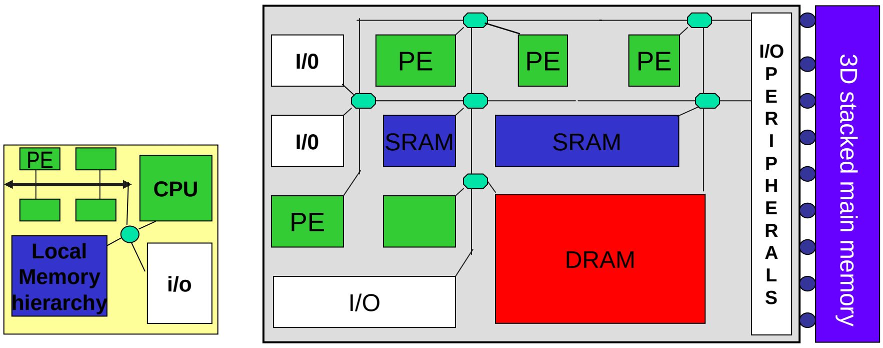
Luca Benini – DEIS Università di Bologna
lbenini@deis.unibo.it

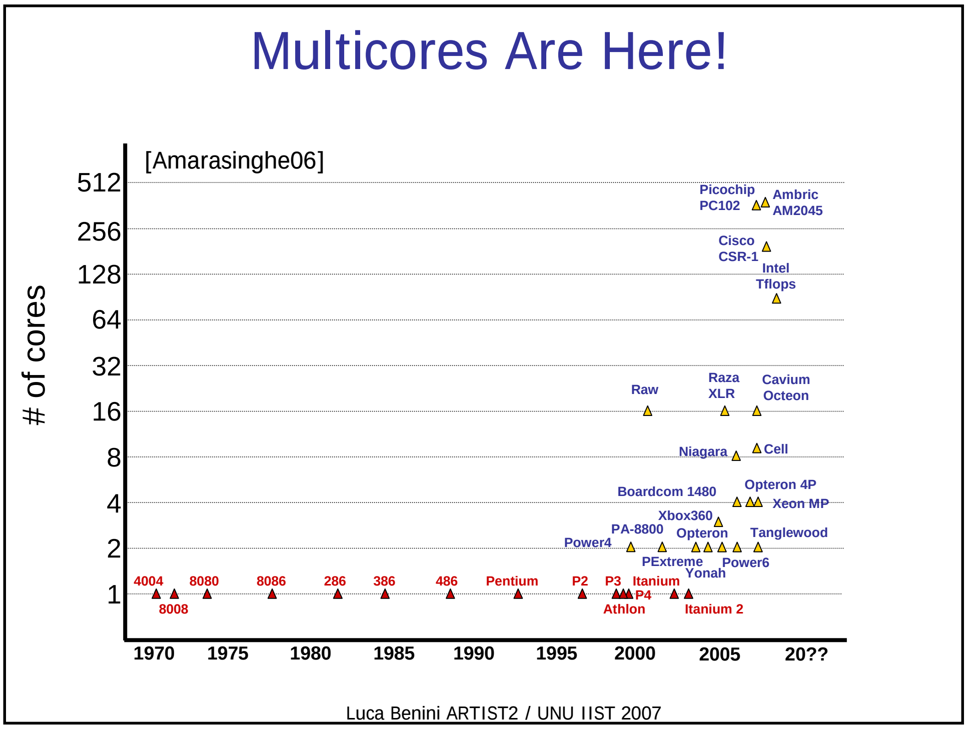
MPSoCs – Hardware platforms

- Why MPSoCs?
- MSoC architectures
- Case studies

Architecture Evolution

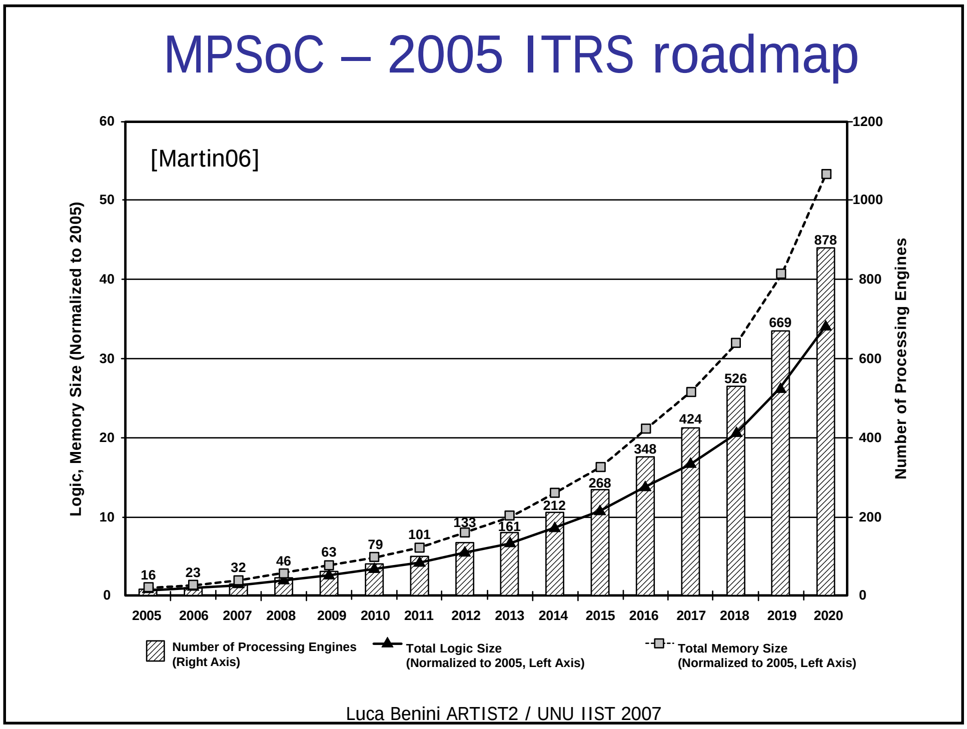


- Roadmap continues: 90→65→45 nm
- “Traditional” Bus based SoCs fit in one tile !!
- Communication demand is staggering, but unevenly distributed, because of architectural heterogeneity



Multicores Are Here!

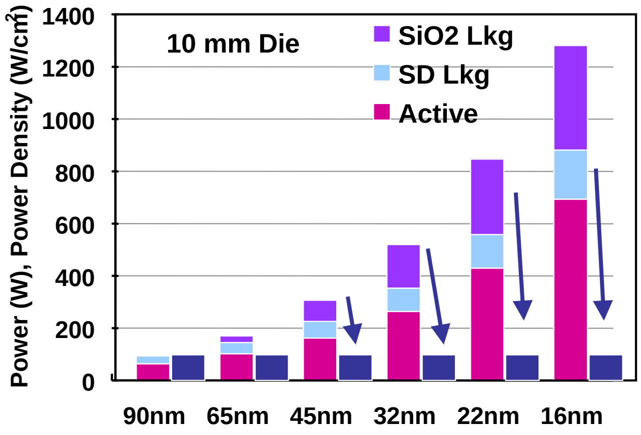
Luca Benini ARTIST2 / UNU IIST 2007



MPSoC – 2005 ITRS roadmap

Luca Benini ARTIST2 / UNU IIST 2007

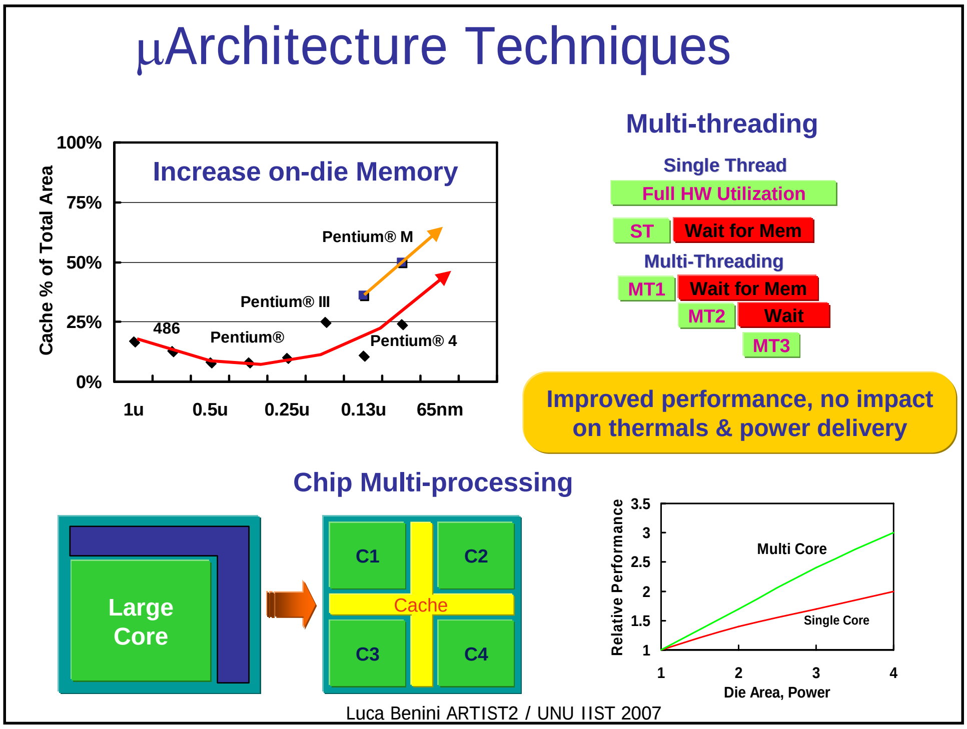
Power is the Challenge!



Technology, Circuits, and Architecture to constrain the power

Near Term Solutions

- Move away from Frequency alone to deliver performance
- More on-die memory
- Multi-everywhere
  - Multi-threading
  - Chip level multi-processing
- Throughput oriented designs
- Performance by higher level of integration

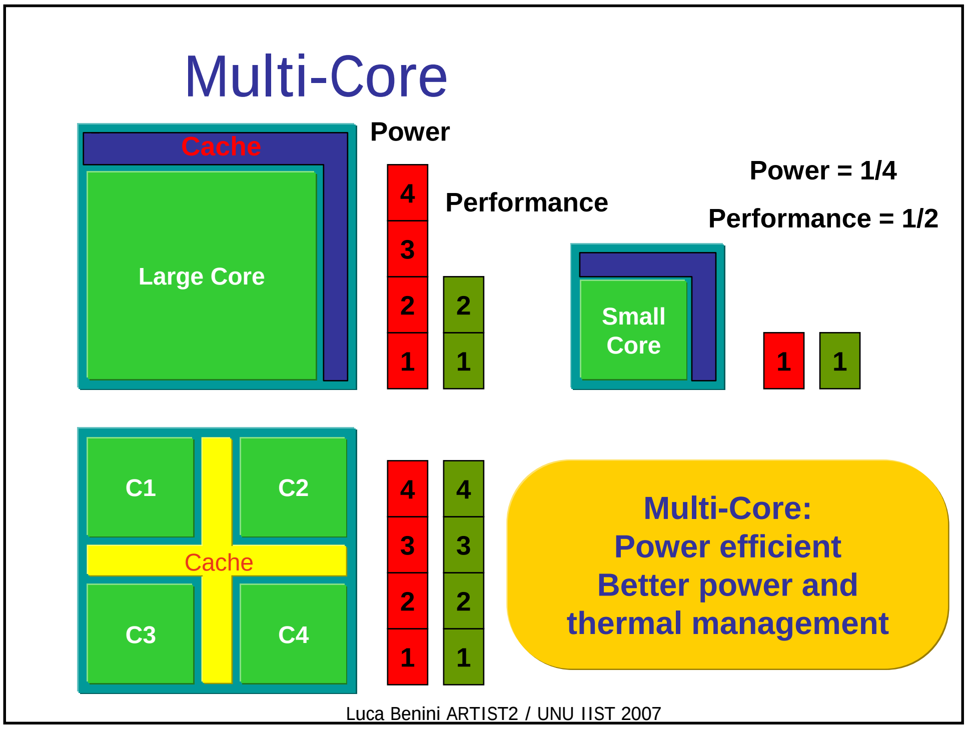


μArchitecture Techniques

Multi-threading

Improved performance, no impact on thermals & power delivery

Chip Multi-processing

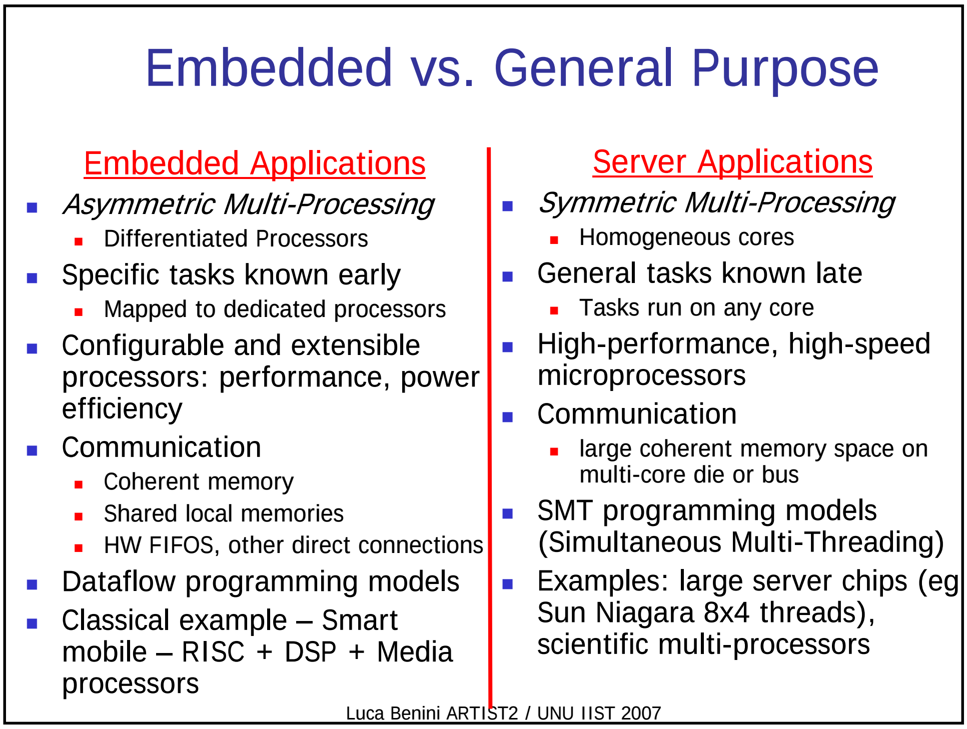


Multi-Core

Power = 1/4

Performance = 1/2

Multi-Core:
Power efficient
Better power and
thermal management



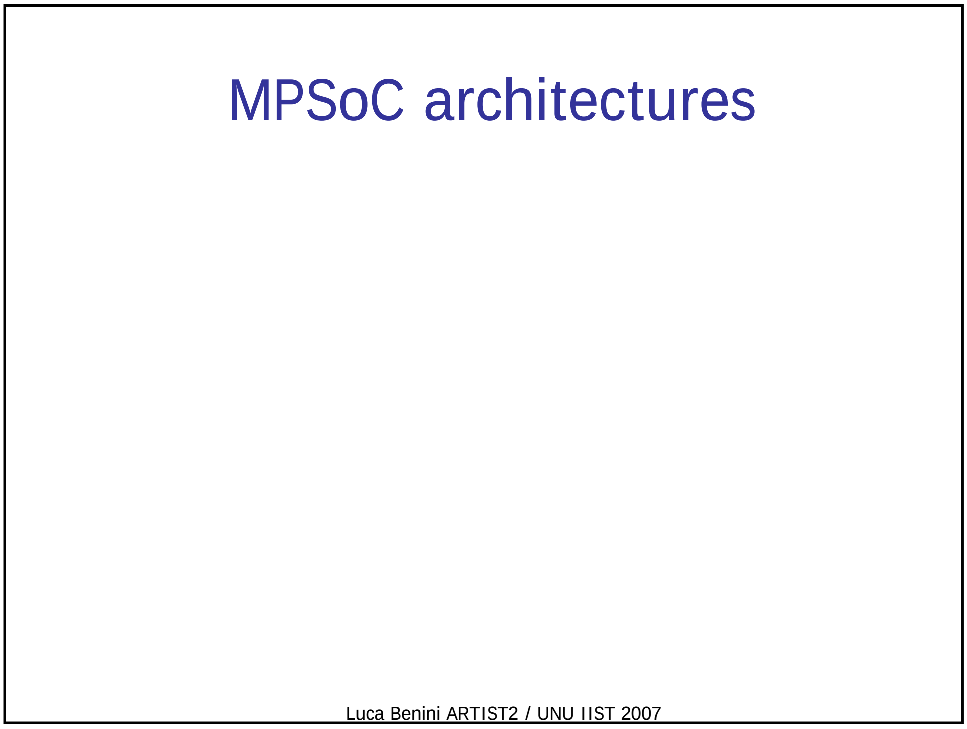
Embedded vs. General Purpose

Embedded Applications

- *Asymmetric Multi-Processing*
  - Differentiated Processors
- Specific tasks known early
  - Mapped to dedicated processors
- Configurable and extensible processors: performance, power efficiency
- Communication
  - Coherent memory
  - Shared local memories
  - HW FIFOS, other direct connections
- Dataflow programming models
- Classical example – Smart mobile – RISC + DSP + Media processors

Server Applications

- *Symmetric Multi-Processing*
  - Homogeneous cores
- General tasks known late
  - Tasks run on any core
- High-performance, high-speed microprocessors
- Communication
  - large coherent memory space on multi-core die or bus
- SMT programming models (Simultaneous Multi-Threading)
- Examples: large server chips (eg Sun Niagara 8x4 threads), scientific multi-processors



MPSoC architectures

Example system platforms

- Generic
- Automotive
- Wireless
- Multimedia

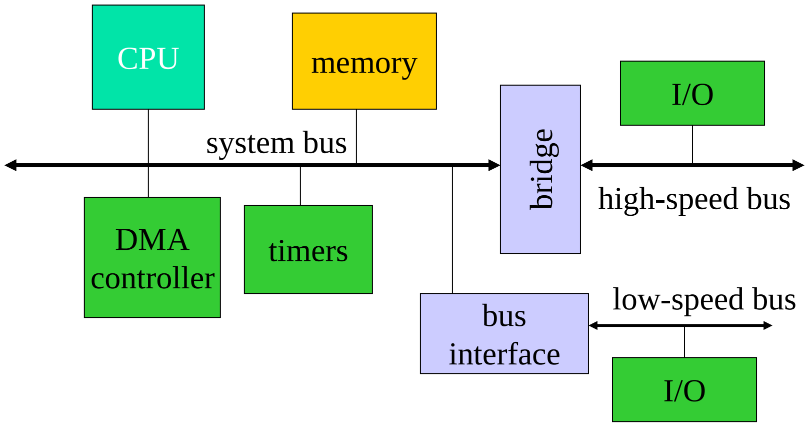
Luca Benini ARTIST2 / UNU IIST 2007

PC-based platform

- Basic hardware components:
  - CPU;
  - memory;
  - timers;
  - DMA;
  - minimal I/O devices.
- Basic software:
  - BIOS.

Luca Benini ARTIST2 / UNU IIST 2007

PC-style hardware architecture



Strong ARM

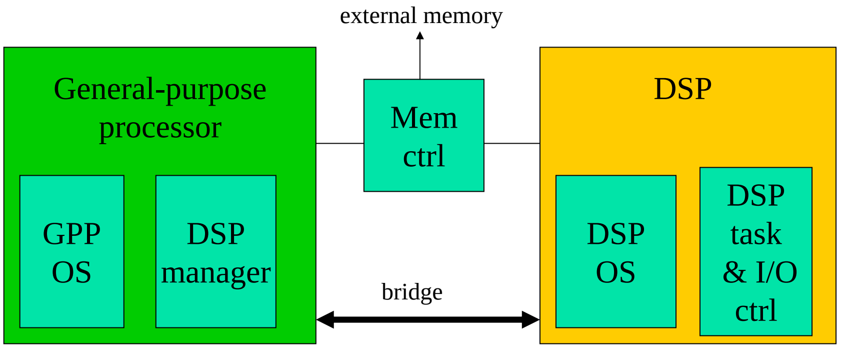
- StrongARM system includes:
  - CPU chip (3.686 MHz clock)
  - system control module (32.768 kHz clock).
    - Real-time clock;
    - operating system timer
    - general-purpose I/O;
    - interrupt controller;
    - power manager controller;
    - reset controller.

Pros and cons

- Plentiful hardware options.
- Simple programming semantics.
- Good software development environments.
- Performance-limited.

TI Open Wireless Multimedia Applications Platform

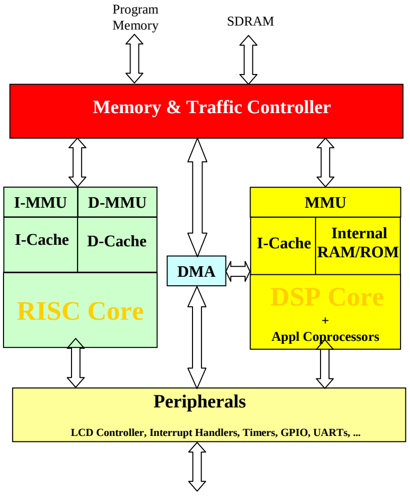
- Dual-processor shared memory system:



http://www.ti.com/sc/docs/apps/wireless/omap/overview.htm

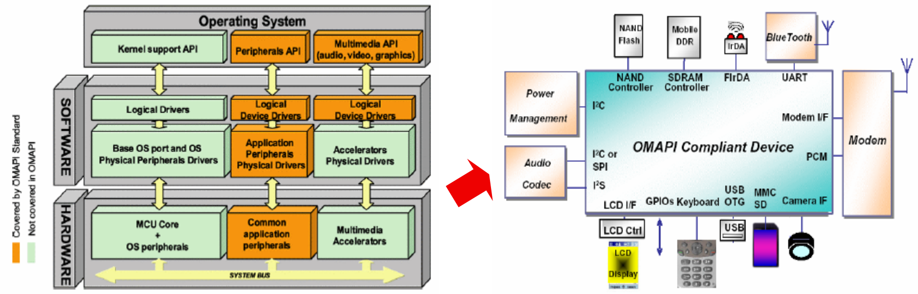
TI OMAP™ Hardware platform

- ARM9 core
- 16KB I-cache
- 8KB D-cache
- 2-way set associative
- 150 MHz



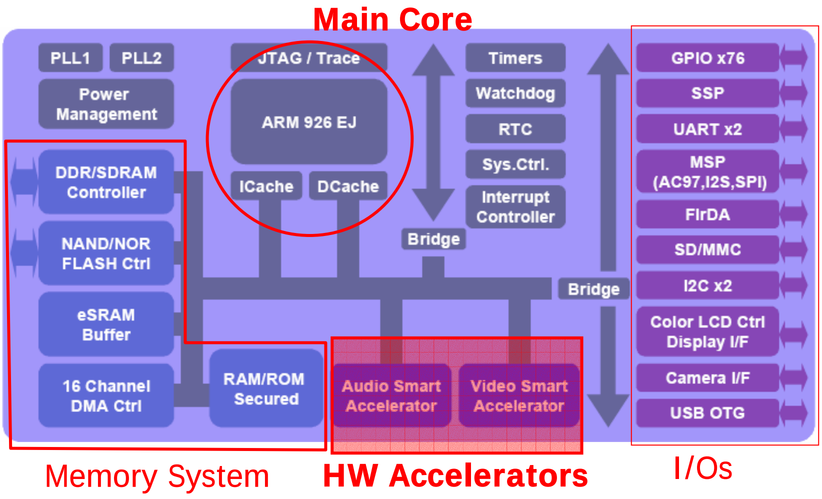
- C55x DSP core
- 16KB I-cache
- 8KB RAM set
- 2-way set associative
- 200 MHz

OMAPI Standard (ST/TI)



- Goal: standardize the interfaces between application processor and peripheral devices in a mobile product
- Provide standard services (APIs) in the OS that can be used by application developers

STMicro Nomadik platform



Main Core

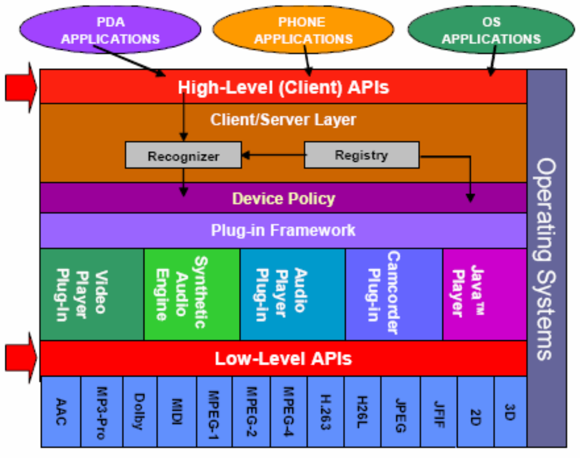
PLL1
PLL2
Power Management
JTAG / Trace
ARM 926 EJ
ICache
DCache
Timers
Watchdog
RTC
Sys.Ctrl.
Interrupt Controller
Bridge
Bridge
DDR/SDRAM Controller
NAND/NOR FLASH Ctrl
eSRAM Buffer
16 Channel DMA Ctrl
RAM/ROM Secured
Audio Smart Accelerator
Video Smart Accelerator
GPIO x76
SSP
UART x2
MSP (AC97,I2S,SPI)
FIrDA
SD/MMC
I2C x2
Color LCD Ctrl Display I/F
Camera I/F
USB OTG

Memory System

HW Accelerators

I/Os

Nomadik SW platform



PDA APPLICATIONS
PHONE APPLICATIONS
OS APPLICATIONS
High-Level (Client) APIs
Client/Server Layer
Recognizer
Registry
Device Policy
Plug-in Framework
Video Player Plug-in
Synthetic Audio Engine
Audio Player Plug-in
Camcorder Plug-in
Java™ Player
Low-Level APIs
AAC
MP3-Pro
Dolby
MIDI
MPEG-1
MPEG-2
MPEG-4
H.263
H26L
JPEG
JFIF
2D
3D
Operating Systems

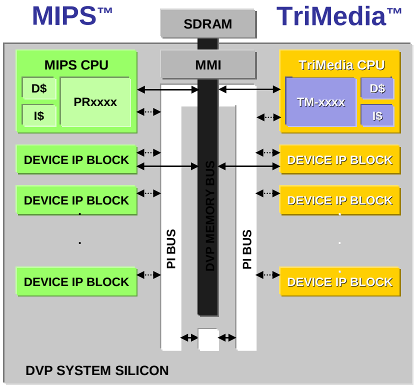
- Compliant with OMAPI standard

Philips Digital Video Nexperia Platform

General-purpose Scalable RISC Processor
- 50 to 300+ MHz
- 32-bit or 64-bit

Library of Device IP Blocks
- Image coprocessors
- DSPs
- UART
- 1394
- USB

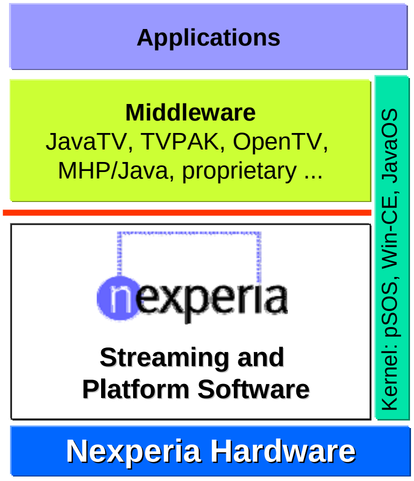
…and more



Scalable VLIW Media Processor:
- 100 to 300+ MHz
- 32-bit or 64-bit

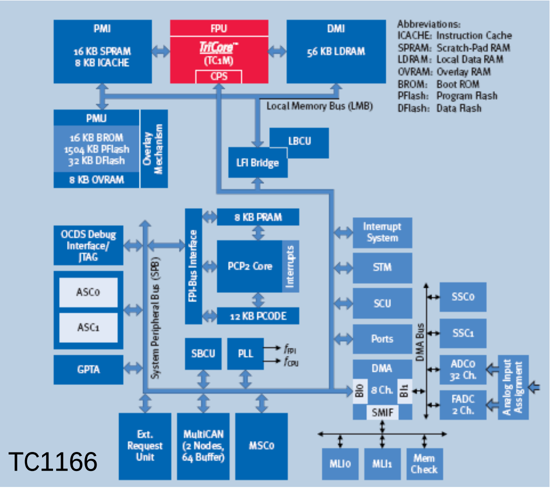
Nexperia™ System Buses
- 32-128 bit

Nexperia-DVP Software


Applications
Middleware
JavaTV, TVPAK, OpenTV, MHP/Java, proprietary ...
nexperia
Streaming and Platform Software
Kernel: pSOS, Win-CE, JavaOS
Nexperia Hardware

- Nexperia™ -DVP Software Architecture
  - Supports multiple OSs and middleware software
  - Abstracts platform functionality via consistent APIs
- Nexperia™-DVP Streaming Software
  - Encapsulates implementation of streaming media components (hardware and software)
- Nexperia™ Platform Software
  - OS independent device drivers for on-chip and off-chip devices

Infineon Automotive Platform



TC1166

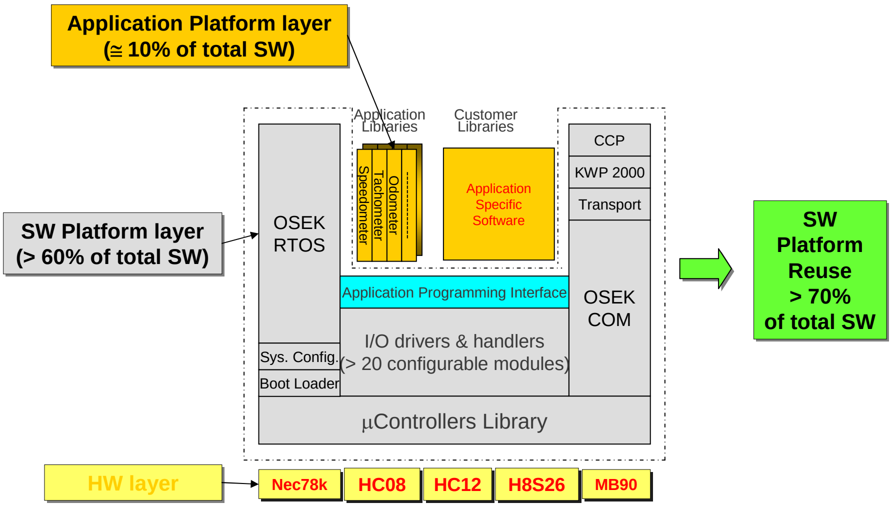
Applications

- High Performance drives / servo drives,
- Industrial control Robotics

Features

- 32-bit super-scalar TriCoreTM V1.3 CPU, 4 stage pipeline
  - Fully integrated DSP capabilities
  - Single precision floating point unit (FPU)
  - 80 MHz at full industrial temperature range
- 32-bit peripheral control processor with single cycle instruction (PCP2)
- Memories
  - 1.5 MByte embedded progr. flash with ECC
  - 32 KByte data flash - EEPROM emulation
  - 56 KBSRAM, 8 KB I$, 16 KB Imem
- 8-channel DMA controller
- Interrupt system with 2 x 255 hardware priority arbitration levels serviced by CPU and PCP2 Coprocessor
- Triple bus structure: 64-bit local memory buses to internal flash and data memory, 32-bit system peripheral bus, 32-bit remote peripheral bus

MOSAIC SW Architecture & Components for Automotive Dashboard and Body Control



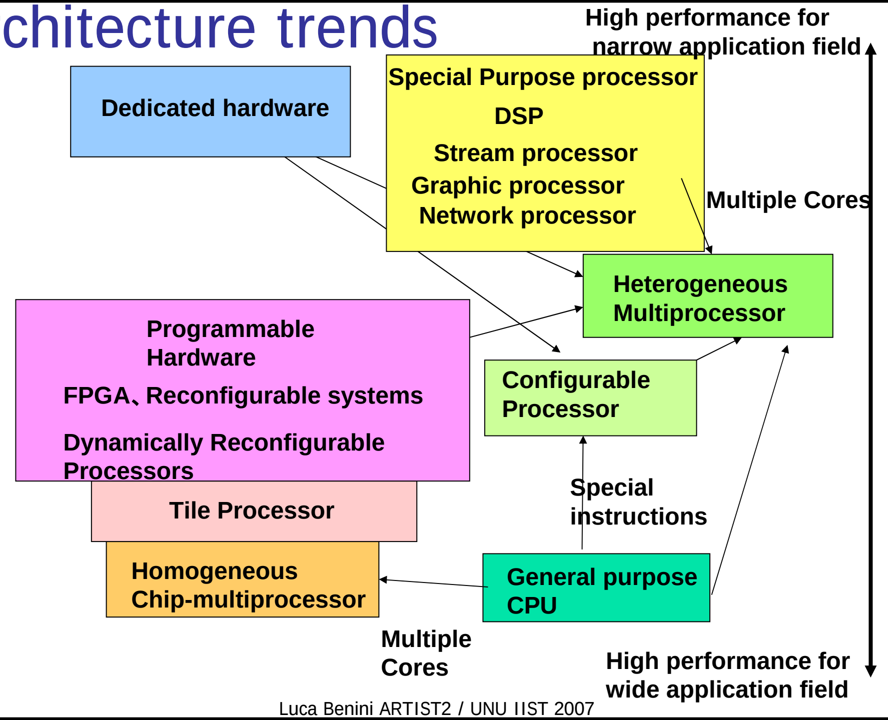
Application Platform layer (≅ 10% of total SW)

SW Platform layer (> 60% of total SW)

HW layer

SW Platform Reuse > 70% of total SW

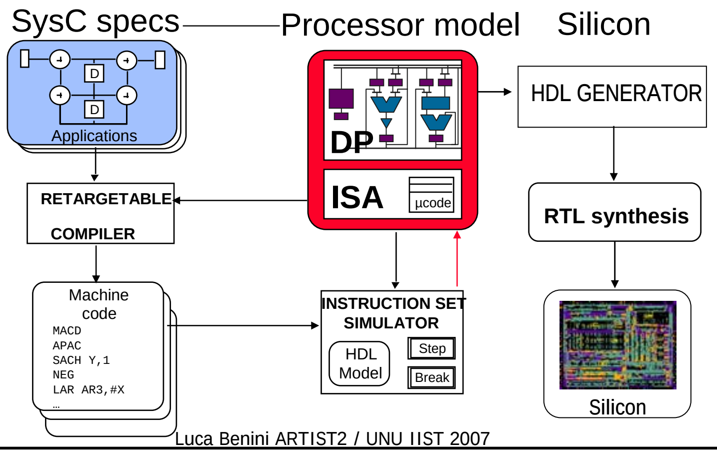
Architecture trends

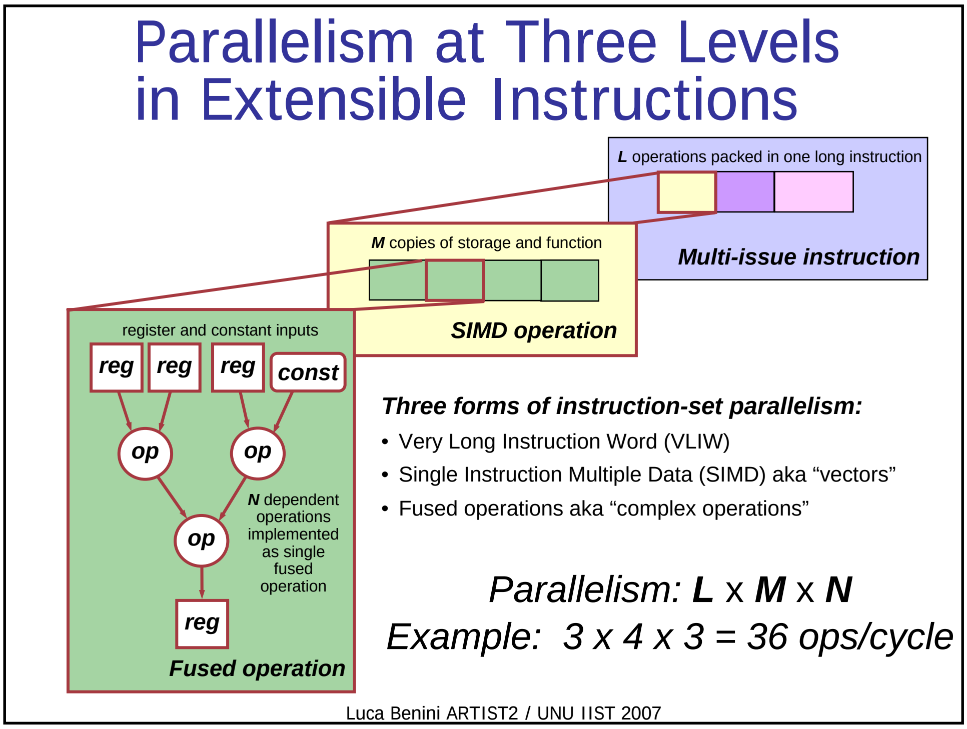

High performance for narrow application field
Dedicated hardware
Special Purpose processor
DSP
Stream processor
Graphic processor
Network processor
Multiple Cores
Heterogeneous Multiprocessor
Programmable Hardware
FPGA、Reconfigurable systems
Dynamically Reconfigurable Processors
Configurable Processor
Tile Processor
Special instructions
Homogeneous Chip-multiprocessor
General purpose CPU
Multiple Cores
High performance for wide application field

Task Specific (configurable) Processors

RWTH AACHEN ⇒ Lisatek(CoWare);
IMEC ⇒Target Compiler T, ARM OptimoDE
PHILIPS ⇒ Siliconhive; TENSILICA, PicoChip…

Courtesy: Target Compilers T





Parallelism at Three Levels in Extensible Instructions

L operations packed in one long instruction

Multi-issue instruction

M copies of storage and function

SIMD operation

register and constant inputs

reg *reg* *reg* *const*

op *op*

op

reg

N dependent operations implemented as single fused operation

Fused operation

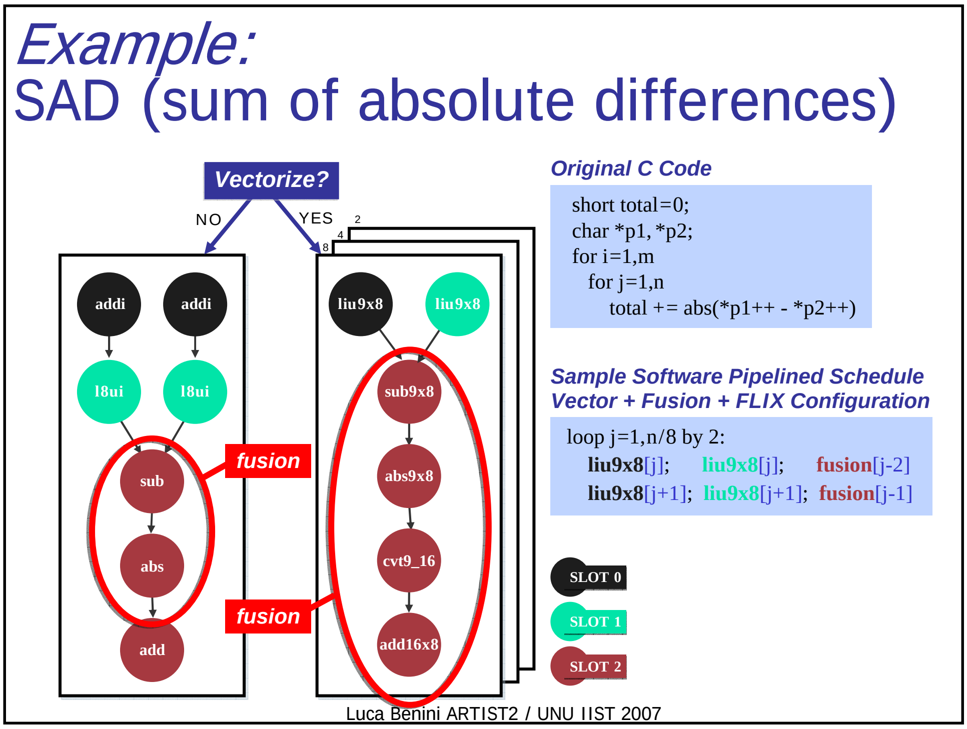
Three forms of instruction-set parallelism:

- Very Long Instruction Word (VLIW)
- Single Instruction Multiple Data (SIMD) aka “vectors”
- Fused operations aka “complex operations”

Parallelism: ***L*** x ***M*** x ***N***

Example: 3 x 4 x 3 = 36 ops/cycle

Luca Benini ARTIST2 / UNU IIST 2007



Example: SAD (sum of absolute differences)

Vectorize?

NO YES

2 4 8

addi addi

l8ui l8ui

sub

abs

add

fusion

fusion

liu9x8 liu9x8

sub9x8

abs9x8

cvt9_16

add16x8

Original C Code

```
short total=0;
char *p1, *p2;
for i=1,m
  for j=1,n
     total += abs(*p1++ - *p2++)
```

Sample Software Pipelined Schedule Vector + Fusion + FLIX Configuration

```
loop j=1,n/8 by 2:
   liu9x8[j];      liu9x8[j];      fusion[j-2]
   liu9x8[j+1];  liu9x8[j+1];  fusion[j-1]
```

● SLOT 0

● SLOT 1

● SLOT 2

Luca Benini ARTIST2 / UNU IIST 2007

Dynamically Reconfigurable Processors

- Reconfigurable systems → Previous lesson
  - Flexible but It takes 10’s milliseconds for dynamic reconfiguration.
- Dynamically Reconfigurable Processors
  - Improves area efficiency by changing hardware structure.
  - IPs used in various SoCs.
  - History
    - Reconfigurable Co-processor Garp(1997), CHIMAERA(2000)
    - Multicontext reconfigurable devices WASMII(1992),Time-multiplexing FPGA(1997), PipeRench(1998), DRL(1998)
    - Functional-level synthesis
  - Various commercial products are available since 2000
    - IPFlex DAPDNA-2, NEC electronics DRP-1, PACT Xpp, Elixent DFabrix
  - SONY’s VME(Virtual Mobile Engine) is embedded in Network Workman and PSP
  - Recently, many Japanese vendors start to develop commercial products
    - Fujitsu
    - Hitachi
    - Lucent
    - Sanyo
    - Toshiba（Mep+D-Fabrix)

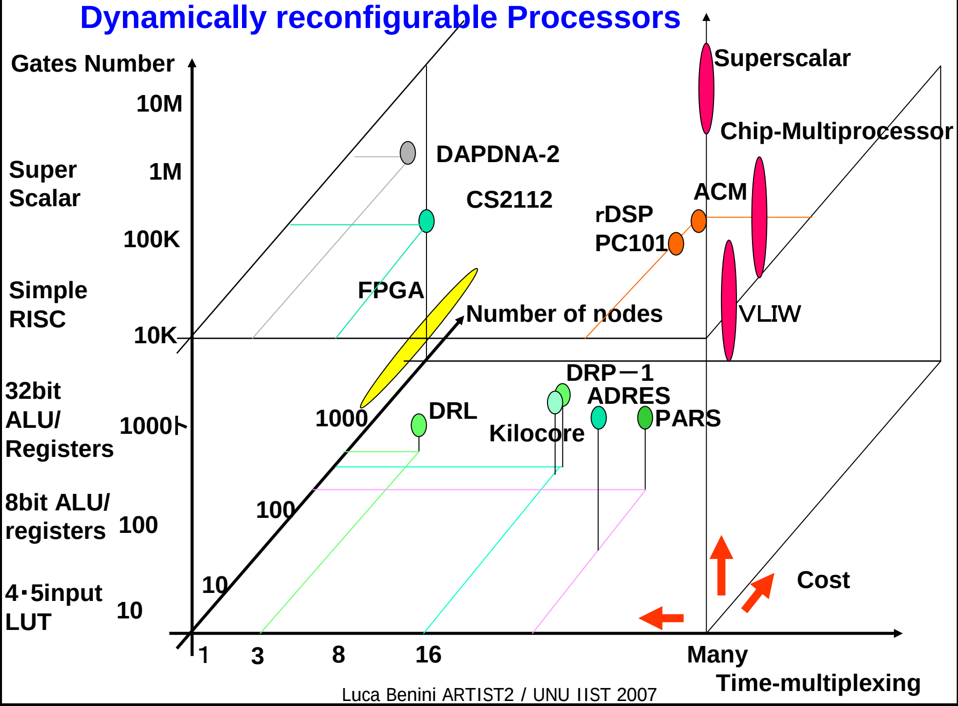
Processing Element

- Specialized for media/stream processing

Coarse grain ⇔ Fine grain: LUT of FPGAs

- Components
  - ALU
  - Shifter＋Mask unit
  - Multiplexers
  - Registers
- Operations and interconnection between components are changeable
- No instruction fetch mechanism : A part of large datapath

Dynamically reconfigurable Processors



Putting it all together

	2004	2006	2008	2010	2012
Technology Node (nm)	90	65	45	32	22
Loosely coupled Sub-Systems	2	4	6	8	12
General Purpose CPU	Single → Multiple				
Hardware Accelerator	Hardwired → Reconfigurable				

- Constant SoC Die Size
- Slow evolution of peripherals (area decrease)
- GP CPU sub system complexity 2x each node (constant area),
- Embedded Memory capacity 2x at each node (constant area)
- Loosely coupled DSP sub system complexity increase by 30% at each node (30% area decrease)

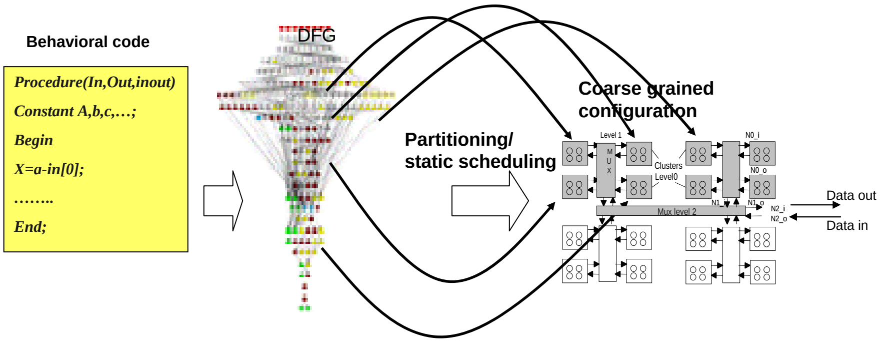
Main trends

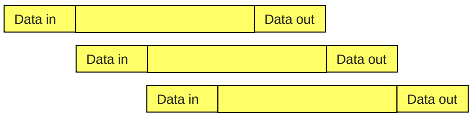
- Host CPU evolving toward multi-core architecture to meet the performance increase requirements
- HW acceleration mapped on reconfigurable arrays
  - Performances close to dedicated HW in many areas
  - Good fit with regular design constraints imposed by 45nm process and beyond
  - Excellent structure for best optimized power management
  - And ... FLEXIBILITY ...

Reconfigurable HW (DSP fabric)

- Target signal processing and arithmetic intensive applications
- Reconfigurable array of simple DSP core (CNode)
- Low power architecture
  - Hierarchical clock gating
  - Distributed leakage control (fine grain power gating)
- Programmable DMA engine
- Reconfigurable at run time, multi task

Mapping Flow





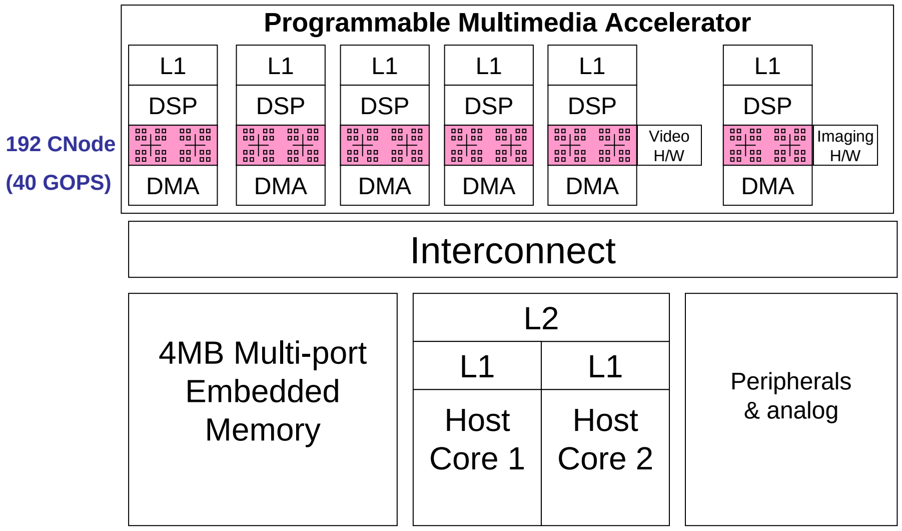
ILP + software pipelining

- Alus execute a cyclic micro-sequence
- Data exchanges through hierarchical clustered interconnect
- Configuration step is sequence loading and interconnect programming

Mapping Flow

- 3D optimization problem (place/route/schedule)
- Traditional scheduling techniques for VLIW or clustered VLIW don’t apply
  - The solution don’t take into account the spatial dimension of the problem
- Traditional P&R used in FPGA don't apply neither because they don't consider the time dimension

What can fit in 45mm² in 45nm



Programmable Multimedia Accelerator

192 CNode

(40 GOPS)

L1 | DSP | DMA (×6)

Video H/W

Imaging H/W

Interconnect

4MB Multi-port Embedded Memory

L2

L1 | L1

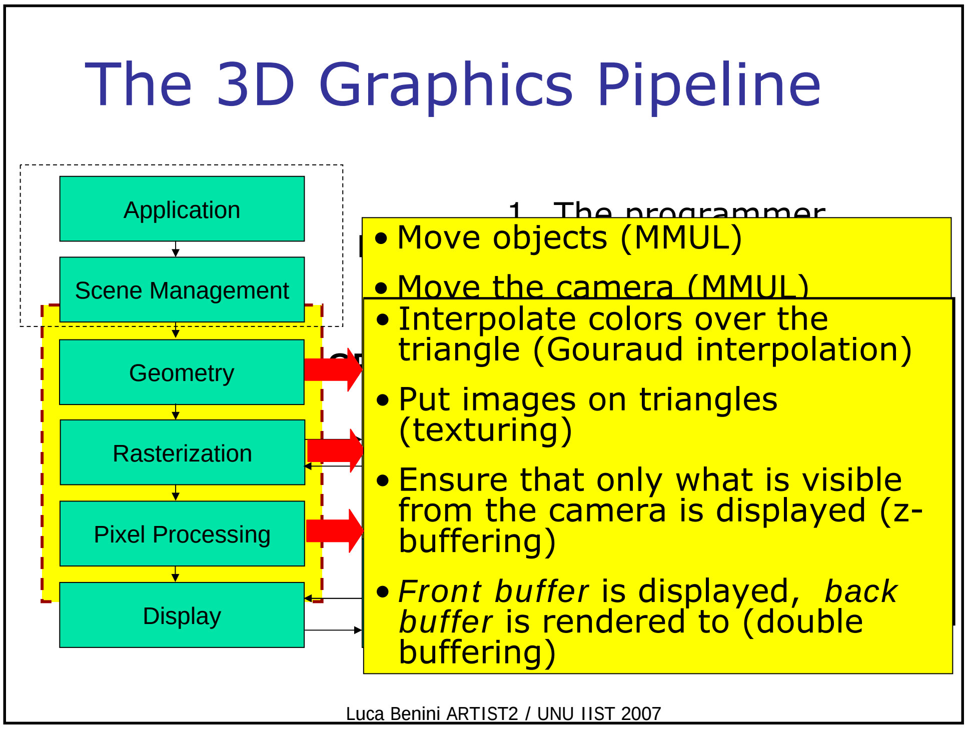
Host Core 1 | Host Core 2

Peripherals & analog

Case Study: GPUs

Mobile graphics platforms

300-400 million mobile phones with graphics hardware (OpenGL ES) by 2009



The 3D Graphics Pipeline

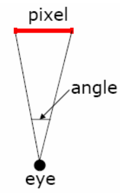
1. The programmer

- Move objects (MMUL)
- Move the camera (MMUL)

- Interpolate colors over the triangle (Gouraud interpolation)
- Put images on triangles (texturing)
- Ensure that only what is visible from the camera is displayed (z-buffering)
- *Front buffer* is displayed, *back buffer* is rendered to (double buffering)

Why is it hard with 3D graphics on mobile devices?

- Small amount of memory
- Limited instruction set
- Low clock frequency
  - 100-200 MHz ARM9–400-600 MHz ARM11
- Small area on the chip for CG
- Must be cheap and physically small
- Powered by batteries!
  - A memory access is one of the most expensive operations
  - Battery growth: 9% per year
  - Performance growth: 40% per year
- Small display, but very close to the eye
  - Avg. Eye-to-pixel angle 1-4x larger than for desktop



Limited resources, but high quality rendering!

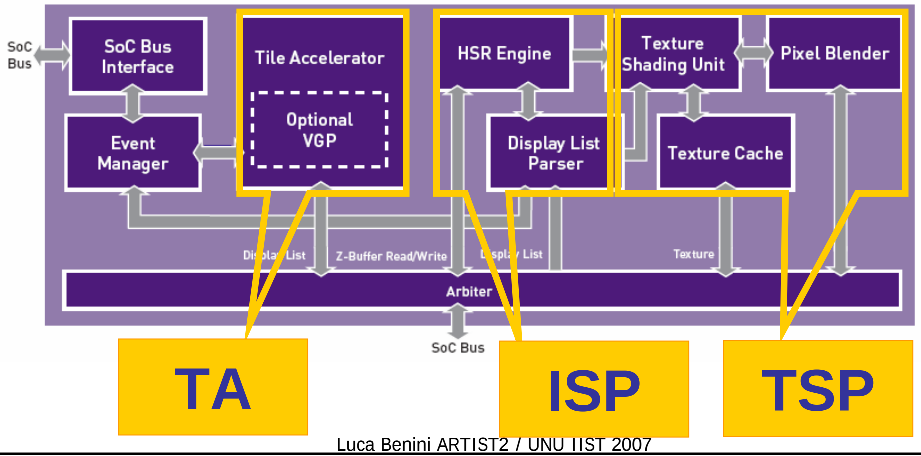
PowerVR MBX low-power GPU

Architecture

- Tile accelerator
- Image synthesis processor
- Texture and shading processor

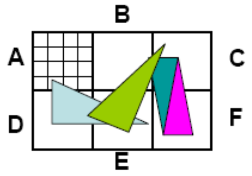
Features

- Tile-based rendering
- ITC™: PowerVR internal true color: color ops on-chip at 32-bpp
- FSAA4Free™: full screen anti-aliasing for realism at mobile display resolutions
- PVR-TC™: texture compression for small memory footprints.

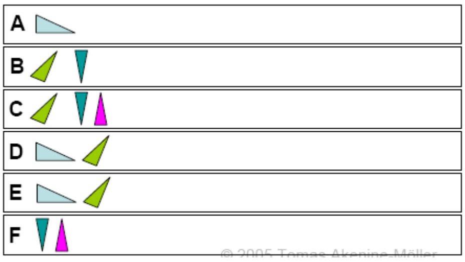


Tiled (sort middle) architecture

- Apply geometry transf. (incl. projection) to vertices

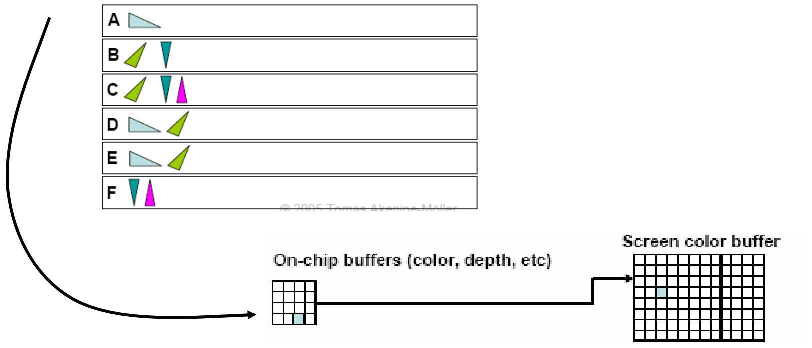


- Create a triangle list for each tile
  - Holds pointers to all triangles overlapping a tile

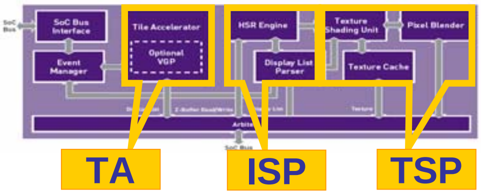


Tiled processing

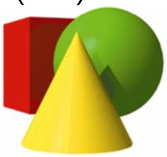
- Process one tile at a time, and rasterize triangles in list
- Work on local (on-chip) tile buffers
  - Color, depth, stencil
- Copy color tile buffer to off-chip display buffer
  - may need to copy depth buffer as well

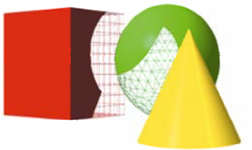

A
B
C
D
E
F
© 2005 Tomas Akenine-Möller
On-chip buffers (color, depth, etc)
Screen color buffer

P, TSP



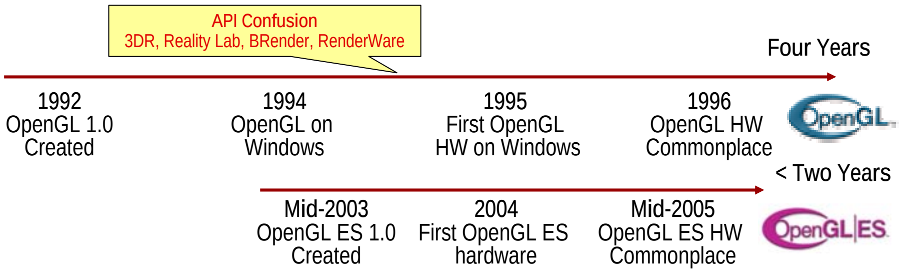
- CPU sends triangle data to MBX
- **Tile Accelerator** (TA): sorts triangles, and creates a list of triangle pointers for each tile
  - Needs an entire scene before ISP and TSP blocks can start
  - So TA works on the next image, while ISP and TSP work on the current image (i.e., they work in a pipelined fashion)
- **Image synthesis processor** (ISP): implements Z-buffer, color buffer, stencil buffer for tile
  - Depth testing: test 32 pixels at a time against Z-buffer
    - Records which pixels are visible
    - Groups pixels with same texture and sends to TSP
    - These are guaranteed to be visible, so we only texture each pixel once (deferred texturing)
- **Texture and Shading Processor** (TSP): Handles texturing and shading interpolation
  - Uses texture compression
  - Performs over-sampling





Mobile 3D API

- The Mobile 3D industry is embryonic - and moving fast!
- **We are where PC graphics were in 1996 - but evolving 2-3 times faster!**
  - Just nine months since OpenGL ES 1.0 released
  - Compliant graphics acceleration already on the market
- OpenGL ES has become the industry standard for embedded graphics
  - We avoided two years of API indecision that occurred on the PC



API Confusion
3DR, Reality Lab, BRender, RenderWare

Four Years

1992
OpenGL 1.0
Created

1994
OpenGL on
Windows

1995
First OpenGL
HW on Windows

1996
OpenGL HW
Commonplace

< Two Years

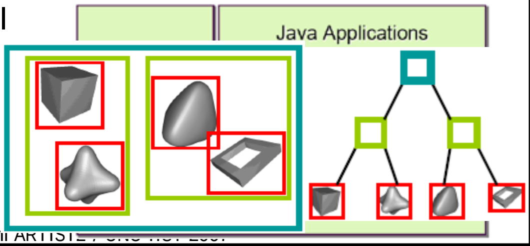
Mid-2003
OpenGL ES 1.0
Created

2004
First OpenGL ES
hardware

Mid-2005
OpenGL ES HW
Commonplace

The case for a higher abstraction

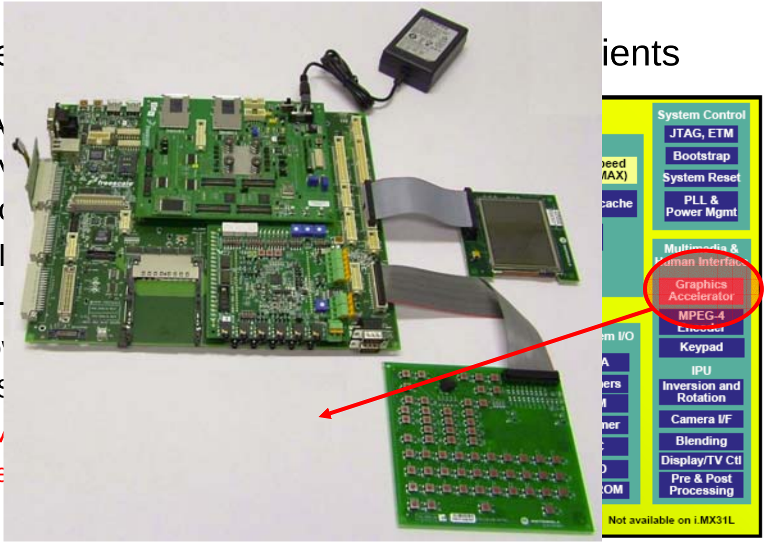
- A game is much more than just 3D rendering
  - Objects, properties, relations (scene graph)
  - Key frame and other animations
  - Etc. (game logic, sounds, …)
- If everything else but rendering is in Java
  - A very large percentage of the processing is in slow Java
  - Even if rendering was 100% in HW, total acceleration remains limited
- A higher level API could help
  - More of the functionality could be implemented in native (=faster) code
  - Only the game logic must remain in Java
- M3G (JSR-184), a new API
  - Nodes and scene graph
  - Extensive animation support
  - Binary file format and loader



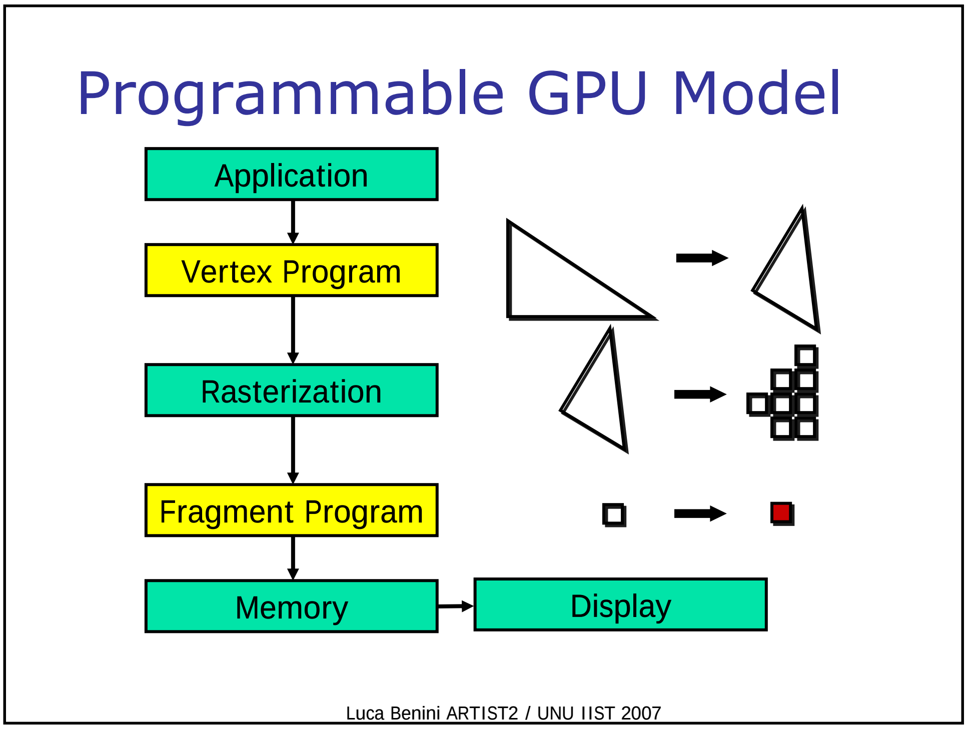
Luca Benini ARTIST2 / UNU IIST 2007

Freescale iMX31

- Syste ients
- CPU: A
- VFP – V Co-proc
- Image I
- MPEG-
- HW Po
  - DVFS
- GPU: M
  - Powe



Luca Benini ARTIST2 / UNU IIST 2007



Programmable GPU Model

Application

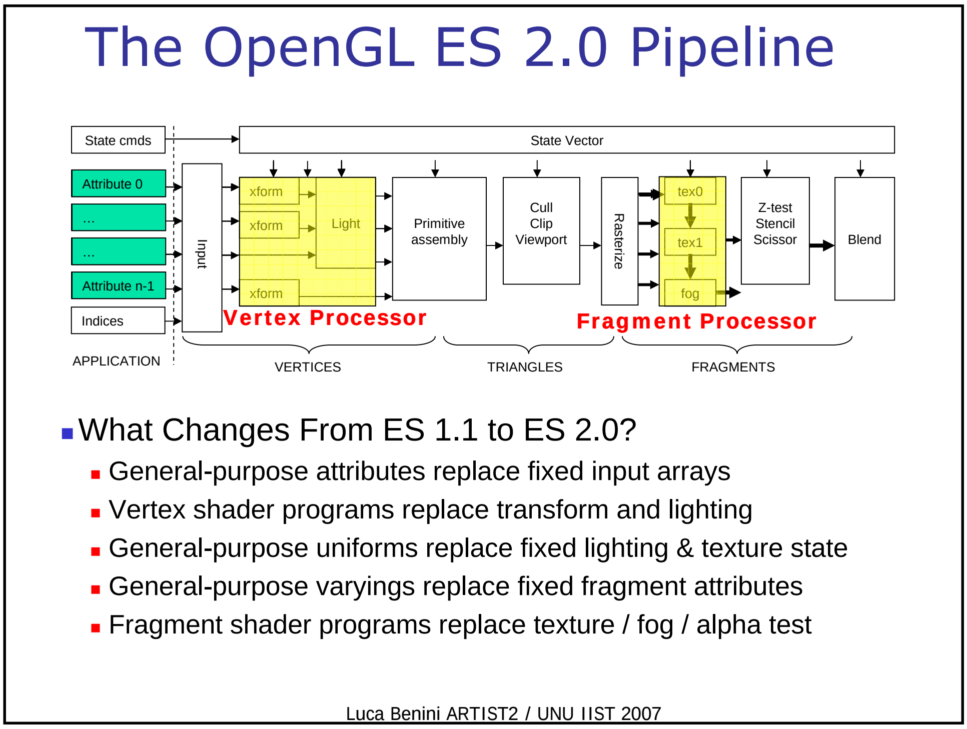
Vertex Program

Rasterization

Fragment Program

Memory

Display



The OpenGL ES 2.0 Pipeline

State cmds

State Vector

Attribute 0

…

…

Attribute n-1

Indices

APPLICATION

Input

xform

xform

Light

xform

Vertex Processor

Primitive assembly

Cull
Clip
Viewport

Rasterize

tex0

tex1

fog

Fragment Processor

Z-test
Stencil
Scissor

Blend

VERTICES

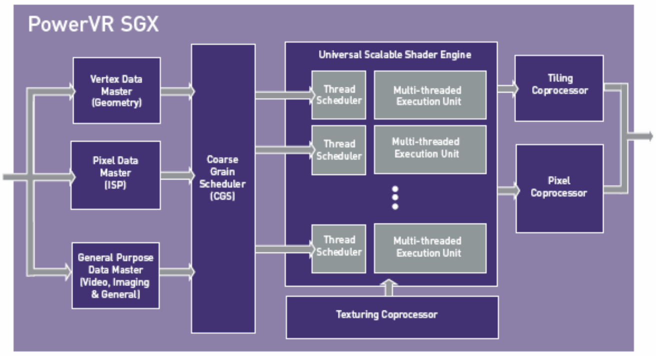
TRIANGLES

FRAGMENTS

- What Changes From ES 1.1 to ES 2.0?
  - General-purpose attributes replace fixed input arrays
  - Vertex shader programs replace transform and lighting
  - General-purpose uniforms replace fixed lighting & texture state
  - General-purpose varyings replace fixed fragment attributes
  - Fragment shader programs replace texture / fog / alpha test

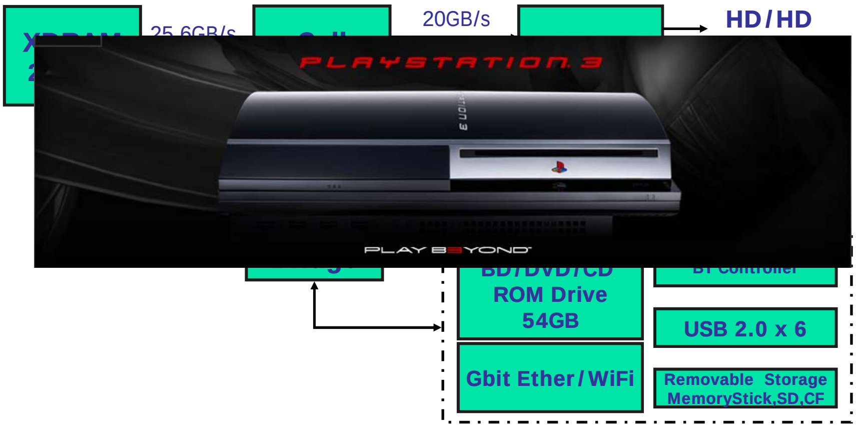
PowerVR SGX

- Advanced shader-based GPU (OpenGL ES 2.0 compliant)
  - USSE: scalable programmable, multi-threaded engine for graphics, video, imaging and other mathematically-intensive tasks.
    - Tasks are automatically broken down into processing packets which are then scheduled across a number of multi-threaded execution units
    - Coprocessors (texture, pixel and tiling accelerators) assist the MT EUs
  - Latency tolerant architecture
    - geometry and rasterisation are decoupled using tile-based rendering, enabling on-chip processing hidden-surface removal and deferred pixel shading


PowerVR SGX
Vertex Data Master (Geometry)
Pixel Data Master (ISP)
General Purpose Data Master (Video, Imaging & General)
Coarse Grain Scheduler (CGS)
Universal Scalable Shader Engine
Thread Scheduler
Multi-threaded Execution Unit
Thread Scheduler
Multi-threaded Execution Unit
Thread Scheduler
Multi-threaded Execution Unit
Texturing Coprocessor
Tiling Coprocessor
Pixel Coprocessor

A high-end system: PS3

A look into the future…


XDRAM 2

25.6GB/s

Cell

20GB/s

HD/HD

BD/DVD/CD ROM Drive 54GB

BT controller

USB 2.0 x 6

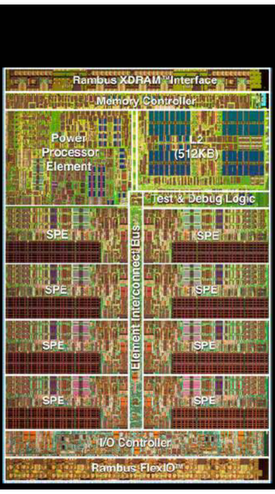
Gbit Ether/WiFi

Removable Storage MemoryStick,SD,CF

Cell: Single-Chip Embedded Multiprocessor

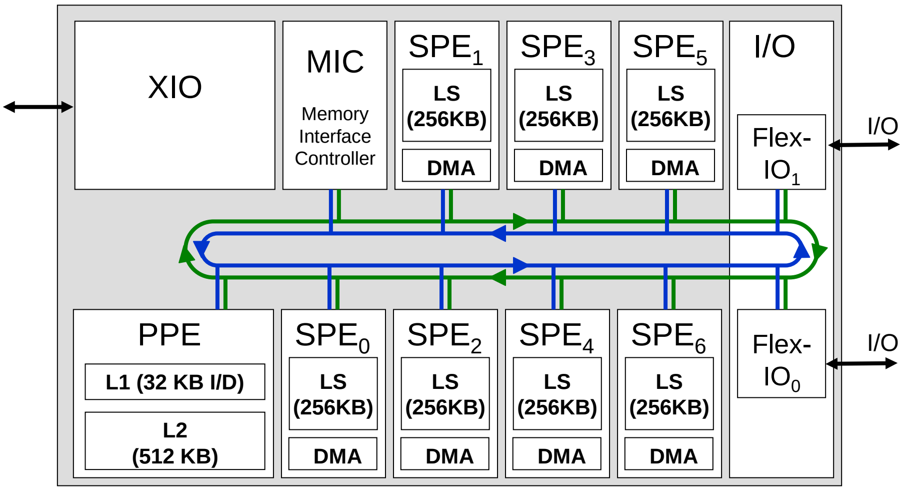
Highlights (3.2 GHz)

- 241M transistors
- 235mm2
- 9 cores, 10 threads
- >200 GFlops (SP)
- >20 GFlops (DP)
- Up to 25 GB/s memory B/W
- Up to 75 GB/s I/O B/W
- >300 GB/s EIB
- Top frequency >4GHz
  (observed in lab)



SONY COMPUTER ENTERTAINMENT IBM Toshiba

Cell Architecture



XIO

MIC

Memory Interface Controller

SPE$_1$ LS (256KB) DMA

SPE$_3$ LS (256KB) DMA

SPE$_5$ LS (256KB) DMA

I/O

Flex-IO$_1$

I/O

PPE

L1 (32 KB I/D)

L2 (512 KB)

SPE$_0$ LS (256KB) DMA

SPE$_2$ LS (256KB) DMA

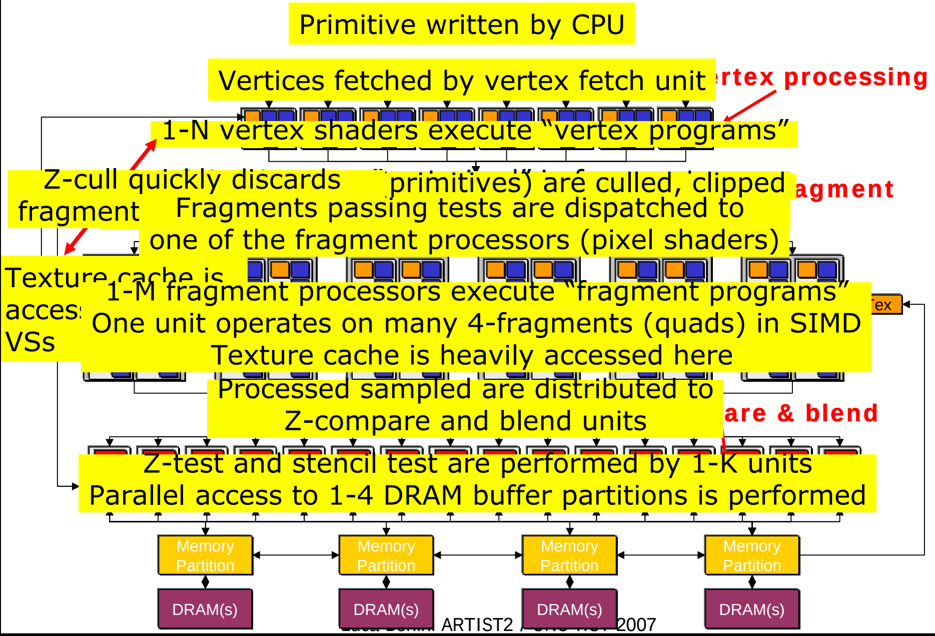
SPE$_4$ LS (256KB) DMA

SPE$_6$ LS (256KB) DMA

Flex-IO$_0$

I/O

NVidia GeForce 7800 Architecture



Primitive written by CPU

Vertices fetched by vertex fetch unit

1-N vertex shaders execute "vertex programs"

Z-cull quickly discards fragment

primitives) are culled, clipped

Fragments passing tests are dispatched to one of the fragment processors (pixel shaders)

Texture cache is acces VSs

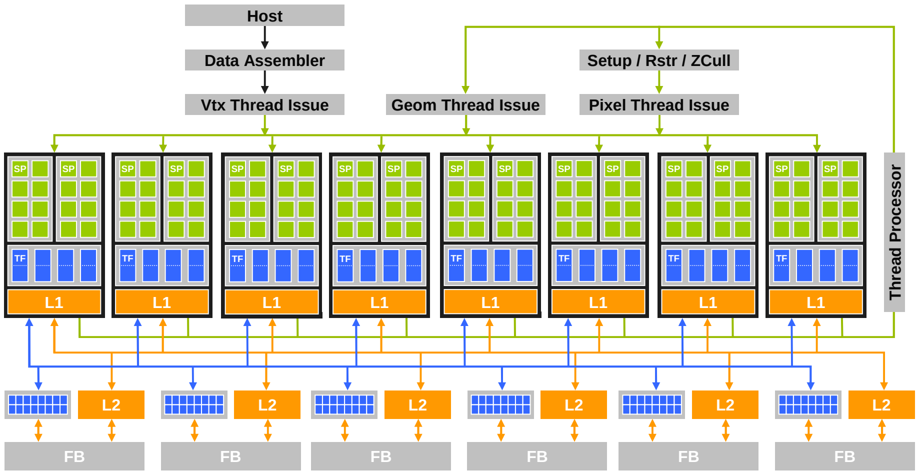
1-M fragment processors execute "fragment programs"
One unit operates on many 4-fragments (quads) in SIMD
Texture cache is heavily accessed here

Processed sampled are distributed to Z-compare and blend units

Z-test and stencil test are performed by 1-K units
Parallel access to 1-4 DRAM buffer partitions is performed

GF8800 replaces the pipeline model

- The future of GPUs is programmable processing
- So – build the architecture around the processor



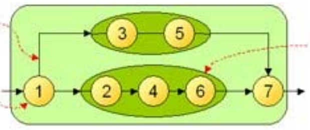
Case study: Polycores & NoCs

Embedded SoC Architecture Trends

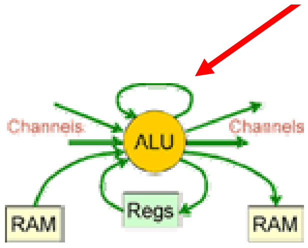
“Distributed” stream processors

Asynchronous channel

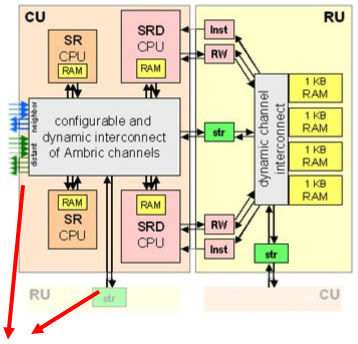
Computation kernel



Composite kernel



Microarchitecture



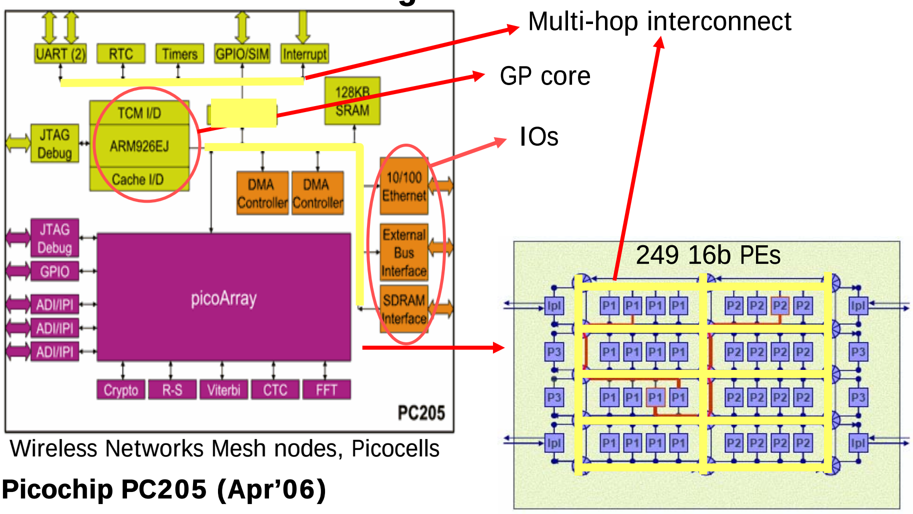
Communication channels

Ambric AM2045 (Oct’06)

- 117MTxn (0.13CMOS)
- 360 PE & 4.06Mb of SRAM
- 1.08 TOPS @ 333MHz (peak)
- 14 Watts → 77GOPS/Watt

Embedded SoC Architecture Trends

Heterogeneous clusters



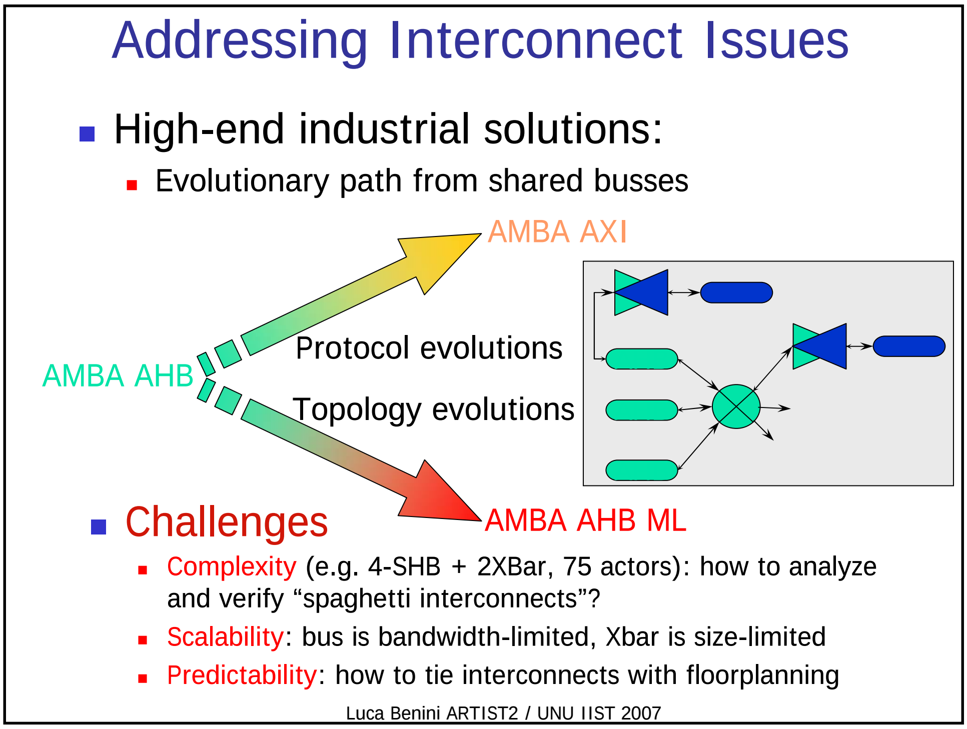
Wireless Networks Mesh nodes, Picocells

Picochip PC205 (Apr'06)

- 260MHz, 31GMAC/s, 160GIP/s
- 64KB I,D$, 128KB SRAM
- Less than 5 W, less than 1$/GMAC

Vision: What Do We Need?

- Scalable
  - Don't want to change the way I design architecture even if requirements scale up exponentially
- Predictable
  - I want to know what to expect (latency, bandwidth), and I want to be able to negotiate it
- Robust
  - Keeps going and going… Even if something is broken inside
- Efficient
  - Silicon is expensive, power is precious
- Easy
  - To create, update, analyze, verify



Addressing Interconnect Issues

- High-end industrial solutions:
  - Evolutionary path from shared busses

AMBA AXI

AMBA AHB

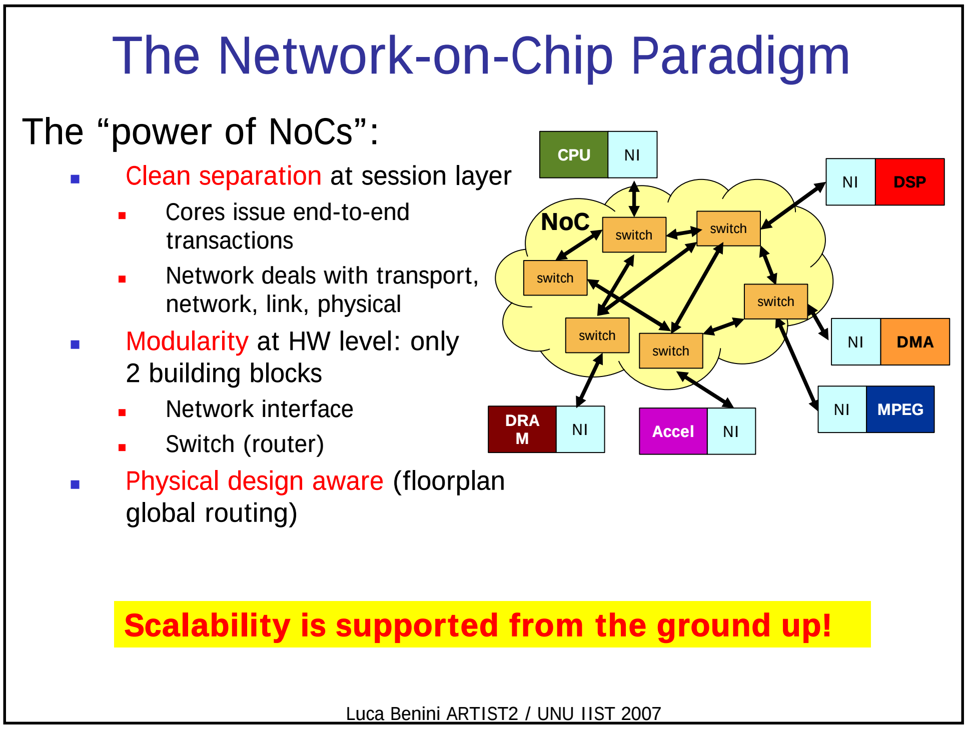
Protocol evolutions

Topology evolutions

AMBA AHB ML

- Challenges
  - Complexity (e.g. 4-SHB + 2XBar, 75 actors): how to analyze and verify “spaghetti interconnects”?
  - Scalability: bus is bandwidth-limited, Xbar is size-limited
  - Predictability: how to tie interconnects with floorplanning

Luca Benini ARTIST2 / UNU IIST 2007



The Network-on-Chip Paradigm

The “power of NoCs”:

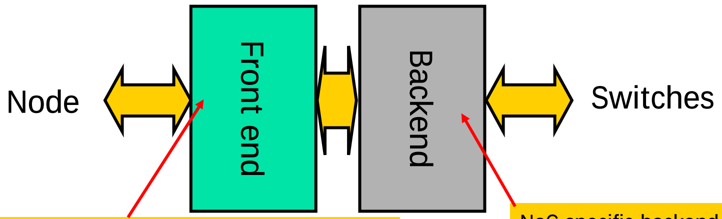
- Clean separation at session layer
  - Cores issue end-to-end transactions
  - Network deals with transport, network, link, physical
- Modularity at HW level: only 2 building blocks
  - Network interface
  - Switch (router)
- Physical design aware (floorplan global routing)

Scalability is supported from the ground up!

Luca Benini ARTIST2 / UNU IIST 2007

Building blocks: NI

- Session-layer interface with nodes
- Back-end manages interface with switches



Standardized node interface @ session layer.
Initiator vs. target distinction is blurred

1. Supported transactions (e.g. QoSread...)
2. Degree of parallelism
3. Session prot. control flow & negotiation

NoC specific backend (layers 1-4)

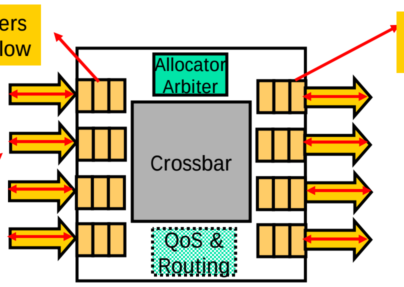
1. Physical channel interface
2. Link-level protocol
3. Network-layer (packetization)
4. Transport layer (routing)

Building blocks: Switch

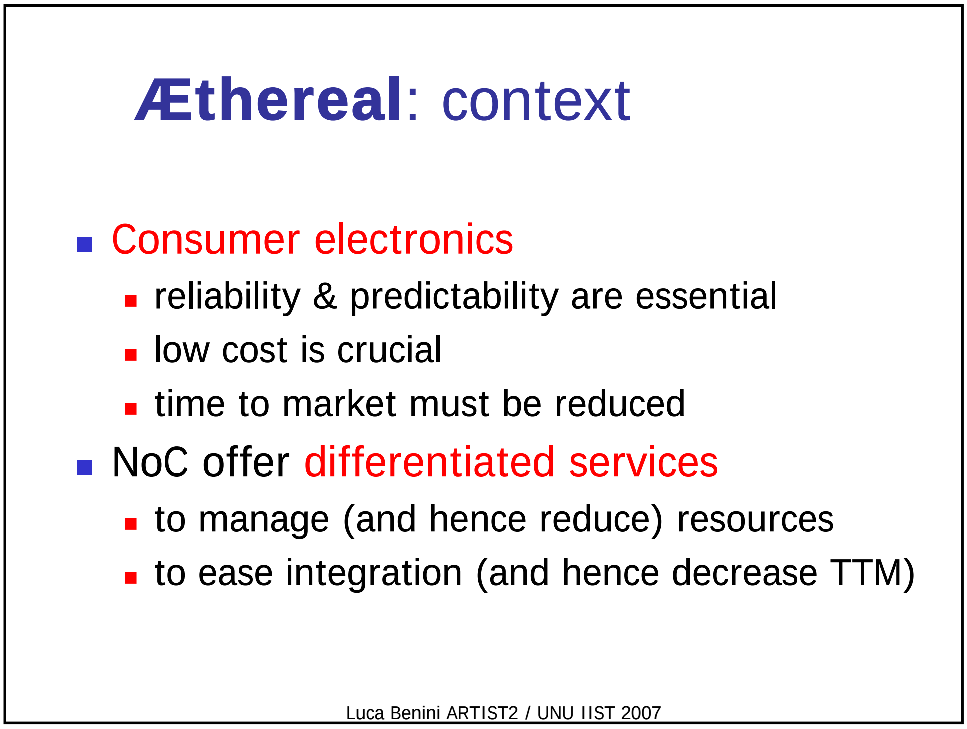
- Router: receives and forwards packets
  - NOTE: Packet-based does not mean datagram!
- Level 3 or Level 4 routing
  - No consensus, but generally L4 support is limited (e.g. simple routing)

Input buffers & control flow

Output buffers & control flow

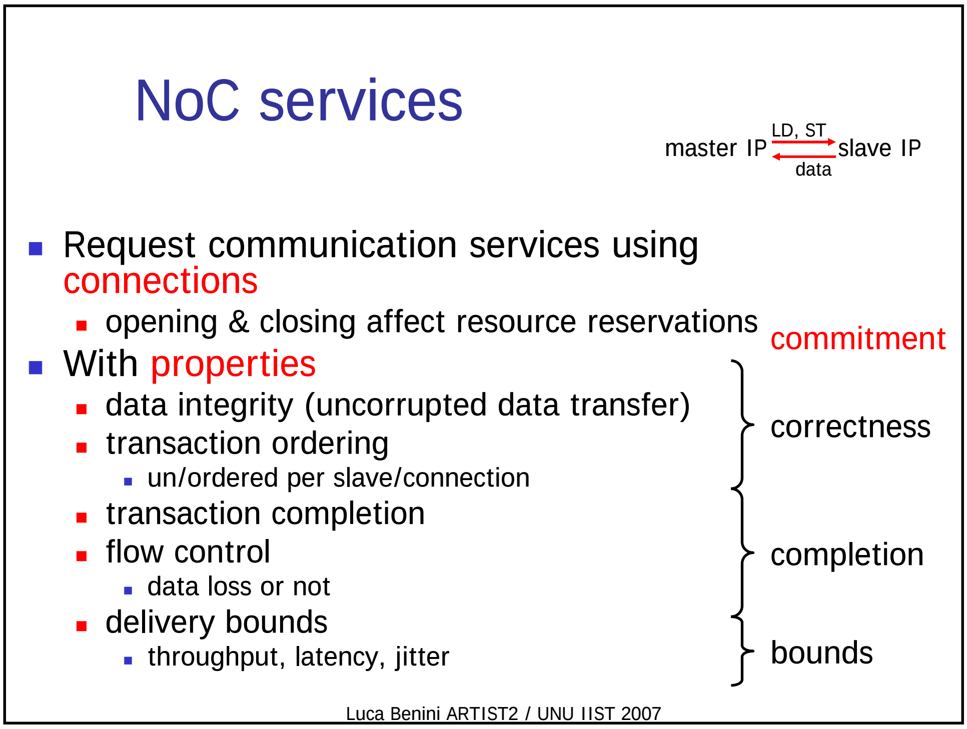


Data ports
with control flow
wires



Æthereal: context

- Consumer electronics
  - reliability & predictability are essential
  - low cost is crucial
  - time to market must be reduced
- NoC offer differentiated services
  - to manage (and hence reduce) resources
  - to ease integration (and hence decrease TTM)



NoC services

master IP $\underset{\text{data}}{\overset{\text{LD, ST}}{\rightleftarrows}}$ slave IP

- Request communication services using connections
  - opening & closing affect resource reservations
- With properties
  - data integrity (uncorrupted data transfer)
  - transaction ordering
    - un/ordered per slave/connection
  - transaction completion
  - flow control
    - data loss or not
  - delivery bounds
    - throughput, latency, jitter

commitment

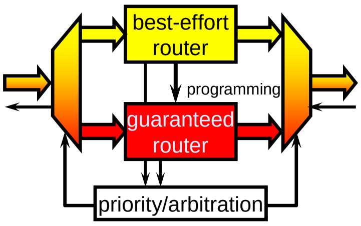
correctness

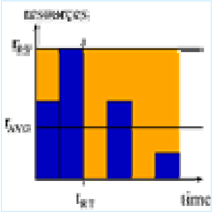
completion

bounds

Æthereal: features

- Conceptually, two disjoint networks
  - a network with throughput+latency guarantees (GT)
  - a network without those guarantees (best-effort, BE)
- Several types of commitment in the network
  - combine guaranteed worst-case behaviour with good average resource usage





Luca Benini ARTIST2 / UNU IIST 2007

Router architecture

- Best-effort router
  - Worm-hole routing
  - Input queueing
  - Source routing
- Guaranteed throughput router
  - Contention-free routing
    - synchronous, using slot tables
    - time-division multiplexed circuits
  - Store-and-forward routing
  - Headerless packets
    - information is present in slot table

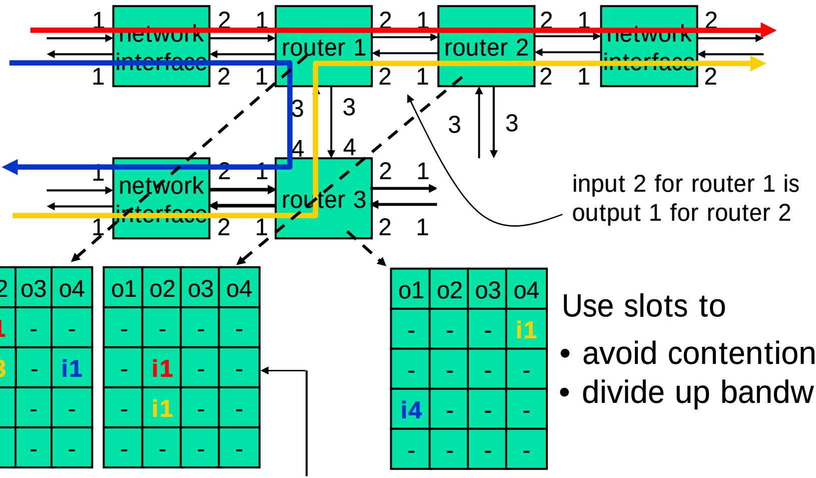
Luca Benini ARTIST2 / UNU IIST 2007

Contention-free routing

- Latency guarantees are easy in circuit switching
- Emulate circuits with packet switching
- Schedule packet injection in network

  such that they never contend for same link at same time
  - in space: disjoint paths
  - in time: time-division multiplexing
  - or a combination

CFR Example



input 2 for router 1 is output 1 for router 2

o1	o2	o3	o4
-	i1	-	-
-	i3	-	i1
-	-	-	-
-	-	-	-

o1	o2	o3	o4
-	-	-	-
-	i1	-	-
-	i1	-	-
-	-	-	-

o1	o2	o3	o4
-	-	-	i1
-	-	-	-
i4	-	-	-
-	-	-	-

the input routed to the output at this slot

Use slots to

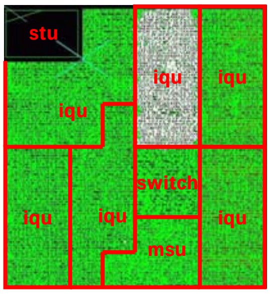
- avoid contention
- divide up bandwidth

CFR setup

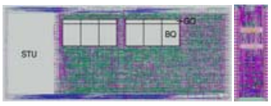
- Use best-effort packets to set up connections
  - set-up & tear-down packets like in ATM (asynchronous transfer mode)
- Distributed, concurrent, pipelined
- Safe: always consistent
- Compute slot assignment compile time, run time, or combination
- Connection opening is guaranteed to complete (but without a latency guarantee) with commitment or rejection

Router implementation

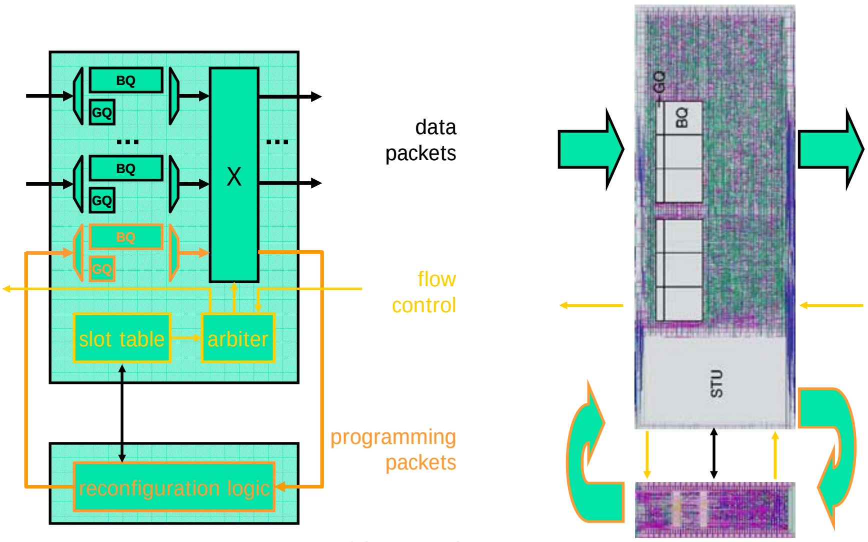
- Memories (for packet storage)
  - Register-based FIFOs are expensive
  - RAM-based FIFOs are as expensive
    - 80% of router is memory
  - Special hardware FIFOs are very useful
    - 20% of router is memory
- Speed of memories
  - registers are fast enough
  - RAMs may be too slow
  - Hardware FIFOs are fast enough



routers based on
register-file and hardware fifos
drawn to approximately
same scale (1mm2, 0.26mm2)

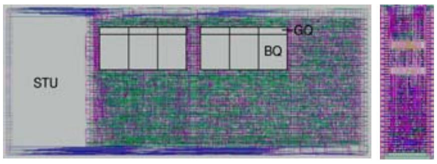


Layout



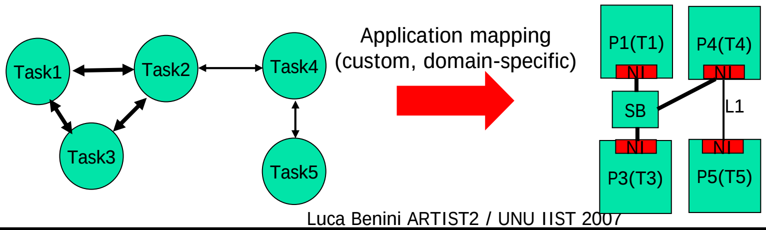
Results

- 5 input and 5 output ports (arity 5)
- 0.25 mm2 CMOS12
- 500 MHz data path, 166 MHz control path
- flit size of 3 words of 32 bits
- 500x32 = 16 Gb/s throughput per link, in each direction
- 256 slots & 5x1 flit fifos for guaranteed throughput traffic
- 6x8 flit fifos for best effort traffic



xpipes: context

- Typical applications targeted by SoCs
  - Complex
  - Highly heterogeneous (component specialization)
  - Communication intensive
- xpipes is a synthesizable, heterogeneous NoC infrastructure
- Three year lifetime, mature research project
  - University of Bologna (architecture)
  - Stanford University (design technology)
  - **University of Cagliari (design and backend)**

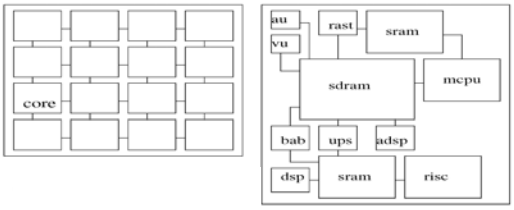


Application mapping (custom, domain-specific)

Heterogeneous topology

SoC *component specialization* leads to the integration of *heterogeneous cores*

Ex. MPEG4 Decoder



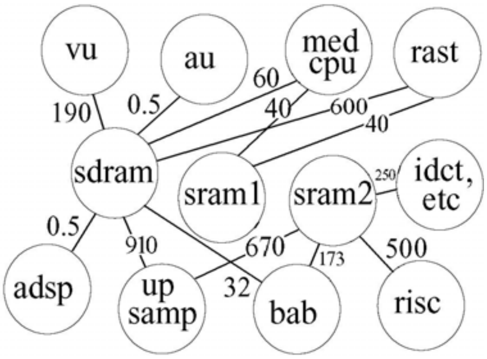
- Non-uniform block sizes
- SDRAM: communication bottleneck
- Many neighboring cores do not communicate

On a homogeneous fabric:

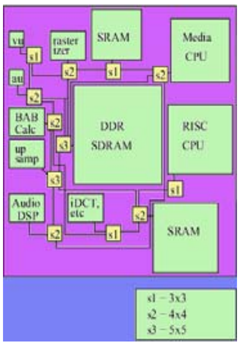
- Risk of under-utilizing many tiles and links
- Risk of localized congestion

Example: MPEG4 decoder

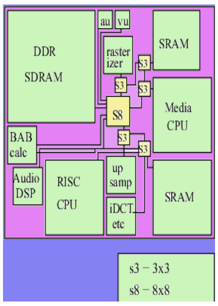
- Core graph representation with annotated average communication requirements



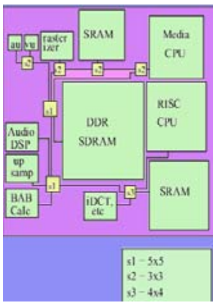
NoC Floorplans



General purpose: mesh

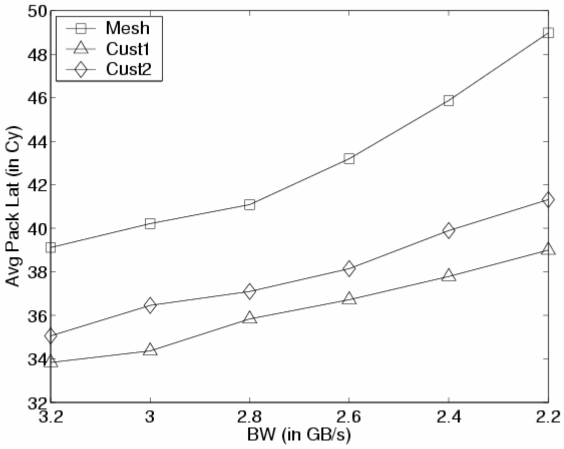


Application Specific NoC1 (centralized)



Application Specific NoC2 (distributed)

Performance, area and power

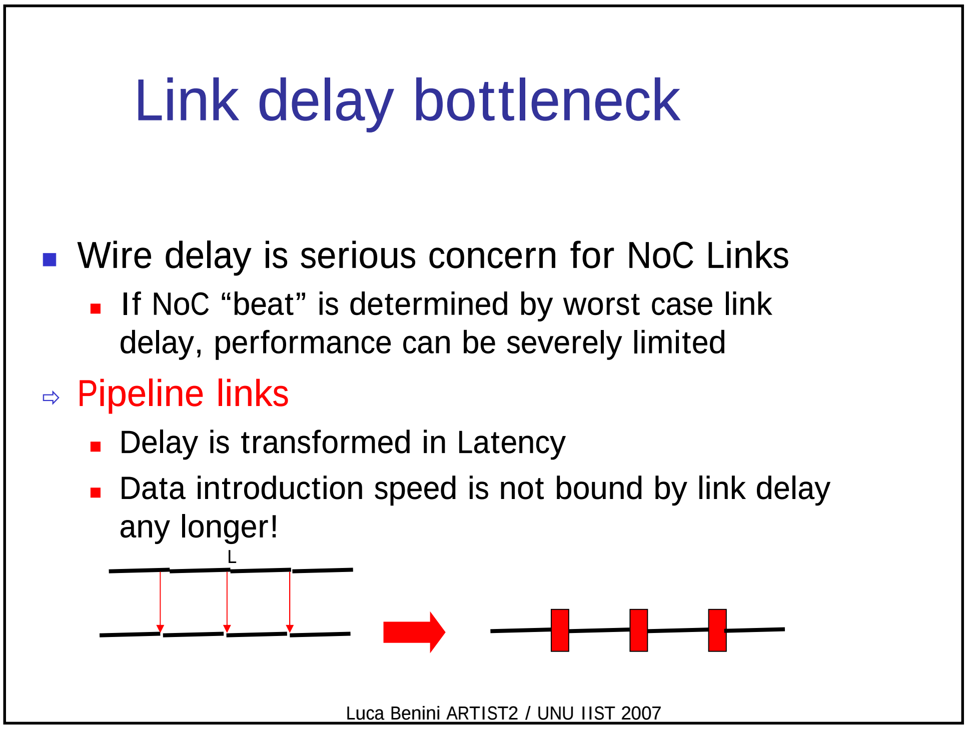


Less latency and better Scalability of custom NoCs

- Relative link utilization (customNoC/meshNoC): 1.5, 1.55
- Relative area (meshNoC/customNoC): 1.52, 1.85
- Relative power (meshNoC/customNoC): 1.03, 1.22

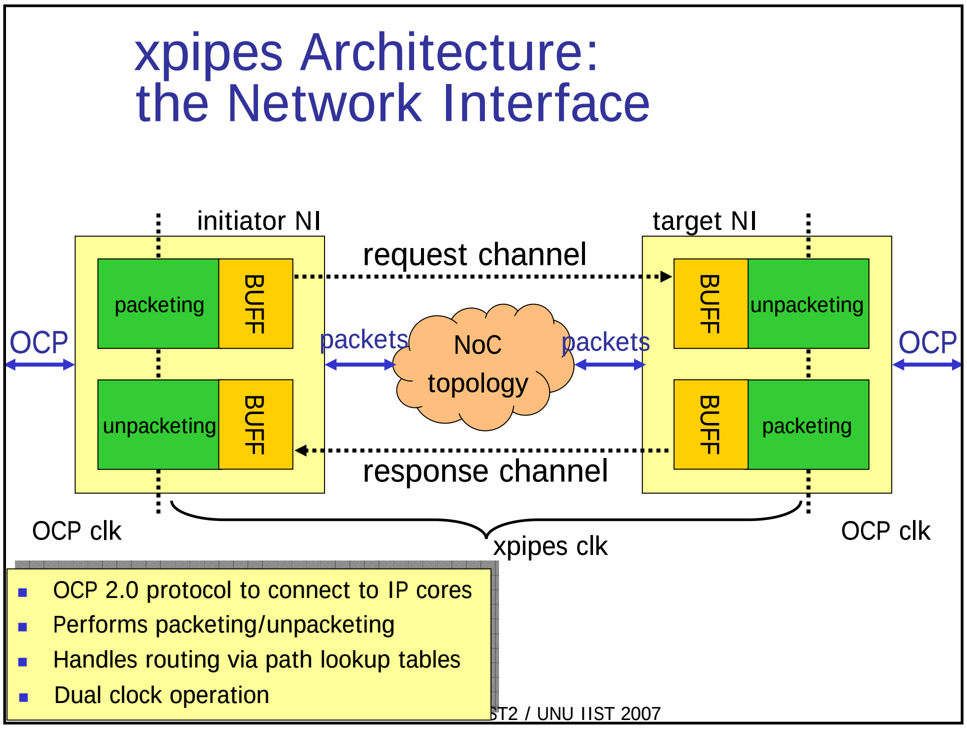
Xpipes: features

- Source based routing
  - Very high performance switch design
- Wormhole switching
  - Minimize buffering area while reducing latency
- Pipelined links
  - Link data introduction interval is not bound by wire delay
  - Link-latency (# of repeater stages) insensitive operation
- Parameterizable network building blocks
  - Plug-and-play composable for arbitrary network topology
  - Design time tunable buffer size, link width, virtual channels, # of switch I/Os
- Standard OCP interface



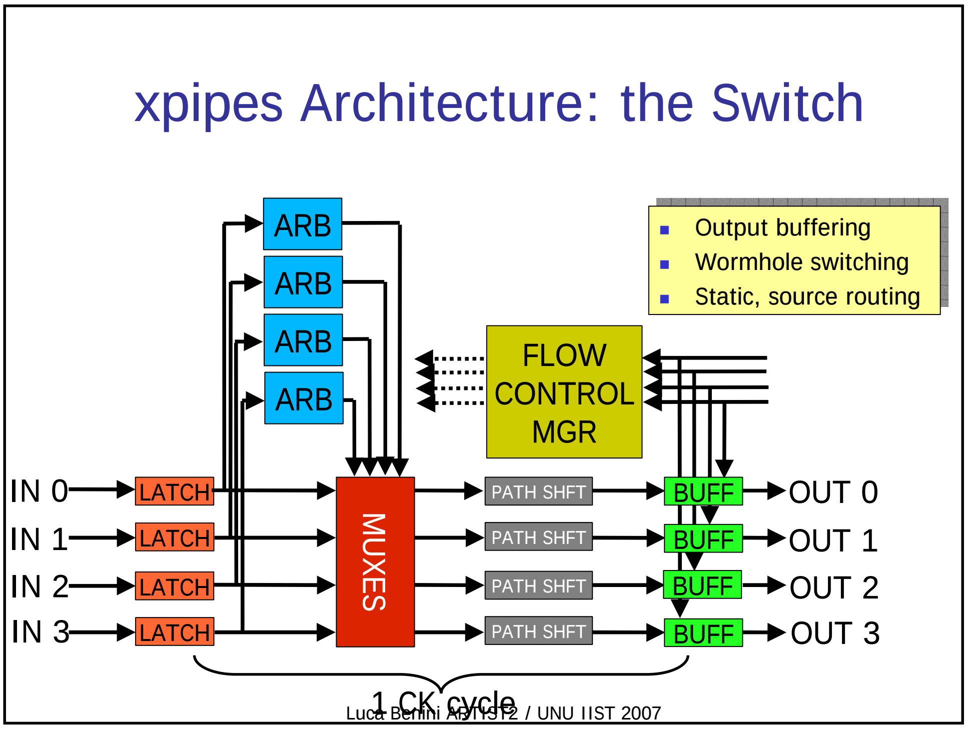
Link delay bottleneck

- Wire delay is serious concern for NoC Links
  - If NoC “beat” is determined by worst case link delay, performance can be severely limited
- ⇨ Pipeline links
  - Delay is transformed in Latency
  - Data introduction speed is not bound by link delay any longer!



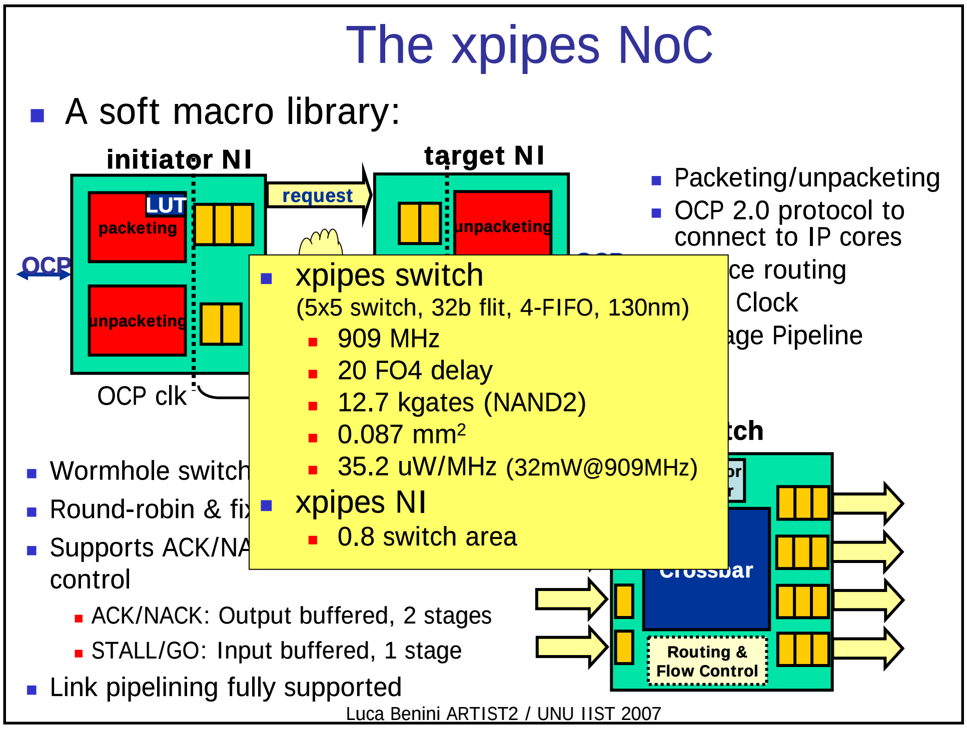
xpipes Architecture: the Network Interface

- OCP 2.0 protocol to connect to IP cores
- Performs packeting/unpacketing
- Handles routing via path lookup tables
- Dual clock operation



xpipes Architecture: the Switch

- Output buffering
- Wormhole switching
- Static, source routing



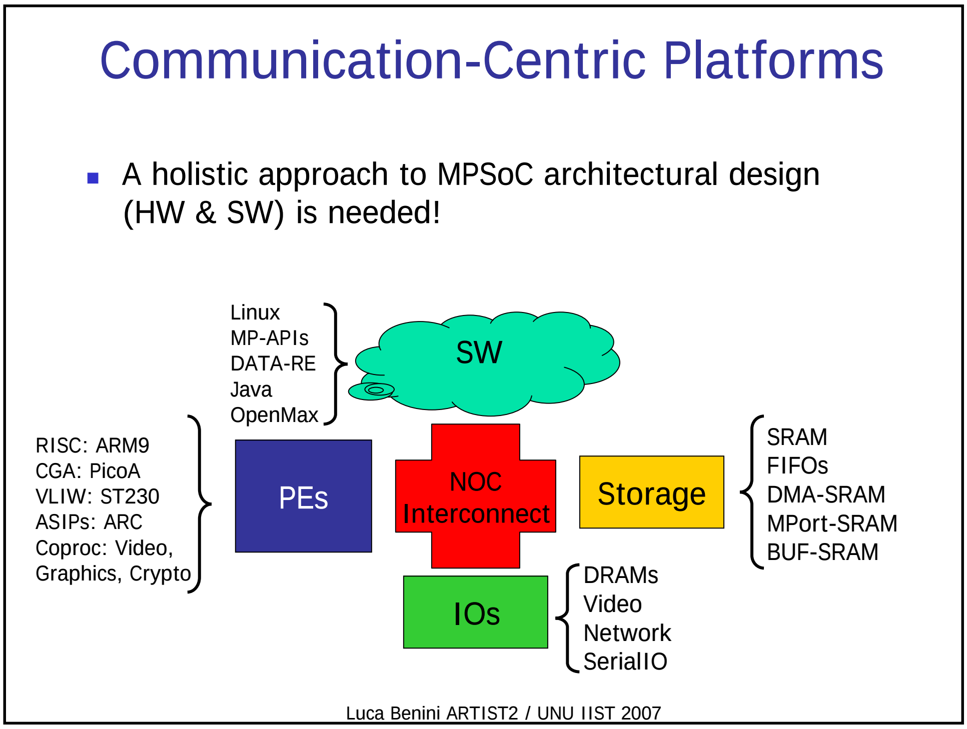
The xpipes NoC

- A soft macro library:

- Packeting/unpacketing
- OCP 2.0 protocol to connect to IP cores
- ce routing
- Clock
- age Pipeline

- xpipes switch
  (5x5 switch, 32b flit, 4-FIFO, 130nm)
  - 909 MHz
  - 20 FO4 delay
  - 12.7 kgates (NAND2)
  - 0.087 mm^2
  - 35.2 uW/MHz (32mW@909MHz)
- xpipes NI
  - 0.8 switch area

- Wormhole switch
- Round-robin & fi
- Supports ACK/NA control
  - ACK/NACK: Output buffered, 2 stages
  - STALL/GO: Input buffered, 1 stage
- Link pipelining fully supported



Communication-Centric Platforms

- A holistic approach to MPSoC architectural design (HW & SW) is needed!

Linux
MP-APIs
DATA-RE
Java
OpenMax

SW

RISC: ARM9
CGA: PicoA
VLIW: ST230
ASIPs: ARC
Coproc: Video, Graphics, Crypto

PEs

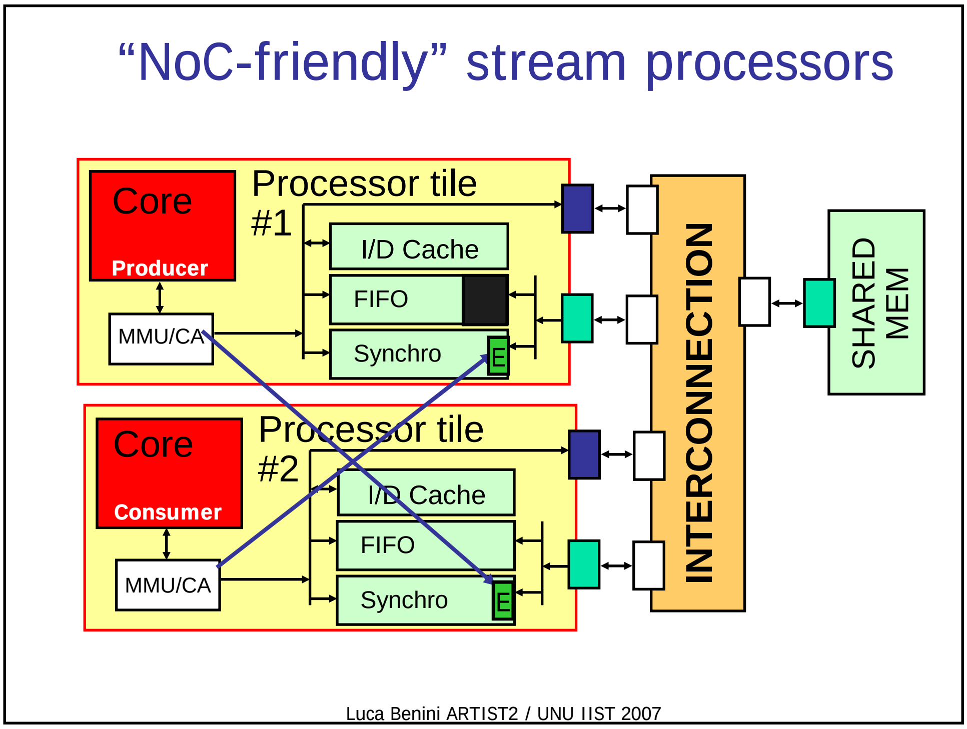
NOC Interconnect

Storage

SRAM
FIFOs
DMA-SRAM
MPort-SRAM
BUF-SRAM

IOs

DRAMs
Video
Network
SerialIO



“NoC-friendly” stream processors

Core
Producer

Processor tile #1

MMU/CA

I/D Cache

FIFO

Synchro

E

Core
Consumer

Processor tile #2

MMU/CA

I/D Cache

FIFO

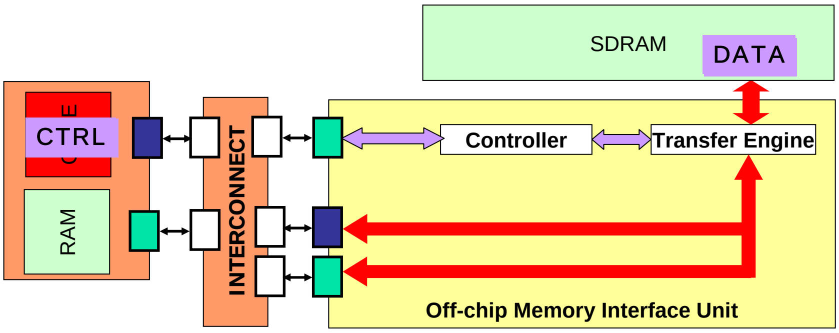
Synchro

E

INTERCONNECTION

SHARED MEM

External Memory Bottleneck



- *"Pull" memory channel*
  - *Control Block* keeps programmable table of objects to be moved
  - Table entries can be programmed by different cores
  - *Transfer Engine* shuffles data among bus and Memory Controller
  - Triggers bus or SDRAM transactions
  - *Memory Controller* handles SDRAM accesses

Summary

- Why SoCs?
- SoC Platforms
- From SoC to MPSoC
- From MPSoC to NoC