Parallel Computer Architecture III

Stefan Lang

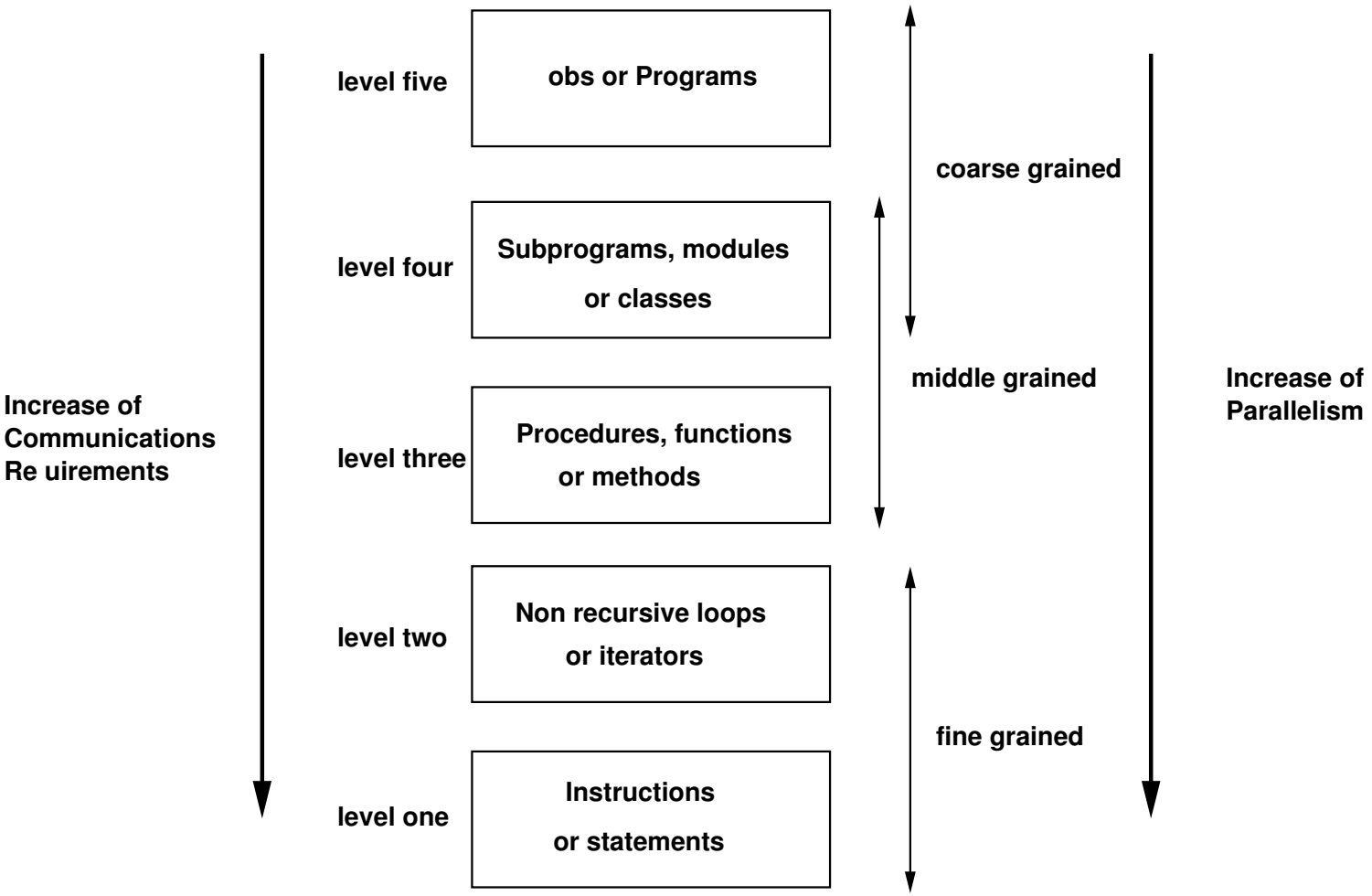
Interdisciplinary Center for Scientific Computing (IWR)
University of Heidelberg
INF 368, Room 532
D-69120 Heidelberg
phone: 06221/54-8264
email: Stefan.Lang@iwr.uni-heidelberg.de

WS 14/15

Parallel Computer Architecture III

- Parallelism and Granularity
- Graphic cards
- I/O
- Detailed study Hypertransport Protocol

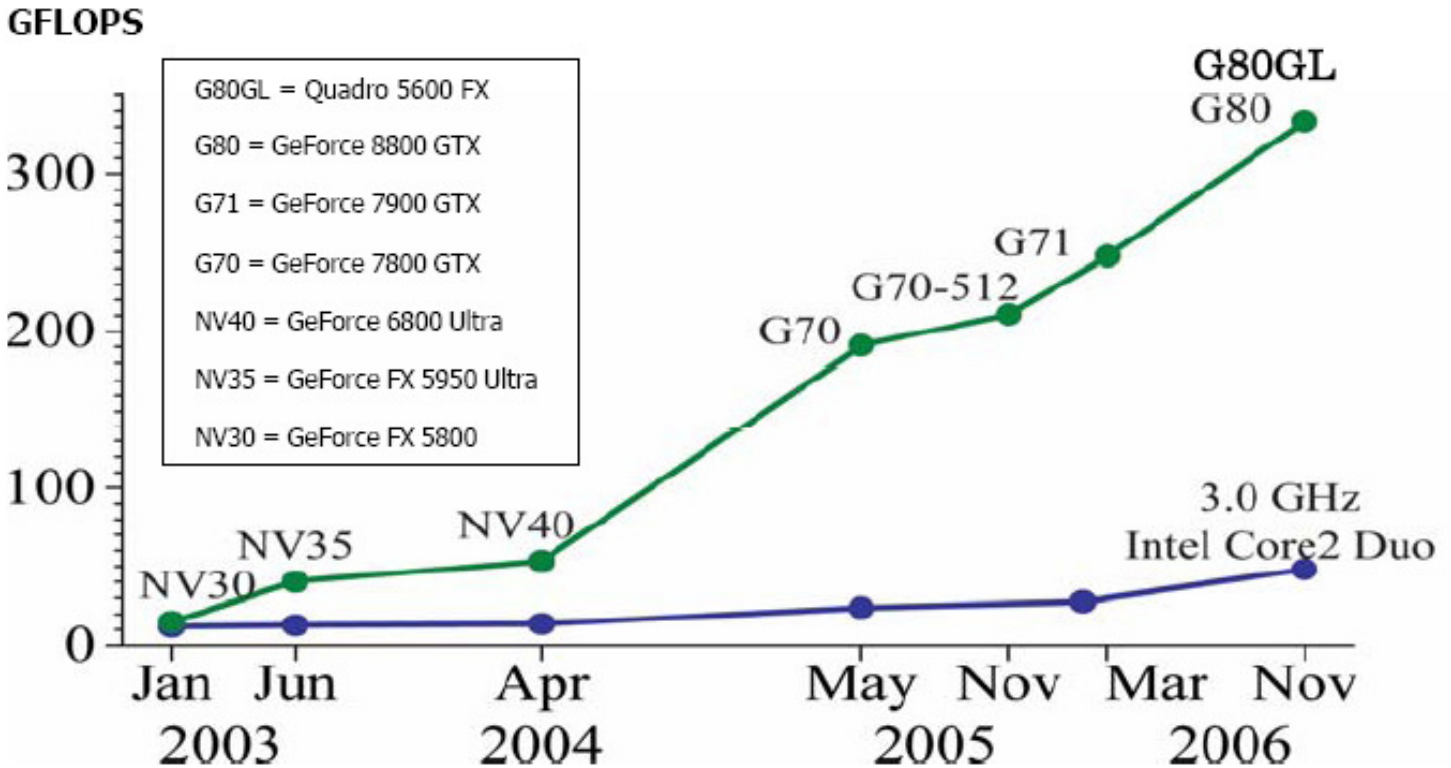
Parallelism and Granularity



Graphics Cards

- GPU = Graphics Processing Unit
- CUDA = Compute Unified Device Architecture
  - Toolkit by NVIDIA for direct GPU Programming
  - Programming of a GPU without graphical API
  - GPGPU
- compared to CPUs strongly increased computing performance and storage bandwidth
- GPUs are cheap and broadly established

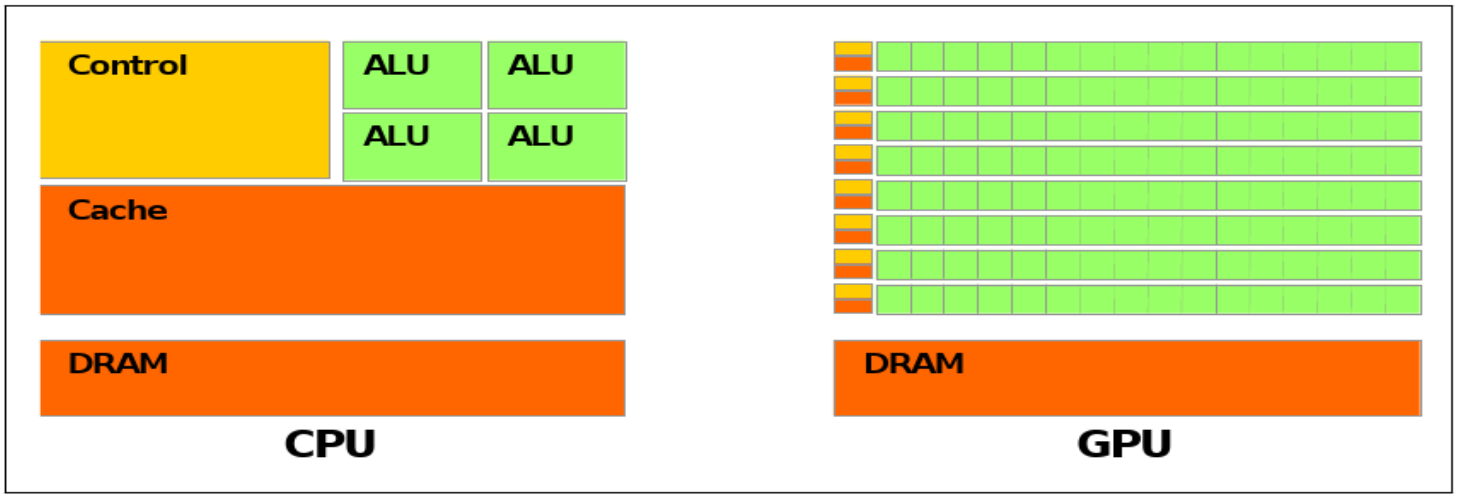
Computing Performance: CPU vs. GPU



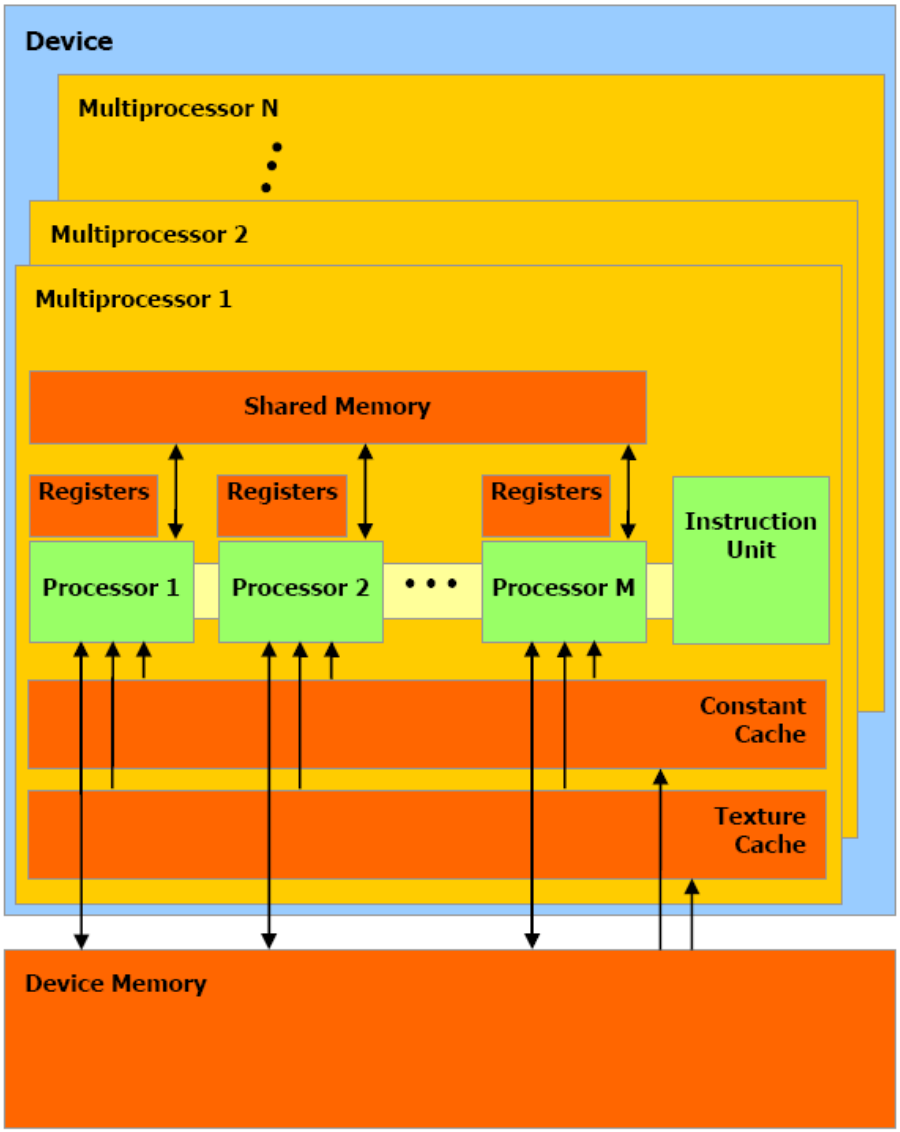
Graphics Cards: Hardware Specification

GeForce GTX 285	
Fab (nm)	55
Transistors (million)	1400
Memory (MB)	1024
Multiprocessors	30
Streaming Processors	240
Shader Clock (MHz)	1476
Memory Bandwidth (GB/s)	159
Memory Bus width (bit)	512
SP GFLOPs (MADD + MUL)	1063
DP GFLOPs (MADD)	89
TDP (Watt)	183
Price (EUR)	~350

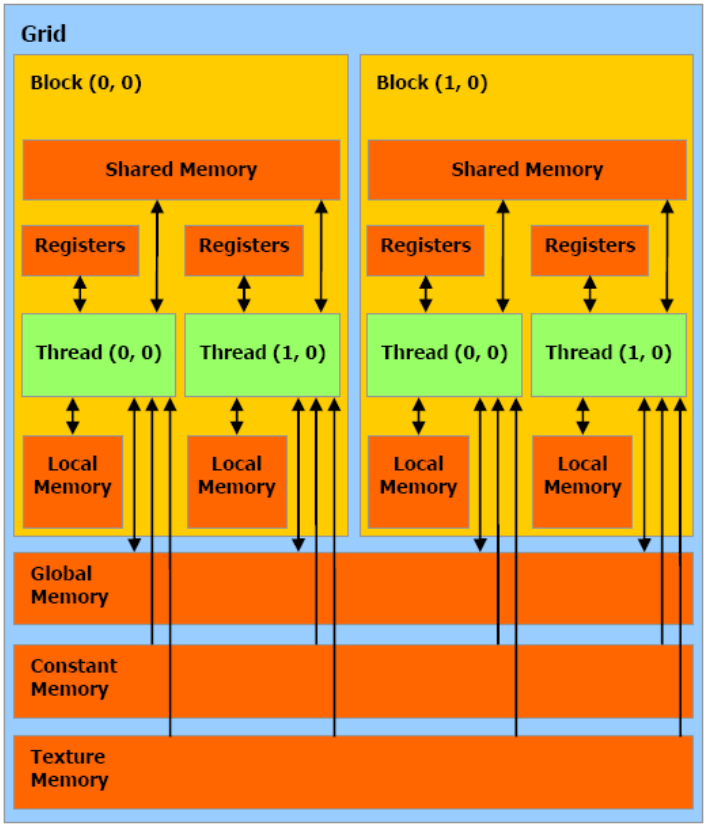
Chip Architecture: CPU vs. GPU



Graphics Cards: Hardware Design

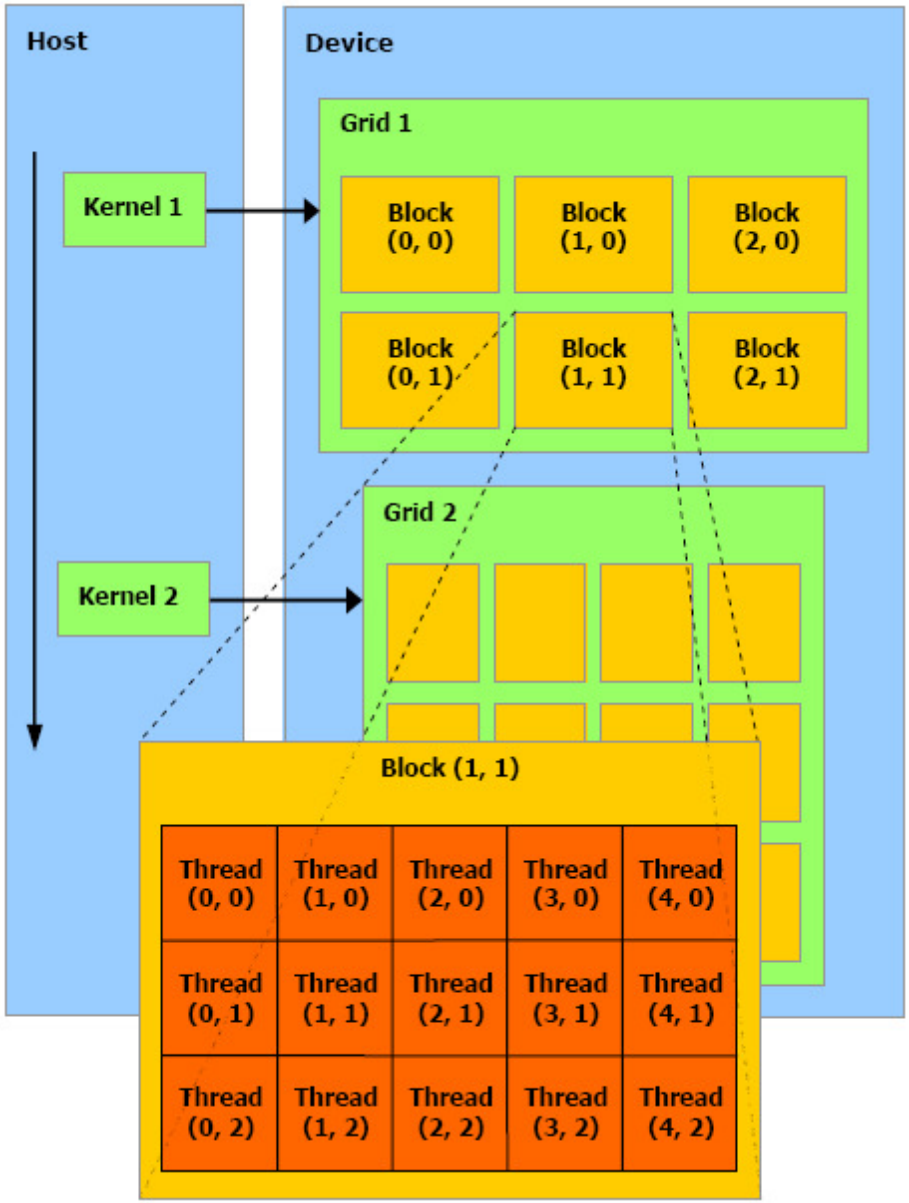


Graphics Cards: Memory Design

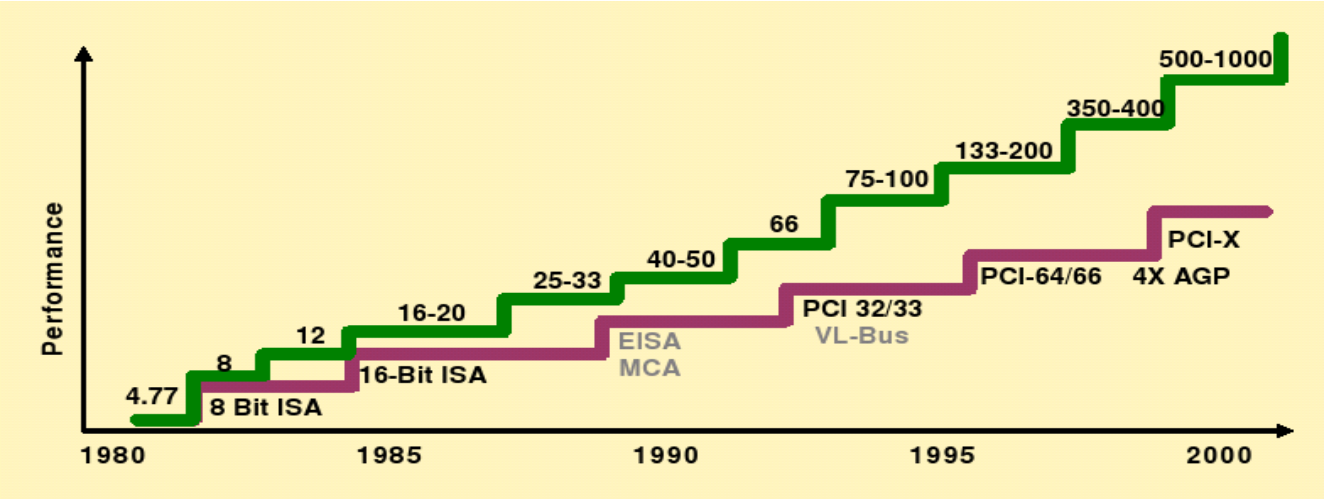


- 8192 registers (32-bit), in total 32KB per multiprocessor
- 16KB of fast shared memory per multiprocessor
- Large global memory (hundreds of MBs, e.g. 512MB-6GB)
- Global memory uncached, latency time 400-600 clock cycles
- Local memory is part of global memory
- Read-only constant memory
- Read-only texture memory
- Registers and shared memory are shared between blocks, that are executed on a sinlge multiprocessor
- Global memory, constant and texture memory are accessible by the CPU

Graphics Card: Programming Modell



Performance Development concerning I/O

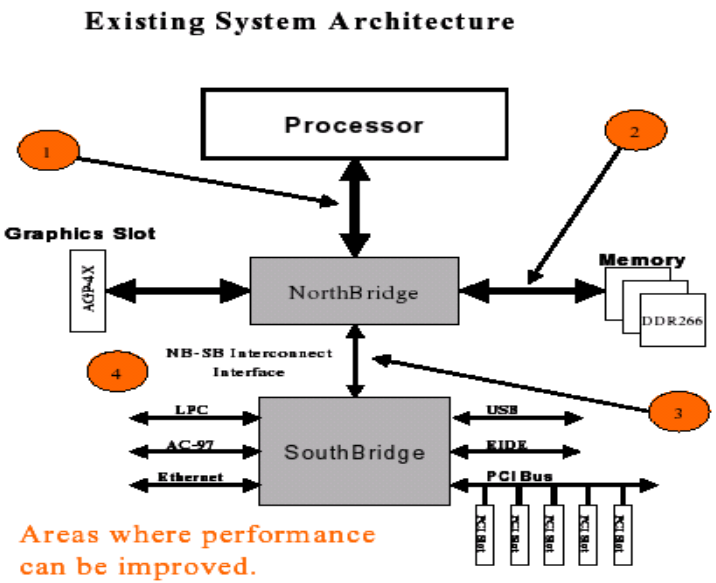


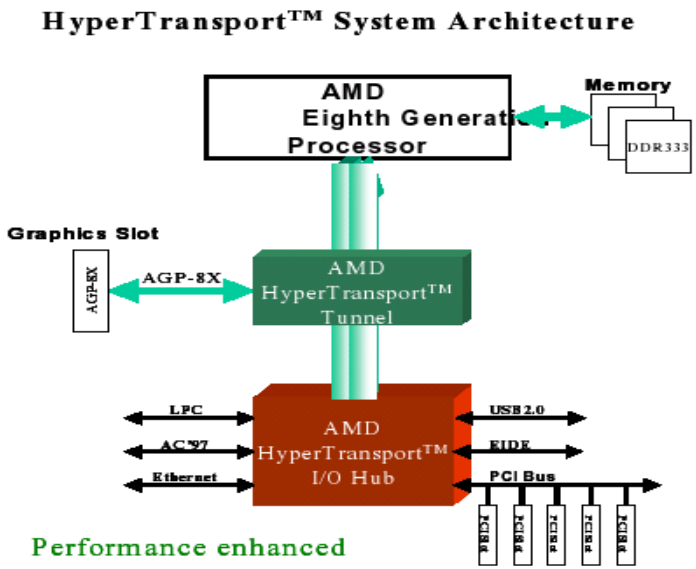
- Micro processor performance doubles about each 18 months
- Performance of the I/O architecture doubles in about 36 months
- Server and workstations necessitate different high speed buses (PCI-X, PCI-E, AGP, USB, SATA)

→ High complexity and needless diversity with limited performance

- Requirements that need bandwidth increase: High-resolution 3D graphics and video (CPU – graphics processor), interprocessor (CPU – CPU), high performance networks e.g. Gigabit Ethernet and Infiniband (CPU – I/O)

Bottlenecks in I/O Section





Possible bottlenecks are:

- Front-Side Bus
- Memory interface
- Chip-to-Chip connection
- I/O to other bus systems

Standardization of I/O Requirements

Development of Hypertransport (HT) I/O connection architecture (since 1997)

- HyperTransport is an In-The-Box solution (intraconnect technology)
- Complementary technique to network protocols like Infiniband and 10 Gb/ 100 Gb ethernet as Box-to-box solutions (interconnect technology)
- HT is open industry standard, no product (no license fees)
- Further development and standardization by consortium of industry partners, like AMD, Nvidia, IBM, Apple
- Former code name: Lightning Data Transfer (LDT)

Functional Overview

Feature/Function	HyperTransport Technology
Bus Type	Dual, unidirectional, point-to-point links
Link Width	2, 4, 8, 16, or 32 bits
Protocol	Packet-based, with all packets multiples of four bytes (32 bits). Packet types include Request, Response, and Broadcast, any of which can include commands, addresses, or data.
Bandwidth (Each Direction)	100 to 6400 Mbytes/s
Data Signaling Speeds	400 MHz to 1.6 GHz
Operating Frequencies	400, 600, 800, 1000, 1200, and 1600 Megatransfers/second
Duplex	Full
Max Packet Payload or Burst Length	64-byte packet
Power Management	ACPI-compatible
Signaling	1.2-V Low-Voltage Differential Signaling (LVDS) with a 100-ohm differential impedance
Multiprocessing Support	Yes
Environment	Inside the box
Memory model	Coherent and noncoherent

Design Goals in the Design of HyperTransport

- Improvement of system performance
  - Higher I/O bandwidth
  - Avoidance of bottlenecks by slower devices in critical information pathes
  - lower count of system buses
  - lower response time (low latency)
  - reduced energy consumption
- Simplification of system design
  - Unified protocol for In-box connections
  - Usage of small pin counts for high packaging density and to ensure low costs
- Higher I/O flexibility
  - Modular bridge architecture
  - Different bandwidth in upstream/downstream direction
- Compatibility with existing systems
  - Supplement for standardized, external buses
  - Minor implications on existing operating systems and drivers
- Extendability for system network architectures (SNA busses)
- High scalability in multiprocessor systems

Flexible I/O Architecture

Hypertransport architecture is organized in 5 layers
Structure is adopted from **Open-Sytem-Interconnection** (OSI) reference model

- **Bit transmission layer:** Physical and electrical properties of data, control, clockwires
- **Data connection layer:** Initialisation and configuration of connections, periodic cyclic redundance (CRC), Dis-/Reconnection, packet generation control flow and error management,
- **Protocol layer:** Virtual channels and commands
- **Transaction layer:** Read- and write actions by using the data connection layer
- **Session layer:** Power-, interrupt- and system management

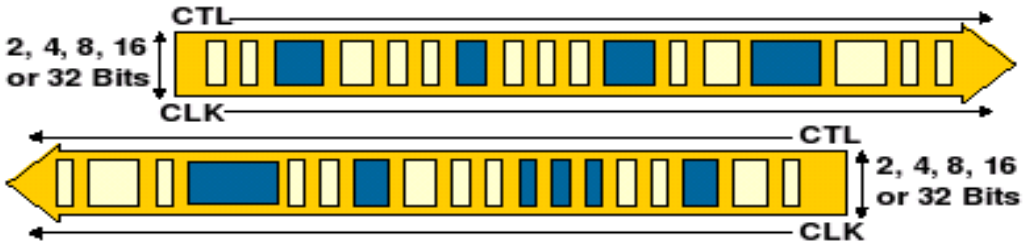
Device classes and -configurations

3 device classes are defined regarding function and location inside the HT-chain: Cave, Tunnel und Bridge

- HT Bridge: Connector between primary side (CPU resp. memory) and sekundary side (HT devices) of a chain
- HT Tunnel: has two sides each with a receive and a send unit, e.g. network card of bridge to further protocol
- HT Cave: markes end of a chain and has only one communication side. By connecting of minimal a HT bridge and a HT Cave an easy HT chain can be estabilished.

Bit Transmission Layer I

Establishing a HT connection



- Two unidirectional point-to-point data pathes
- Data path width: 2,4 8 and 16 bit depending on device requirements
- Commands, adressing and data (CAD) use the same signal wires $\rightarrow$ smaller wire count, smaller packets, smaller energy consumption, improved thermal properties
- Packets contain CAD and are multiple of 4 Byte (32 bit)
- Connections with lower than 32 bit use subsequent cycles to transmit packets completely

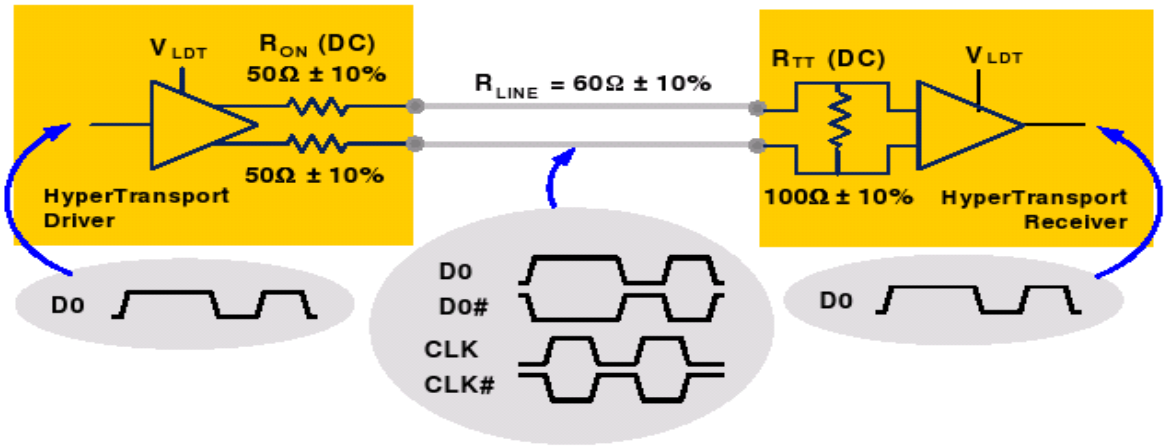
High performance and scalability

- High data rates because of low voltage differential signaling (LVDS, 1,2V $\pm$ 5%)
- Scalable bandwidth by scaling of transmission frequency and link width

Bit Transmission Layer II

Hypertransport Low Voltage Differential Signaling

- Differential signal transmission using two wires: voltage difference ($\pm$ 0,3 V) represents logic state Logic change by repoling of wires (symmetric signal transmission, double-ended)
- HT signal transmission complies to extended IEEE LVDS standard (1,2 V instead of 2,5 V)
- 60 cm maximal wire length at 800 Mbit/s
- Transceiver are integrated into control chips, 100 Ω impedance avoids reflections
- Simple realizable using standard 4-layer PCBs (Printed Circuit Boards) and ready for future



Bitübertragungsschicht III

Electric signals

Signal Name	Description	Comment
CAD	**Commands, Addresses and Data:** Carries command, address, or data information.	CAD width can be different in each direction.
CTL	**Control:** Used to distinguish control packets from data packets.	
CLK	**Clock:** Forwarded clock signal.	Each byte of CAD has a separate clock signal. Data is transferred on each clock edge.
PWROK	**Power OK:** Power and clocks are stable.	Single-ended.
RESET#	**HyperTransport Technology Reset:** Resets the chain.	Single-ended.
LDTSTOP#	**HyperTransport Technology Stop:** Enables and disables links during system state transitions.	Used in systems requiring power management. Single-ended.
LDTREQ#	**HyperTransport Technology Request:** Requests re-enabling links for normal operation.	Used in systems requiring power management. Single-ended.

For any 8 bit data width there is a clock wire from sender to receiver that is used to scan the data on the 8 data wires at the receiver (source synchronous clock)

$\rightarrow$ minimal deviation from source clock

Bit Transmission Layer IV

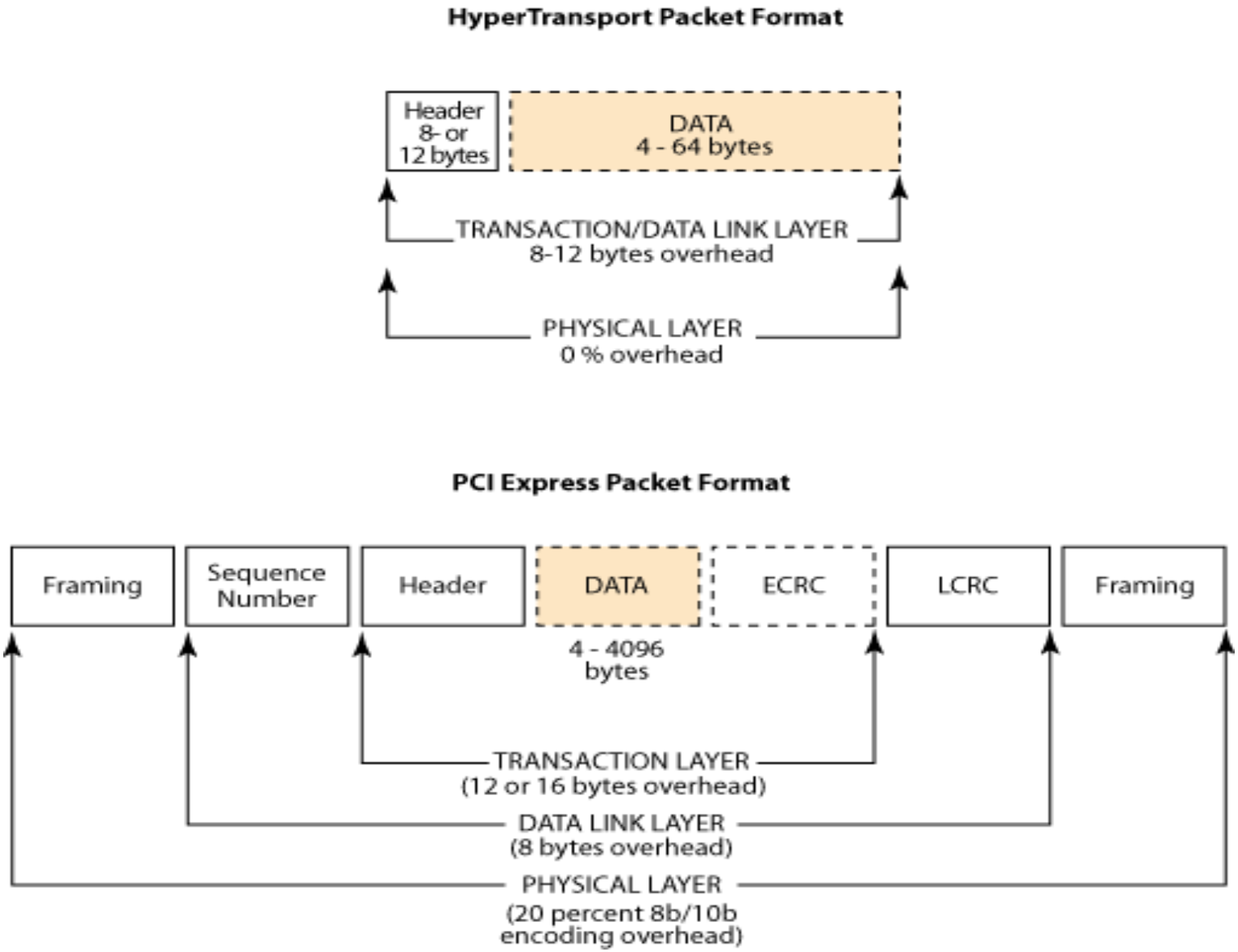
Band width scaling

- Data transmission by CAD at rising and falling side of clock signal (DDR)
  $\rightarrow$ connection cycle at 800 MHz corresponds to 1600 MHz data cycle
- Wire count enables adaptation onto band width needs
  $\rightarrow$ 2 x 16 CAD bits + 800 GHz clock = 2 x 3,2 GByte bandwidth (103 Pins)
  $\rightarrow$ 2 x 2 CAD bits + 400 MHz clock = 2 x 200 MByte bandwidth (24 Pins)

Link Width (Each Way)	2	4	8	16	32
Data Pins (total)	8	16	32	64	128
Clock Pins (total)	4	4	4	8	16
Control Pins (total)	4	4	4	4	4
Subtotal (High Speed)	**16**	**24**	**40**	**76**	**148**
V_{LOT}	2	2	3	6	10
GND	4	6	10	19	37
PWROK	1	1	1	1	1
RESET#	1	1	1	1	1
Total Pins	**24**	**34**	**55**	**103**	**197**

Connection Layer

Packet structure of HT compared to PCI-E



Protocol- and Transaction Layer

- Protocol layer: Commands, virtual channels and flow control
- Transaction layer: Performs actions e.g. read requests and responses

Command overview

Virtual Channel	Command	Comment
Posted	Posted Write	Followed by data packet(s).
	Broadcast	Issued by host bridge downstream to communicate information to all devices.
	Fence	All posted requests in a stream cannot pass it.
Non-Posted	Non-Posted Write	
	Read	Designates whether response can pass posted requests or not.
	Flush	Forces all posted requests to complete.
	Atomic Read-Modify-Write	Generated by I/O devices or bridges and directed to system memory controlled by the host.
Responses	Read Response	Response to read command, is followed by data packet(s).
	Target Done	A transaction not requiring returned data has completed at its target.

Packet Structure

Bit-Time	7	6	5	4	3	2	1	0
0	SeqID[3:2]		Cmd[5:0]					
1	PassPW	SeqID[1:0]	UnitID[4:0]					
2	*Command-Specific*							
3	*Command-Specific*							
4	Addr[15:8]							
5	Addr[23:16]							
6	Addr[31:24]							
7	Addr[39:32]							

Packet Routing

Routing methods of HT-3.0 specification

- Store-and-Forward routing: A packet is received completely and buffered, the check sum (CRC) is calculated and compared to packet crc, the packet header is decoded for receiver determination, then the packet itself is processed or routed over the connection in receiver direction (1 hop)
- Cut-Through Routing: Routing of packet as soon as the packet header is received and decoded, the packet is routed without buffering
  Problem: further-routed packets with CRC error Solution: Termination of routing, receiver determines packet error and removes the packet
- Speculative Routing: Speculation on correctness and receiver port, only determine whether the packet specifies a correct command (1 byte)
- Speculative Routing with aggr. implementation: Packet is routed immediately with the first clock cycle and without any decoding

All methods can be implemented without changing the connection specification!

Packet Routing

Packet with n bytes (incl. CRC), bandwidth w bit, pathlength d hops
t_{wire} transmission time, t_{proc} time for CRC check + receiver determination
h byte to receive til forwarding
Latency of different routing methods

- Store-and-Forward Routing

$$L = d \cdot (n \cdot 8/w + t_{wire} + t_{proc})$$

- Cut-Through Routing

$$L = d \cdot (max(1, 8 * h/w) + t_{wire} + t_{proc}) + 8/w \cdot (n - max(h, w/8))$$

- Speculative Routing

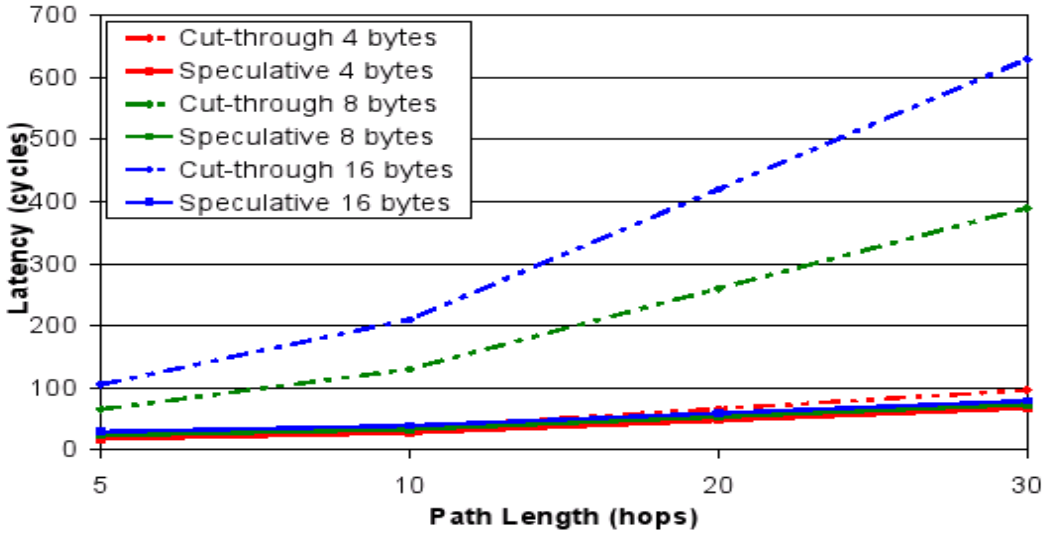
$$L = d \cdot (max(1, 8/w) + t_{wire} + t_{proc}) + 8/w \cdot (n - max(1, w/8))$$

- Speculative Routing with aggr. implementation

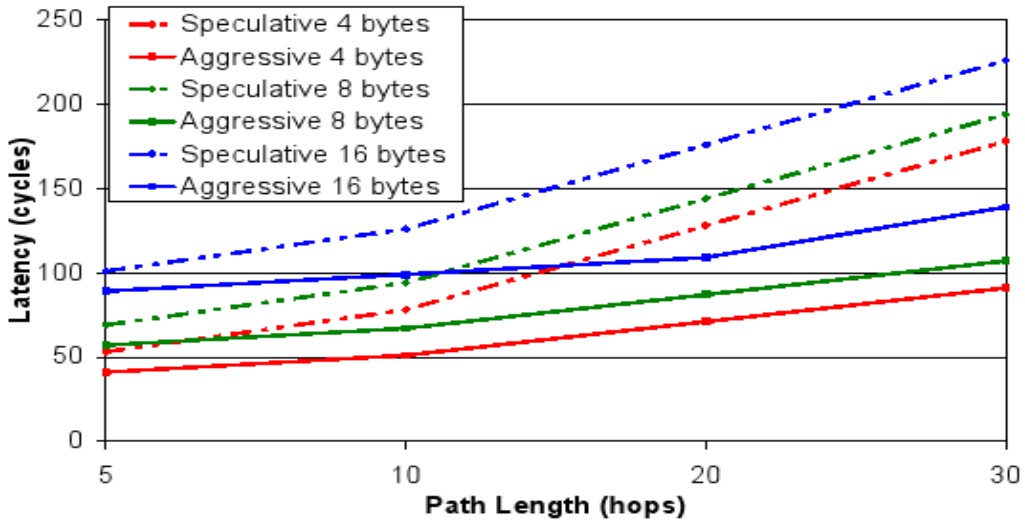
$$L = d \cdot (1 + t_{wire} + t_{proc}) + 8/w \cdot (n - 1)$$

Packet Routing

Latency for standard vs. aggressive speculative routing



8-bit connection width



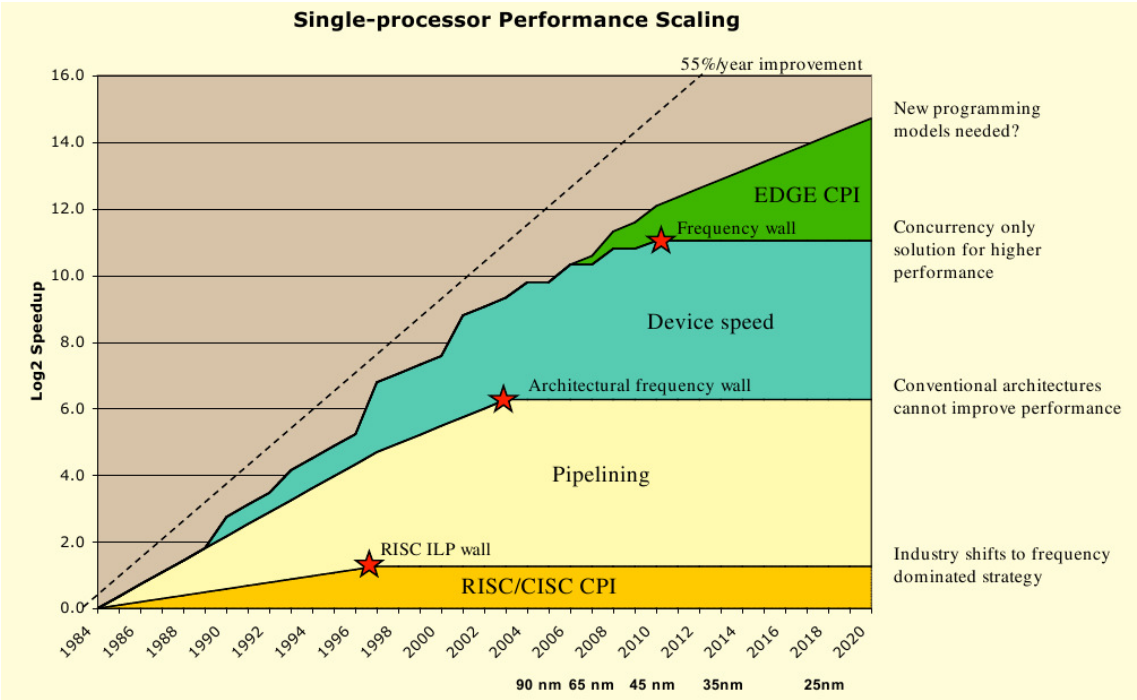
Innovative Architectural Approaches

Two development directions

- Design beyond the RISC architecture
  - TRIPS
  - WaveCore
- Many-Core processors
  Many-Core chip development (chronological)
  - Intel Tera
  - Intel Larrabee
  - Tilera Tile64
  - Adapteva Epiphany

TRIPS Processor

- TRIPS = Tera-op, Reliable, Intelligently adaptive Processing System
- Research processor (UTA, Texas)
- Processor architecture enables easy addition of further cores
- Project funding e.g. by IBM, Intel, DARPA, NSF
- EDGE architecture (Explicit Data Graph Execution) based on blocks, that execute elementary instructions independent of each other as well as data-controlled (out-of-order) instruction execution.
- Goal: Realisation of processors with multi Teraflops performance



Design of TRIPS Architecture I

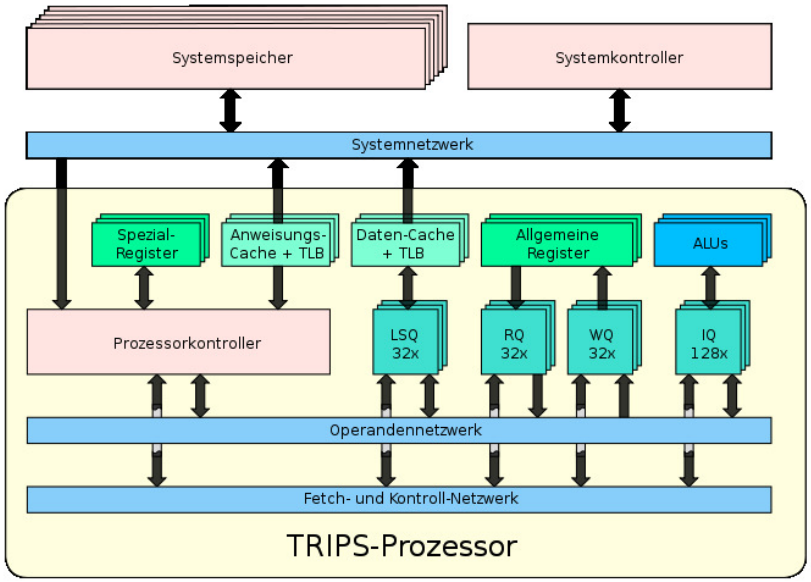
RISC-based Features

- Set of arithmetic-logic units (ALU), caches and registers
- ACUs: Integer and floating-point operations
- Separate caches for data and instructions
- Several compilation buffers (TLB), that map virtual and physical addresses
- Register subdivision into general and special registers (Special Function Register, SFC):
  - General register for arbitrary data or addresses
  - Special register for configuration and control of processor status

Design of TRIPS Architecture II

TRIPS specific concepts

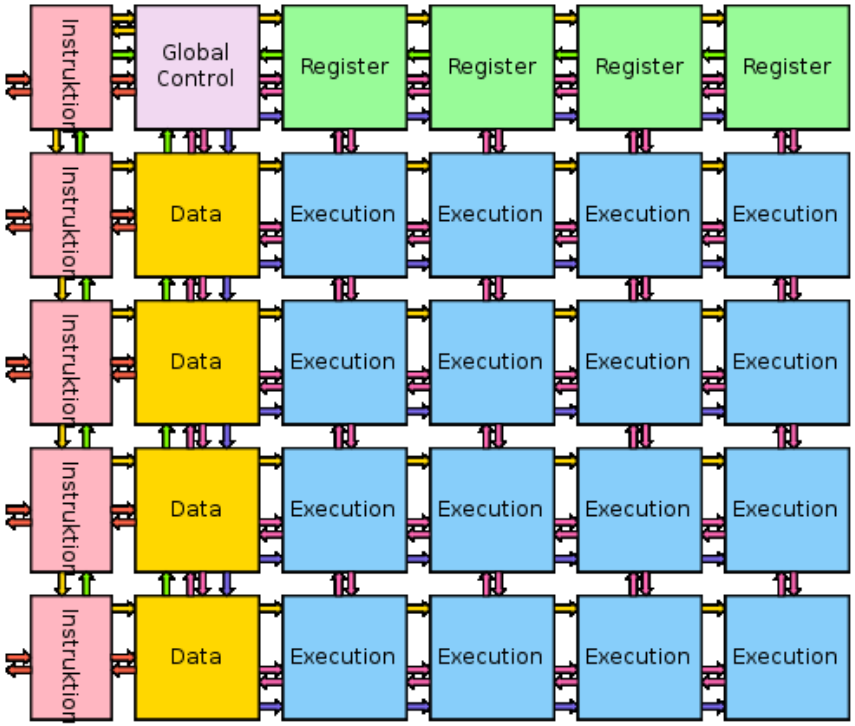
- Definition of a set of internal queues as part of the instruction set and the data flow model
- This enables to execute a series of instructions, instead of only a single instructions
- Rest of TRIPS consists of a system wide network, that connects the different computing blocks
- Access of processors, that want to access the shared memory over this network, are controlled by the system controller



- Instruction Queue (IQ) processes up to 128 instructions at a time
- Read Queue (RQ) buffers 32 read accesses onto general registers
- Write-Queue (WQ) buffers up to 32 write accesses onto general registers
- Additional Load and Store Queue (LSQ) buffers 32 memory accesses
- Queues buffer only transient states, while persistent states are kept in registers, caches and the system memory

Implementation of TRIPS Architecture I

- TRIPS processors are build out of individual tiles
- Each tile fulfills an elementary function
- The individual tiles are organized in a two-dimensional array
- We distinguish the following types of tiles:
  - Execution tiles (ET) contain IQ and ALU.
  - Register tiles (RT) contain general register, RQ and WQ.
  - Data tiles (DT) contain data cache, Data TLB and LSQ.
  - Instruction tiles (IT) contain instruction cache and instruction TLB.
  - Global control tile (GT) contain the special registers and the logic of the global processor control unit.



Implementation of TRIPS Architecture II

- Most processor resources are additionally subdivided into banks and can therefore be used by several tiles
- TRIPS architecture thus shapes a grid architecture at processor level
- This enables high clock frequences, high parallelism of instructions and good extendability
- Disadvantage: High latency in grid architectures, if data of a tile are used by a tile far-away

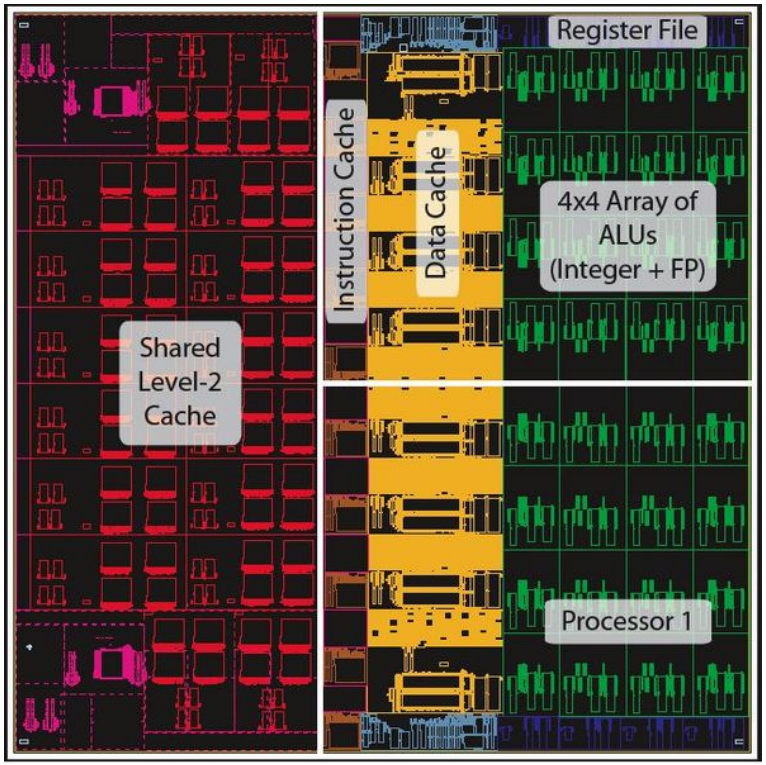
→ scalability problem is at least possible

- Advantage of grid architectures: Queues and register exist multiple times in several instances of identical tiles

→ Processing of a large instruction count as well as up to four threads in parallel

Prototype of TRIPS Processor I

TRIPS prototype with two TRIPS prozessor cores and L2 cache (Secondary Memory System) with interfaces for peripherals to the mainboard is assembled on a single prozessor die



Kürzel	Name	Beschreibung
NT	Network-Tile	Bilden ein Netzwerk, in dem Daten aus und in den Speicher transportiert werden.
MT	Memory-Tile	Bilden den Speicher des L2-Cache, in dem Daten gespeichert werden.
DMA	Direct Memory Access	Kontroller für den Speicherdirektzugriff (Northbridge)
SDC	Static DRAM Controller	Bietet Speicherzugriff auf die SDRAM-Bänke des Arbeitsspeichers
EBC	External Bus Controller	Stellt die Verbindung mit Bussen (Southbridge) her, die sich außerhalb des Prozessors befinden, und kümmert sich um Unterbrechungsanforderungen (*Interrupt Request*, *IRQ*) und externe Busschnittstellen (*External Bus Interface*, *EBI*).
C2C	Chip-to-Chip Connector	Dient dazu, eine direkte Verbindung mit anderen TRIPS-Prozessoren herzustellen. Der C2C ist im TRIPS-Prozessor vierfach vorhanden, um Arrays aus TRIPS-Prozessoren zu bilden und damit Rechencluster aufbauen zu können.

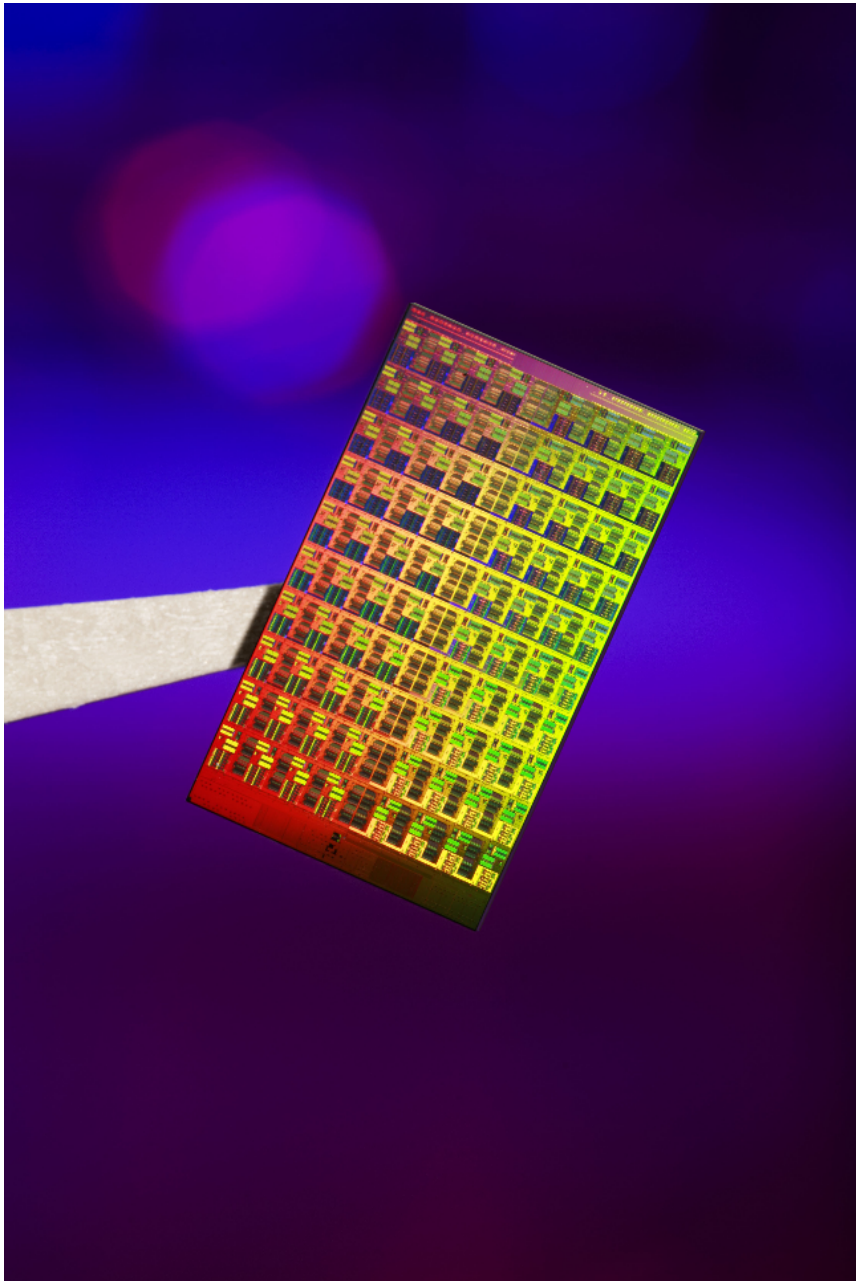
- Prototype assembling as ASIC in a 130 nm process with about 170 million transistors
- 4-fold multithreading and execution of up to 16 instructions each cycle and core at clock frequency of 500 MHz

→ Peak performance of 16 GOps

Prototype of TRIPS Processor II

Dataflow oriented execution
Individual instructions blocks (up to 128 instructions) are not processed in the sequence of instructions as in traditional processors but in the sequence of the data flow.
Dependencies of instructions onto each other are directly stored with the instructions. An instruction is executed as soon as all its data needed is available.

Intel: Tera



Intel TeraFlop Processor (2007)

- 80-core (8x10 array), 100 Mio. Trans, 65nm, $275 mm^2$
  - 2 FPUs with multiply/accumulate operation (FMAC) in SP with a 9 stage pipeline
  - 3 KB instruction memory (IMEM) = 256 96-bit instructions.
  - 2 KB of data memory (DMEM) = 512 single precision numbers.
  - 10 port (6-read, 4-write) 32 entry register file
- network-on-a-chip
  - A five port router
  - 16 Gigabytes/sec over each port
- performance
  - 4 FLOPS/cycle/core at 4,27 GHz and 1,07V = 1,37 TFLOPs peak
  - 97 Watt, 80 degrees

Intel: Tera

Performance of 4 application kernels

Kernel	Problem sizes and FLOP counts:	Theoretical peak performance projections	Observed cycle counts and single precision TFLOPS @ 4.27 GHz
Stencil on NxM grid	M=2240, N=16 FLOP = N*M*10 = 358400	Even mix of adds and multiplies overlapped with loads and comm. for 95% of peak performance.	1536 cycles per iteration 1 TFLOP
Matrix Multiplication C(N,N) = A(N,M)*B(M,N)	N=80, M=206 FLOP = N*N(2*M-1) = 2630400	Limited by loads from DMEM to 50% of peak performance	22000 cycles 0.51 TFLOPS
Spreadsheet: LxN table of (value,weight) pairs	N = 10 L = 1600 FLOP = 4*L*N-N-L = 62390	Limited by loads from DMEM to 50% peak performance	Two phases: 1st phase 484 cycles; 2nd phase 589 cycles Pipelining over multiple spreadsheets to achieve 0.45 TFLOPS
2D FFT $Y(N,N)=F_N \otimes F_N * X(N,N)$	N=64 FLOP = $8\ N^2 \log_2 N$ = 196608	Limited by integer and floating point operation mix and redundant operations to 17.5% of peak	41846 cycles 0.020 TFLOP

Programming the Intel 80-core network-on-a-chip Terascale Processor, 2008

Intel Larrabee

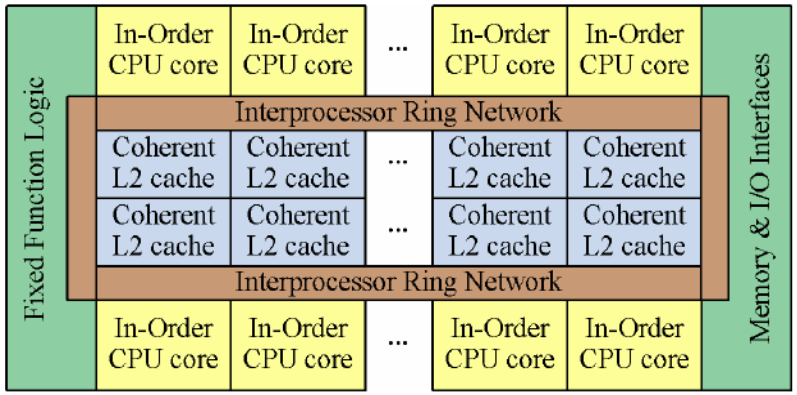
Lessons learned from Programming the 80 . . . Our future work will highlight three major advantages of message passing architectures for many core chips.

1. Managing messages and explicitly distributing data structures **adds considerable complexity** to the software development process. This complexity, however, is balanced by the greater ease of **avoiding race conditions** when all sharing of data is through explicit messages rather than through a shared address space.
2. While a programmer can block data to fit into a cache or use prefetching to prepare a cache in advance of a computation, programmers have little or **no control over eviction of data from a cache**. It is difficult to write software with predictable performance when the state of the cache is so difficult to control. A **NoC chip without cache coherence between cores avoids this problem** altogether.
3. There are software engineering advantages to message passing architectures. Well engineered software is often constructed as modules that can be composed into the final application. Composition is based on isolation of concerns; computing occurs inside a black- box module while sharing of information occurs only through the interfaces to the modules. A shared address space makes it difficult to assure that no unintended sharing is taking place. A **message passing architecture, however, naturally supports a discipline of isolation between modules** since sharing can only occur through the exchange of messages.

Intel Larrabee

Intel Larrabee (2009)

- Many-core processor on basis of modified Pentium (P4) cores with additional vector unit
- Usage as graphics processor was planed
- Ring-shaped communication topology
- Complete paradigm change
- Disappointing first real performance data
- Offically dismissed



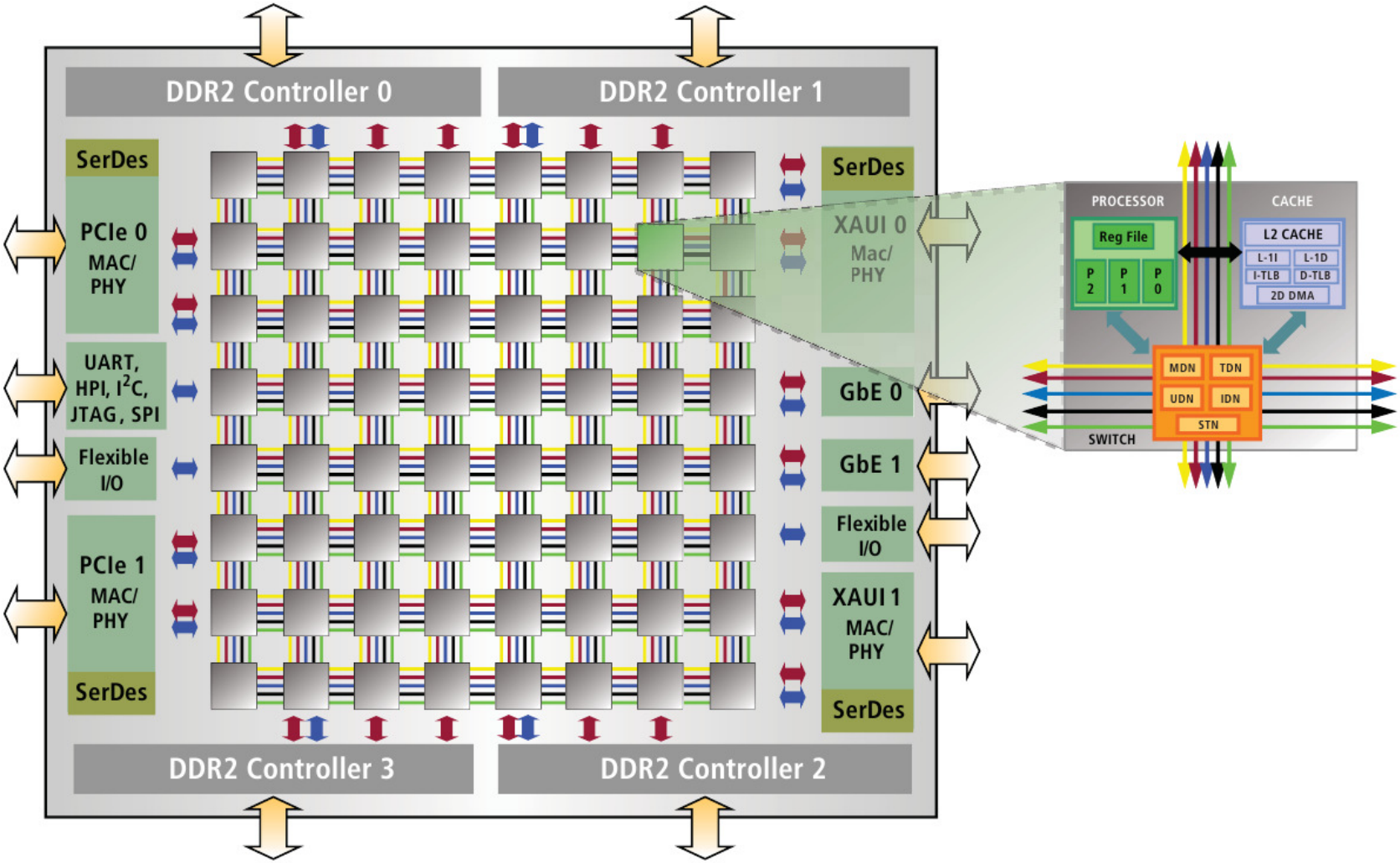
Tilera Tile64 I

Processor Architecture:

- MIMD architecture
- 2D grid of 64 homogeneus, generalpurpose compute elements (tiles)
- Tileras iMesh on-chip network
- 4 DDR2 controllers + IO controllers
- Tiles:
  - Processor
  - L1 & L2 cache
  - non-blocking switch

Tilera Tile64 II

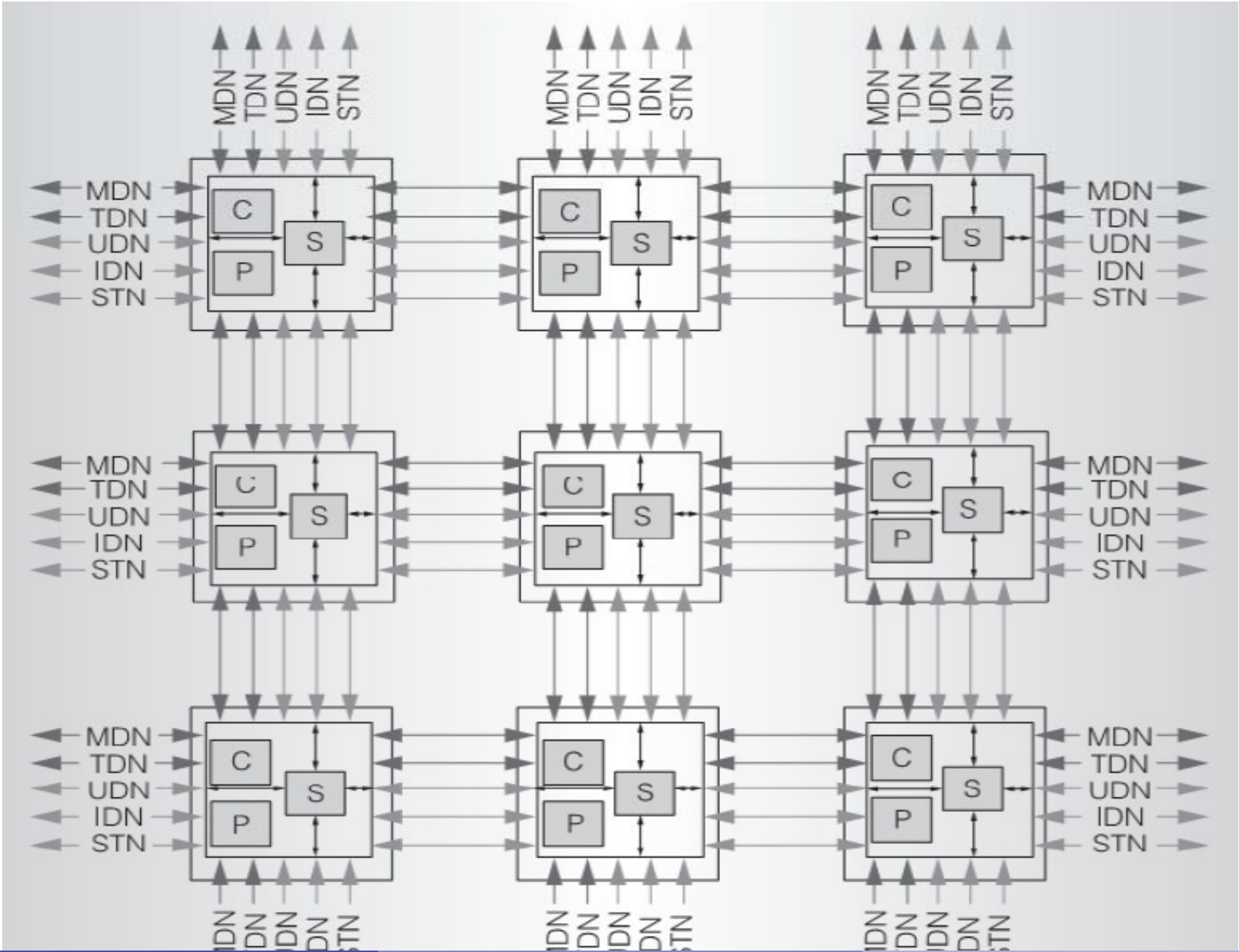
Tile64 floorplan



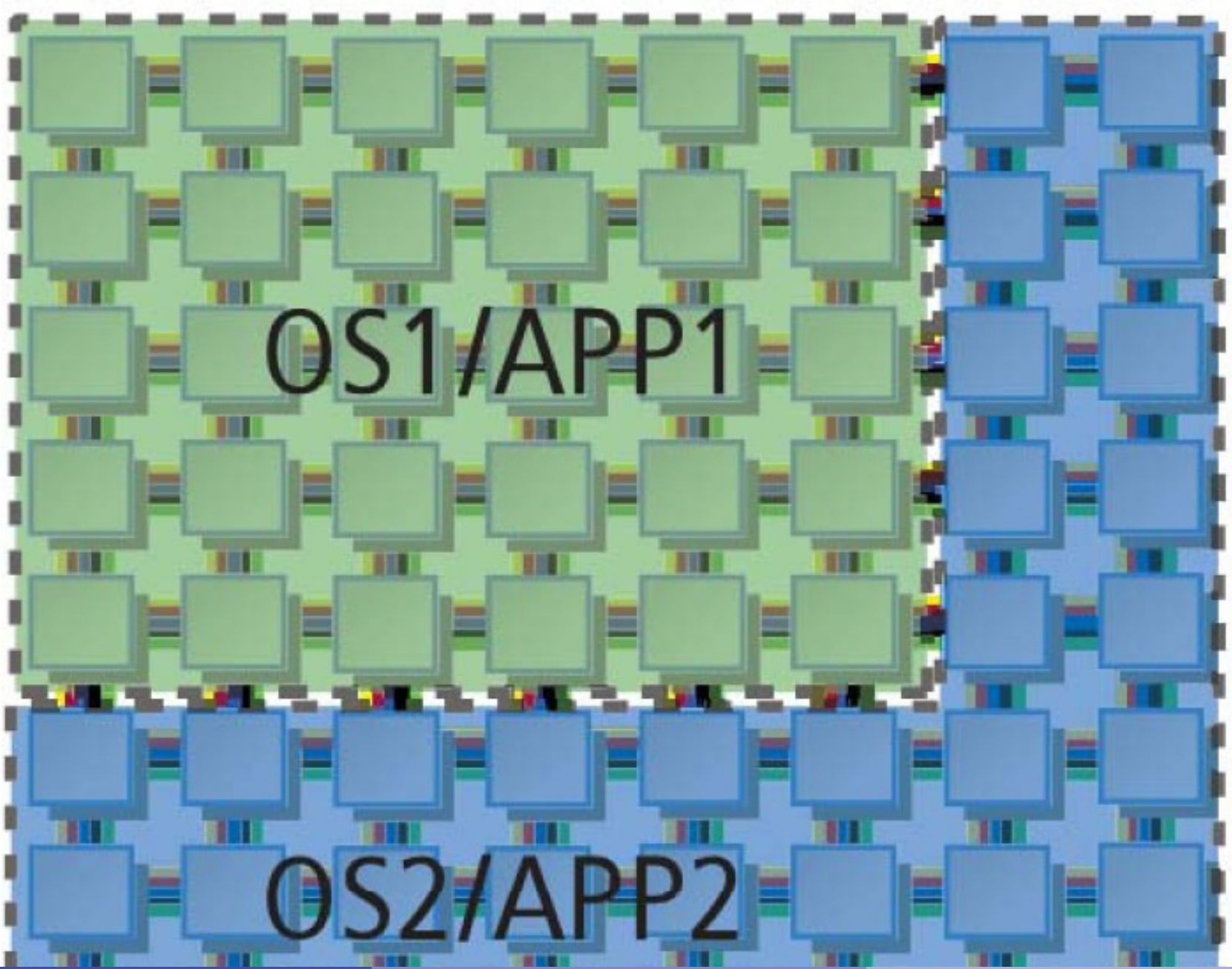
Tilera Tile64 II

	Features	Enables
Massively Scalable Performance	• 8 X 8 grid of identical, general purpose processor cores (tiles) • Three-way VLIW pipeline for instruction level parallelism • 5 Mbytes of on-chip cache • Up to 443 billion operations per second (BOPS) • 31 Tbps of on-chip mesh interconnect enables linear application scaling • Up to 50 Gbps of I/O bandwidth	• 10 Gbps Snort® processing • 20 Gbps nProbe • 16 X 16 SAD at 540 MBlocks/s • H.264 HD video encode: 2+ streams of 720p30
Power Efficiency	• 700MHz - 866MHz operating frequency • 15 - 22W @ 700MHz all cores running full application • Idle Tiles can be put into low-power sleep mode • Power efficient inter-tile communications	• Highest performance per watt • Simple thermal management and power supply design • Lower operating cost
Integrated Solution	• Four DDR2 memory controllers with optional ECC • Two 10GbE XAUI configurable MAC or PHY interfaces • Two 4-lane PCIe interfaces: Root complex or endpoint mode • Two GbE MAC interfaces • Flexible I/O interface	• Reduces BOM cost – standard interfaces included on-chip • Dramatically reduced board real estate • Direct interface to leading L2-L3 switch vendors
Multicore Development Environment	• ANSI standard C / C++ compiler • Advanced profiling and debugging designed for multicore programming • Supports SMP Linux with 2.6 kernel • Tilera Multicore Components (TMC™) libraries for efficient inter-tile communication	• Run off-the-shelf C programs • Reduce debug and optimization time • Faster time to production code • Standard multicore communication mechanisms

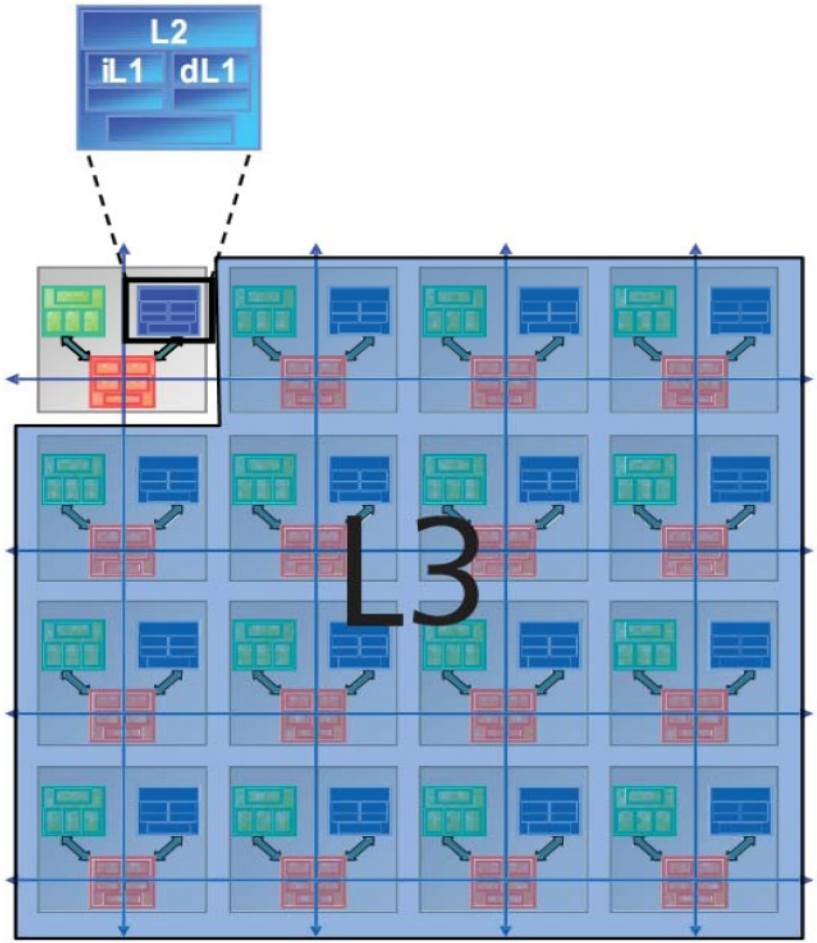
Tilera Tile64 III

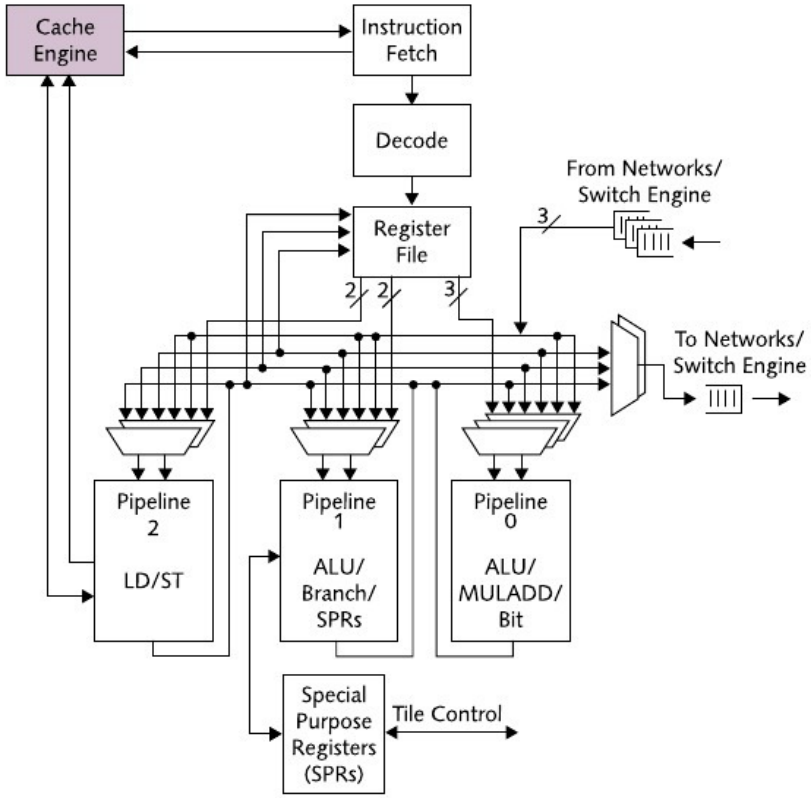


Tilera Tile64 IV



Tilera Tile64 V





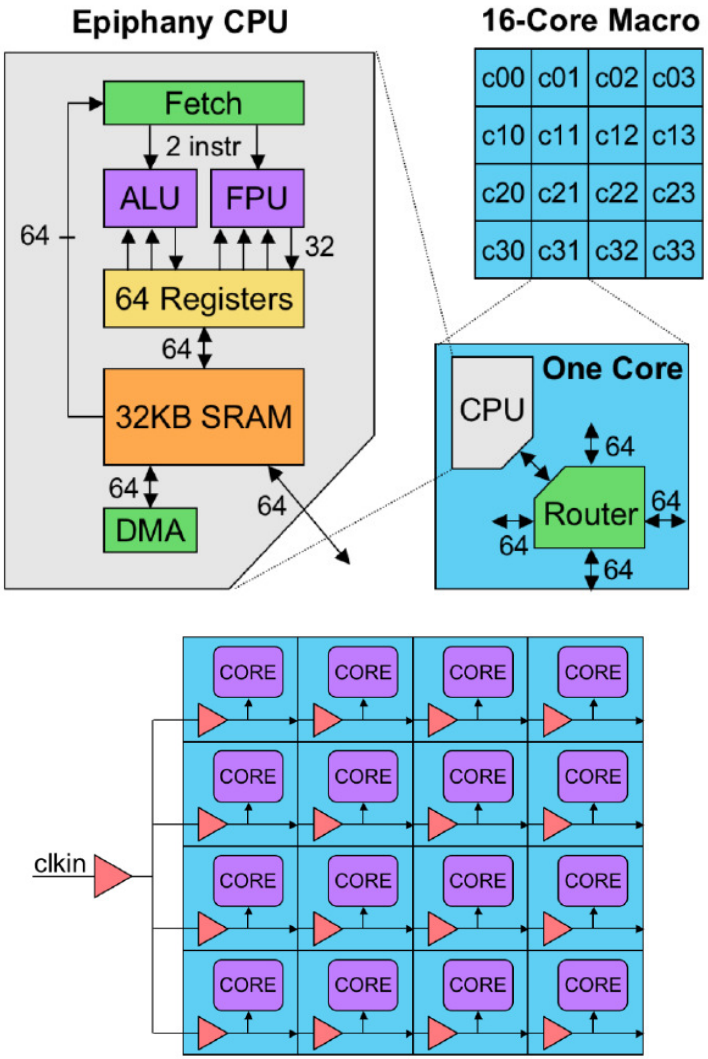
Tilera Tile64 VI

# of Cores	Bus/Mesh	Very Low Traffic	Medium Traffic	High Traffic
4 Cores	Mesh	24	25	29
	Bus	23	28	69
16 Cores	Mesh	26	29	41
	Bus	24	137	>1,000 (Unusable)
64 Cores	Mesh	28	34	58
	Bus	25	>1,000 (Unusable)	>1,000 (Unusable)

Latency in clock cycles

Adapteva Epiphany

- 2D mesh of general-purpose RISC cores hooked up via a high bandwidth, low latency on-chip network
- current implementation has 16 cores with SP FP operations, 4k-core layout with 64 W and 4 TFLOPs peak finished
- 1000 thousand core version is already in the works with DP FP (2011)
- similar to Tilera's manycore chips with focus on floating point power
- processor doesn't have a hardware cache, each core has 32 KB of local memory
- fast on-chip-network allows extremly low-latency data requests
- programming in ANSI-C with message-passing paradigm following MCAPI-Standard not MPI
- currently as co-processor in consumer mobile devices and embedded systems



Adapteva Epiphany

Comparison of mobile Processors

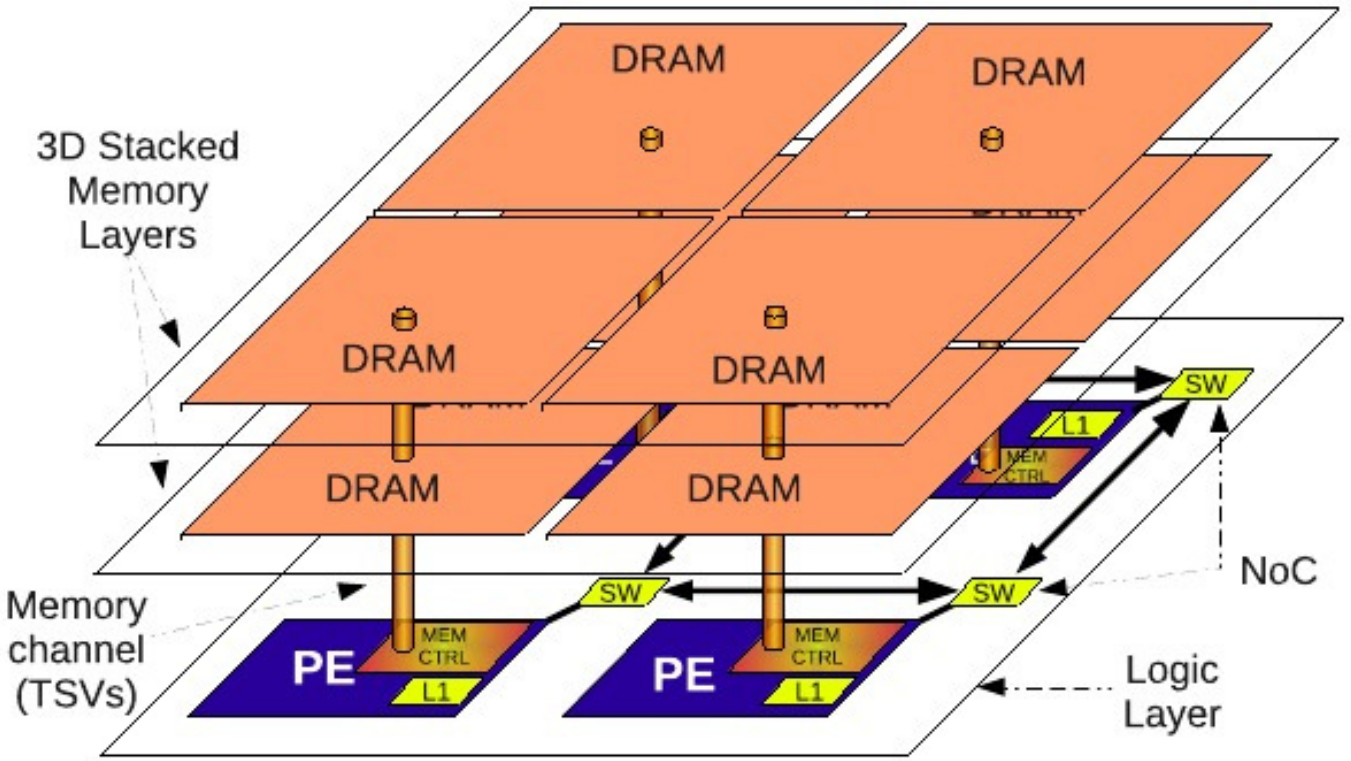
	Adapteva Epiphany*	ARM Cortex-A9*	Vivante GC2000
# of Cores	16 cores	4 cores	4 cores
Total SRAM	512KB SRAM	256KB L1$	152KB SRAM
FP Ops/Cycle	32 FP	16 FP	16 FP
Clock Speed	600MHz	1.2GHz†	1.5GHz†
Peak Mflops	19,200	19,200	24,000
Power (typ)	270mW	1,350mW†	650mW†
Mflops/W	71,000/W	14,000/W	37,000/W
Die Area	2.05mm^2	4.6mm^2†	2.8mm^2†
Mflops/mm^2	9,400/mm^2	4,200/mm^2	8,600/mm^2

NoC: Hiting the memory wall again!

Memory access speed is not the only cause of the memory wall, which is also tied to memory-to-logic interfacing issues. **DRAMs currently are developed using high-density NMOS process** optimized to create high-quality capacitors and low-leakage transistors. On the other hand, logic **chips are manufactured in high-speed CMOS processes** optimized for transistor performance and complex multi-level metalizations. The two processes are not compatible, therefore highly optimized DRAM and logic cannot coexist on the same die1 and they must be interfaced through **off-chip interconnects**. This imposes tight **constraints on the maximum DRAM pin count resulting in a limited-bandwidth** interface between DRAM and Logic chips. A number of countermeasures have been adopted to overcome this problem, such as multiplexing the address in two phase (RAS and CAS) and fast burst-transfer modes to increase per-pin bandwidth. This has lead to an ever-increasing power and signal integrity bottleneck in memory interfaces.

NoC and RAM: Possible Solution Approach?

3D stacking of NoC-processor layers and memory layers



Literature

Literature for Parallel Computer Architecture

- Hennessy J., Patterson D.: Computer Architecture – A Quantitative Approach, 3. Ausgabe, Morgan Kaufmann, 2003
- Hennessy J., Patterson D.: Computer Architecture – A Quantitative Approach, 4. Ausgabe, Morgan Kaufmann, 2007
- Culler D., Singh J., Gupta A.: Parallel Computer Architecture – A Hardware/Software Approach, Morgan Kaufmann, 1999
- HyperTransport Konsortium: www.hypertransport.org
- Hwang K.: Advanced Computer Architecture: Parallelism, Scalability, Programmability, McGraw-Hill, 1993
- Grama A., Gupta A., Karypis G., Kumar V.: Introduction to Parallel Computing, 2. Ausgabe, Benjamin/Cummings, 2003