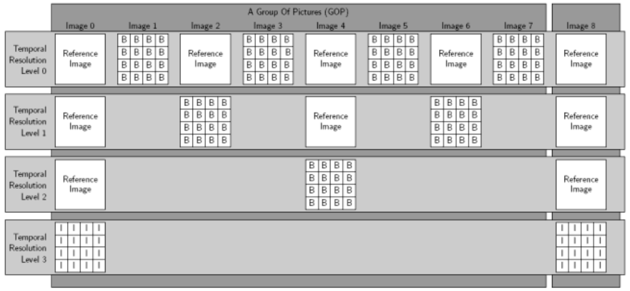


- **Problem #1: The base layer**
- **Problem #2: Drift**

Problem #2: Drift

- In video coding, **drift error** refers to the **continuous decrease** (in picture quality) when a group of motion-compensated interframe pictures have been decoded using frames of reference that are different from the ones used during the encoding step (motion vector field mismatch)
- Scalable video coders have traditionally avoided using enhancement layer information to predict the base layer, so as to avoid so-called “drift”
- As a result, they are less efficient than a one-layer coder

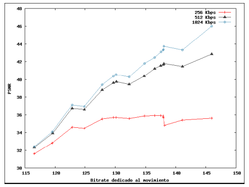
Temporal scalability (FSVC)


A Group Of Pictures (GOP)
Image 0
Image 1
Image 2
Image 3
Image 4
Image 5
Image 6
Image 7
Image 8
Temporal Resolution Level 0
Temporal Resolution Level 1
Temporal Resolution Level 2
Temporal Resolution Level 3
Reference Image
B
I

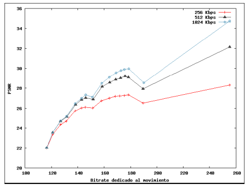
Motion information scalability

- We have been speaking about scalable spatial information but… what about scalable motion information?
- Redundancy can be also found in motion vector fields but it is much more "special" (zoom, accelerated motion, etc.)
- **Problem #1 (again!): The base layer**

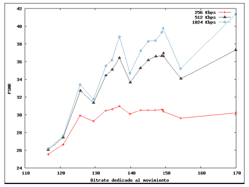
The importance of motion information



Akiyo



Foreman



Container

That's all folks!