



ARTIST Summer School in Europe 2009

*Autrans (near Grenoble), France*

*September 7-11, 2009*

# **Designing Scalable and Predictable SoC Communication Fabrics**

Lecturer: Luca Benini

PhD, Professor

Università di Bologna



# Embedded Applications Trends





# Consumer/Home applications trends



[ARM]



# Consumer/Home applications trends

- Increasingly complex functions
- Highly distributed (multi-standard wireless)
- Widely ranging environments
- Plenty of configurations/ customizations
- Tightening power budget
- Key enabler greener home appliances!
- Soft-RT constrained
- Focus on User Quality-of-Experience

**Scalable**

**Adaptive**

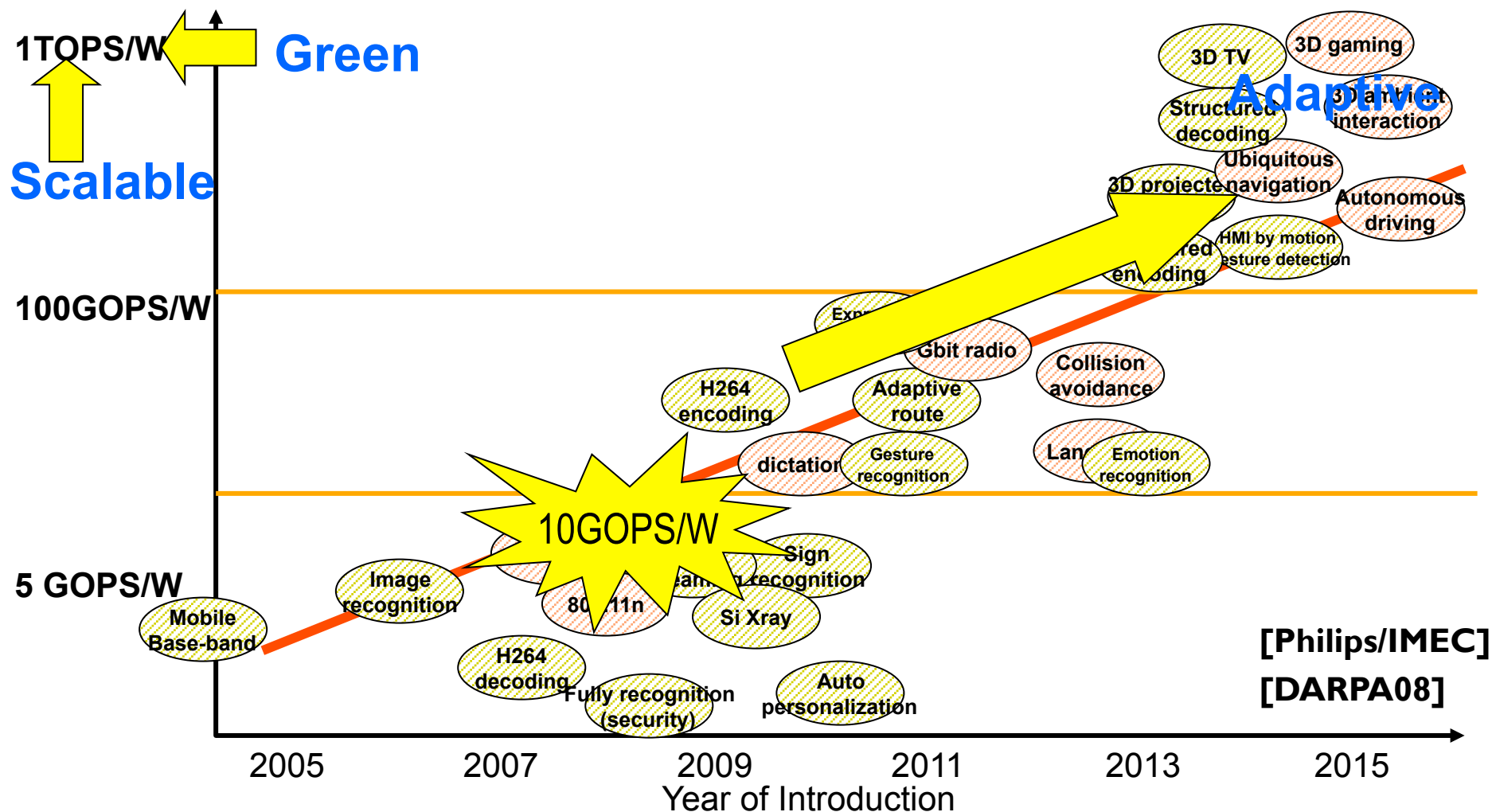
**Green**

**Predictable**

[ARM]



# Embedded applications: Requirements



0.1-1 TOPS/W embedded platforms by 2015!



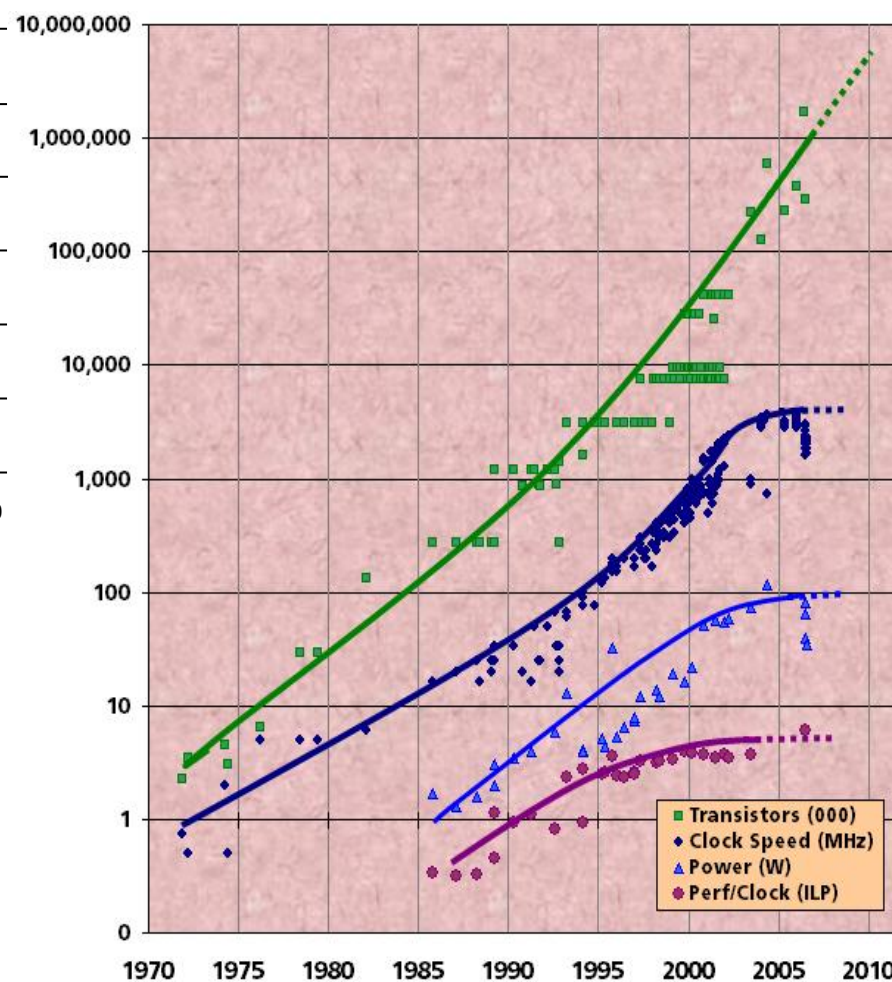
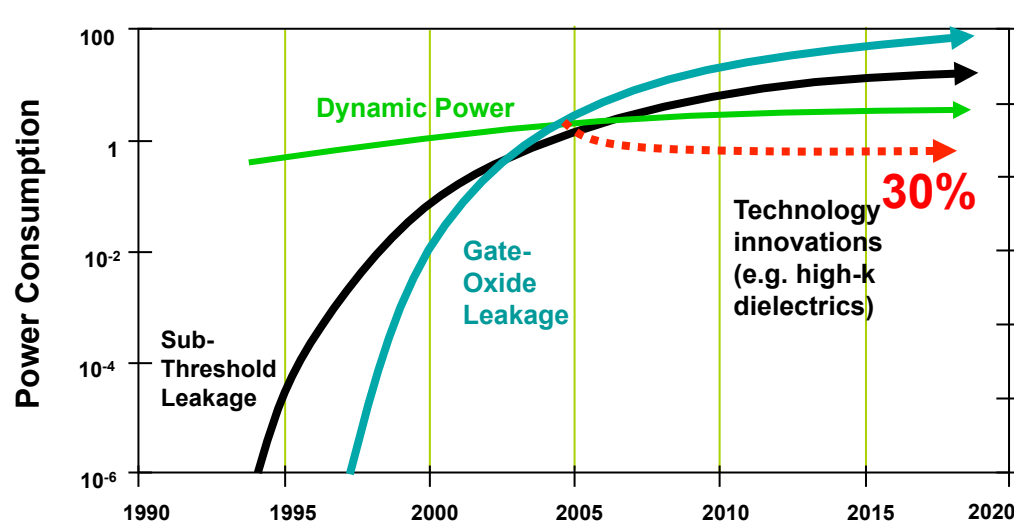
# Technology Limits



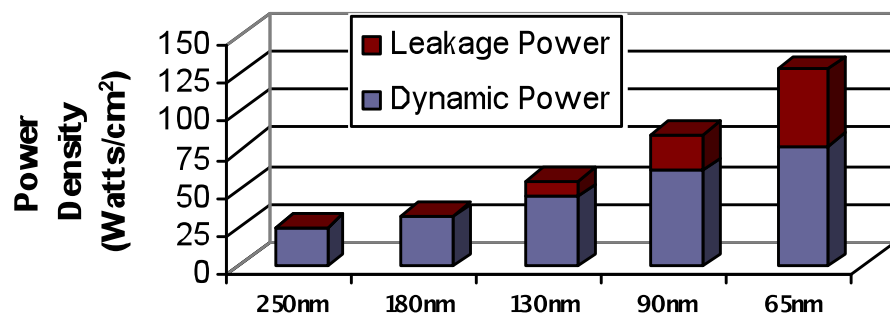


# The Era of “Power Limited Scaling”

Power trend



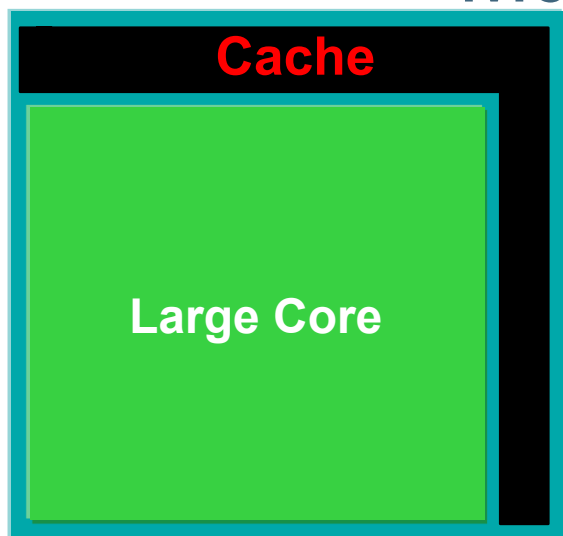
Power density trend



[STM ASIC]

[Intel, Microsoft and Stanford]

# Multi-core and Power



Power

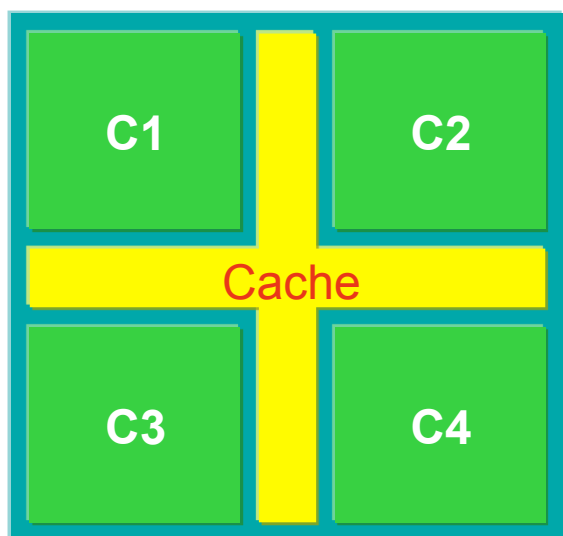
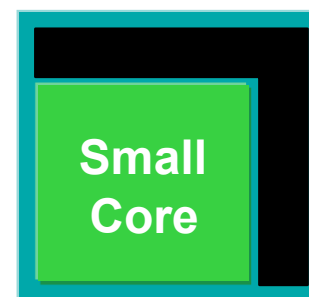


Performance



Power = 1/4

Performance = 1/2



**Multi-Core:**  
**Power efficient**  
**Better power and thermal management**



# How are we going to do it?

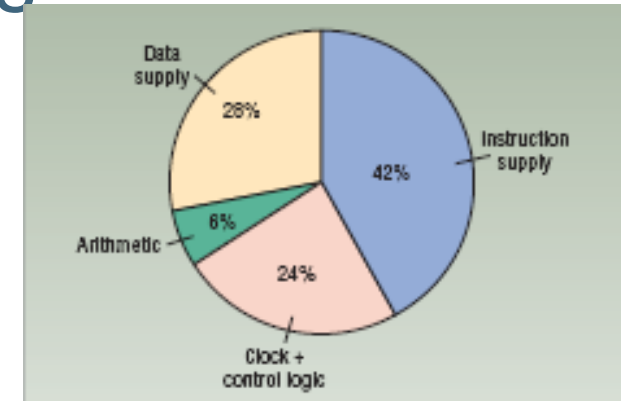
...today: 10s of cores



## Multicore “platforms”

[Dally08]

- Microcontroller (→SMP cluster!) is only the “master processor” (aka *application processor*)
  - Not energy efficient for “number crunching”
- Domain-specific specific functions are integrated at the chip level to supplement the master processor
  - Dedicated IPs for legacy functions and for ultra-high energy efficiency
  - Domain specific processors (DSPs, VLIW, ASIPs) for complex and evolving data-intensive standardized processing (e.g. baseband modem, graphics, multimedia)
- “Kitchen-sink” of standard IOs for maximum interface flexibility (chip reuse + platform derivatives)
- Lots of on-chip memories (caches, scratchpads, buffers)

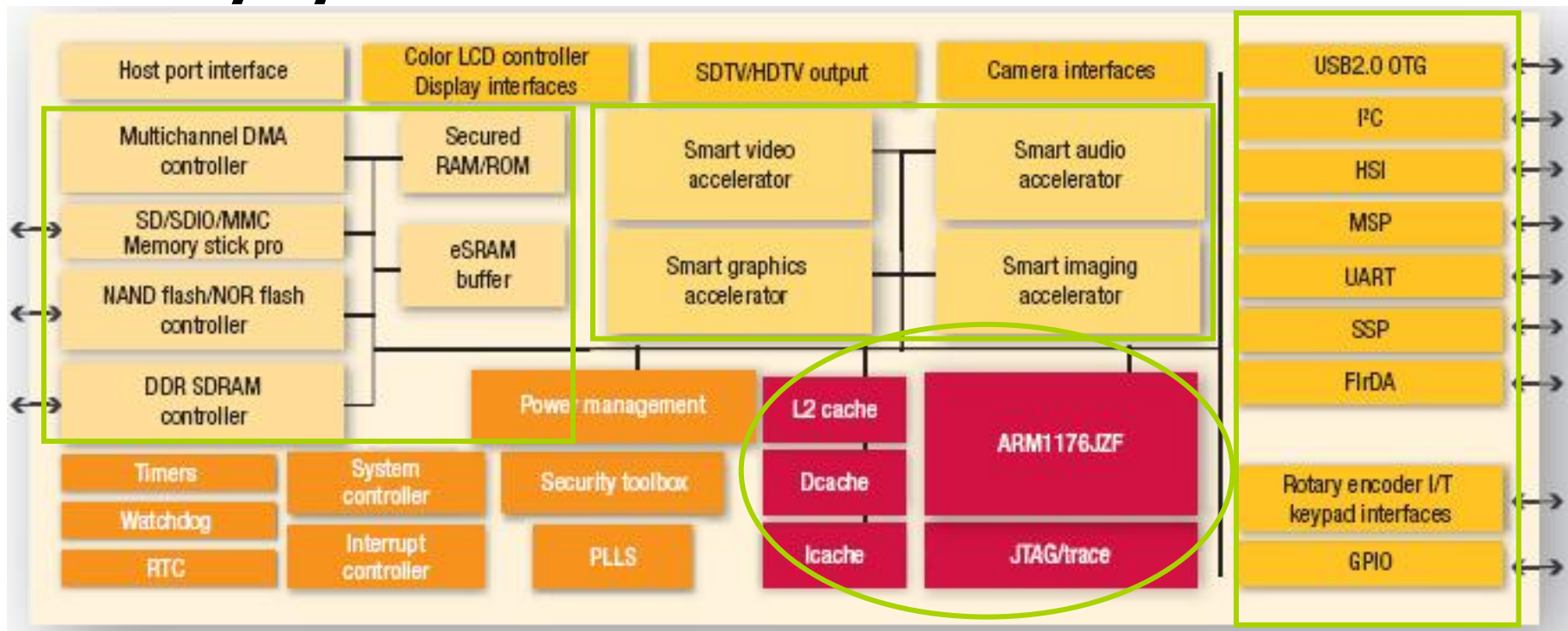




# Nomadik ST-Ericsson (n8820) Application Processor

## Memory System

## HW Accelerators

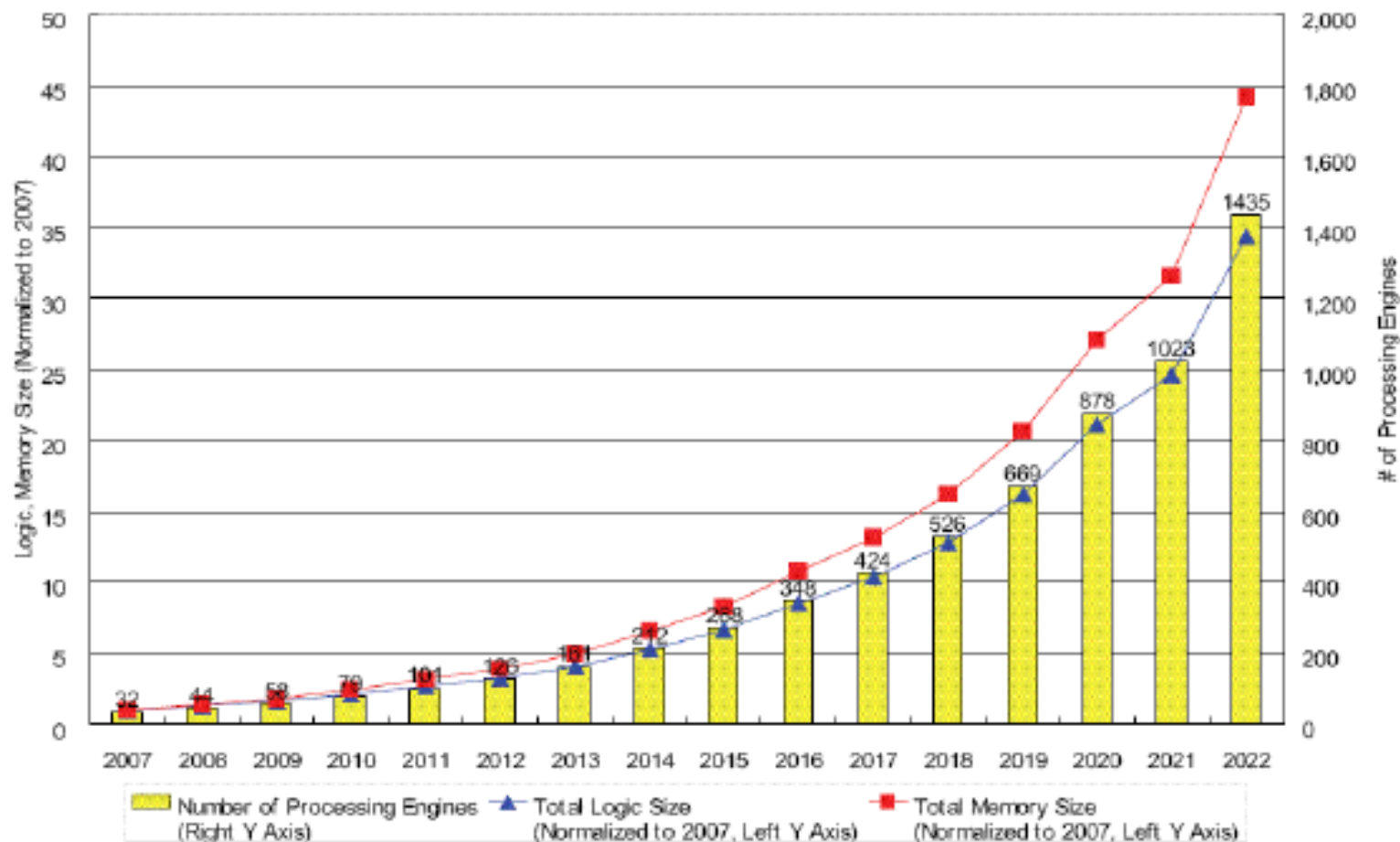


**Main Core I/Os**



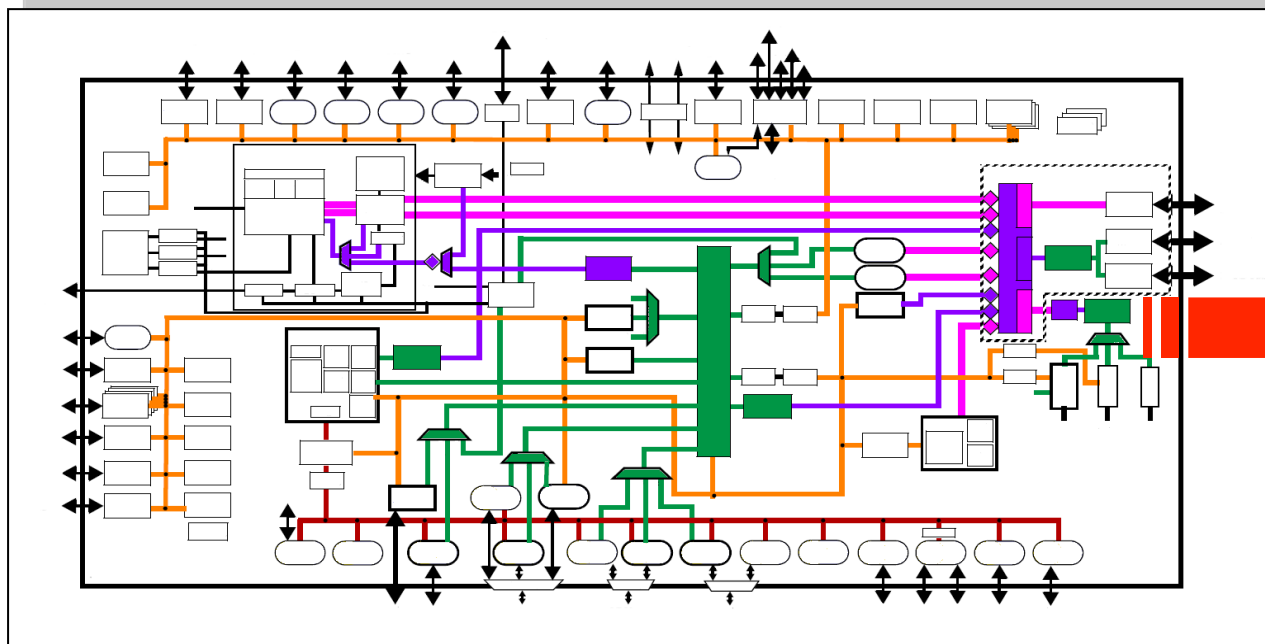
# SoC platform: Quo Vadis?

## ITRS 2007 – SoC Consumer Portable



## Critical analysis

- Complex custom accelerators have high NRE → only massive reuse justifies risk of a new design!
- Kitchen-sink reuse leads to “spaghetti” interconnect



NoCs can help,  
but no miracles

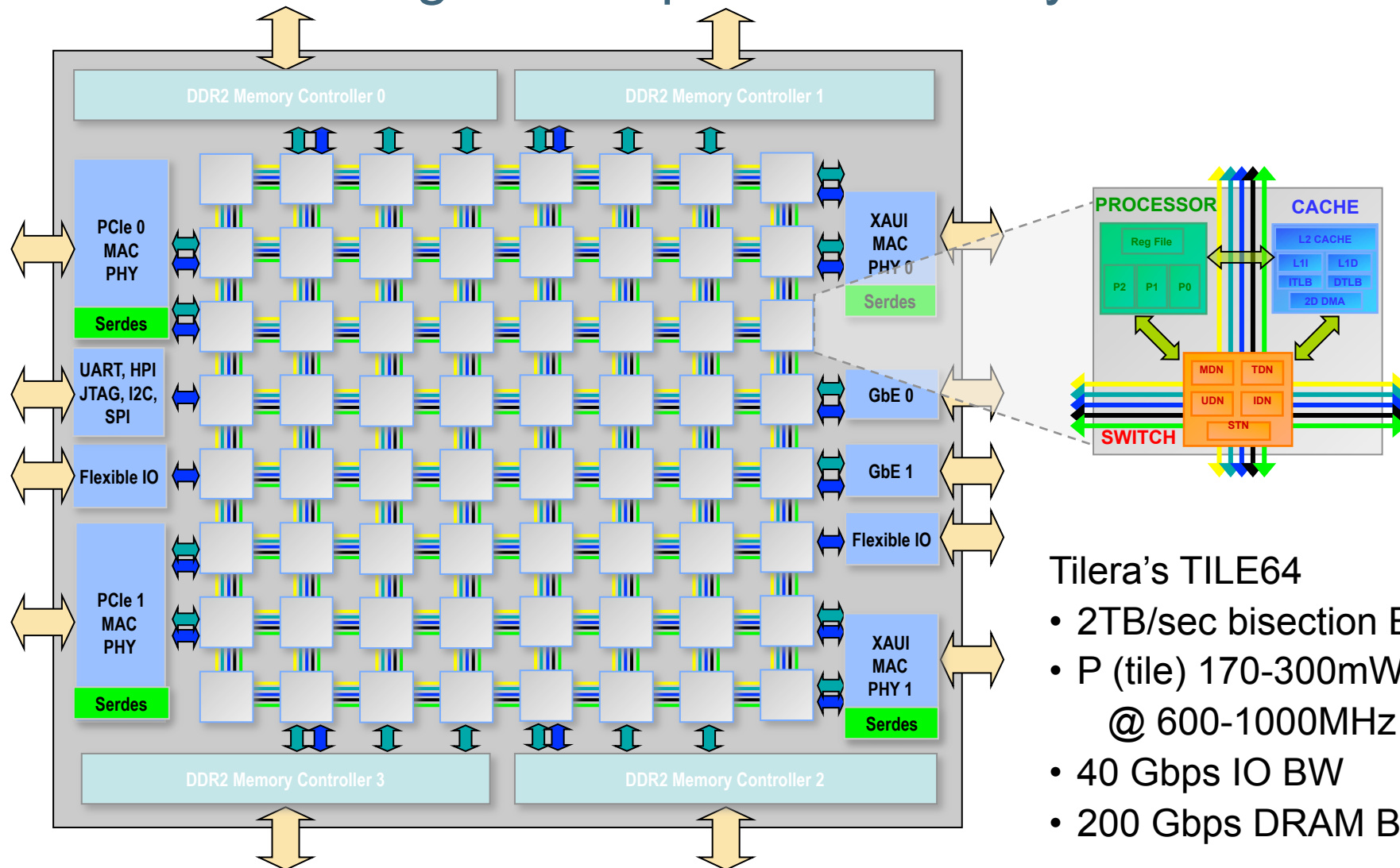
- Programming model is messy → limited flexibility or lots of platform knowledge needed

**Ultimately, not a predictable & scalable solution!**

## Polycore (NoC) Platforms ... the path to 100s cores



# Omogeneous processor arrays



## Tilera's TILE64

- 2TB/sec bisection BW
- P (tile) 170-300mW @ 600-1000MHz
- 40 Gbps IO BW
- 200 Gbps DRAM BW

Regular processor fabric for predictability & efficiency



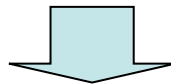
# Tilera's "Gentle Slope" Programming Model

## Gentle slope programming philosophy

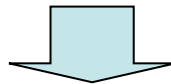
- Facilitates immediate results using off-the-shelf code
- Incremental steps to reach performance goals

## Three incremental steps

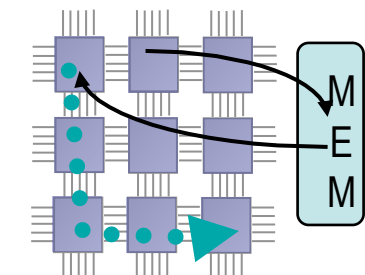
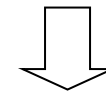
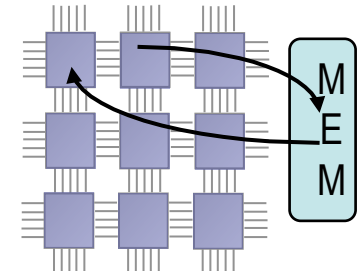
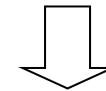
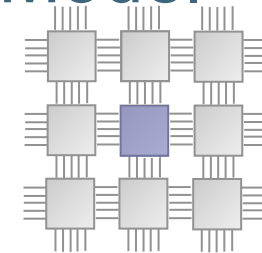
1. Compile and run standard C applications on a single tile



2. Run the program in parallel using standard SMP Linux models – pthreads or processes

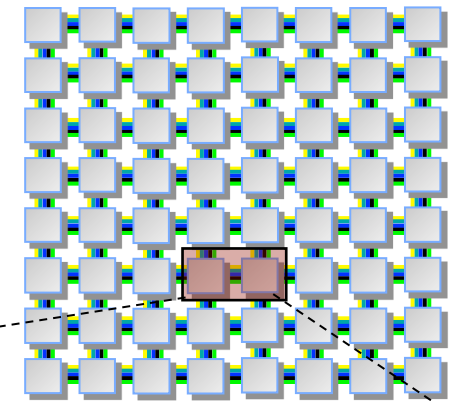


3. Use stream programming using iLib – a light-weight sockets-like API

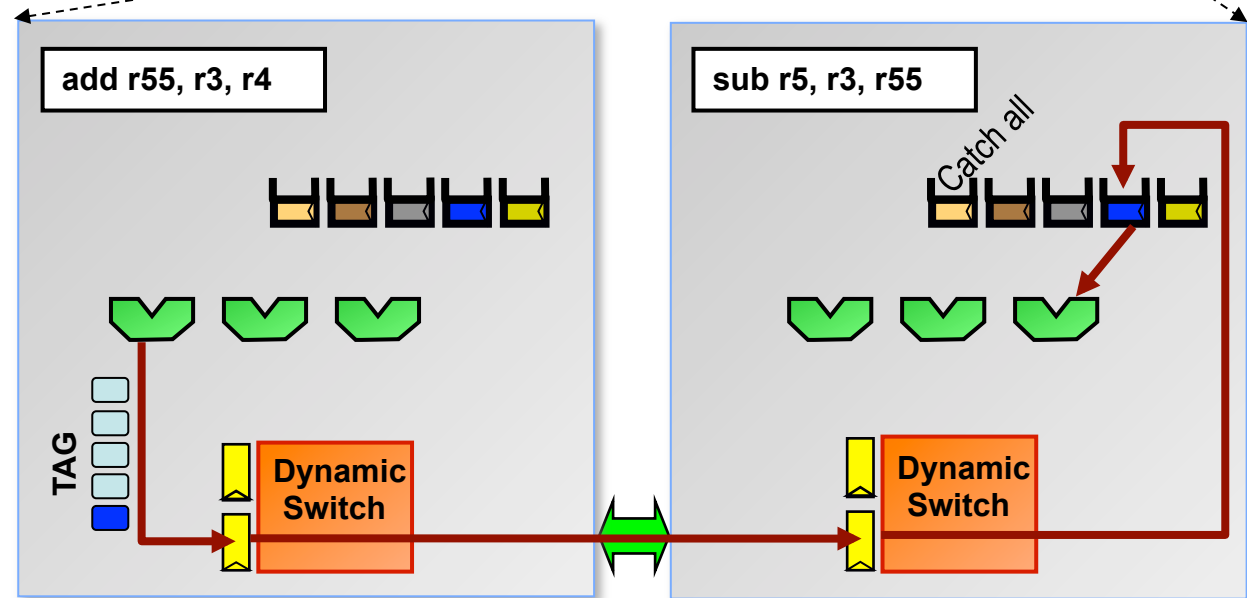


## Stream programming

- Direct user access to interconnect (compiler/library assisted)
- Compute and send in one instruction
- Automatic demultiplexing of streams into registers
- Number of streams is virtualized
- Streams do not necessarily go through memory for power efficiency



**Communication  
exposed  
programming!**





# Predictable execution on 100-cores platforms?

## Good news

- **Distributed storage and multi-hop multi-channel interconnects**
  - More modularity → reduced “implicit” interaction on shared resources
  - More bandwidth available → conservative allocation is viable
- **Simpler processing elements**
  - Less average-case acceleration tricks which adversely impact worst-case
  - Easier construction of high-level performance/power models
- **Explicit communication/ storage programming models**
  - Simplifies high-level SW analysis (distributed, actor-centric programming models)
  - Model-based SW design using pre-verified components

## Bad news

- **Sharing challenge**: Sharing is still needed and must be managed
  - Communication links in NoCs
  - Main memory (DRAM)
  - I/Os
- **Mapping challenge**: Allocation & Scheduling becomes extremely complex
  - Need strong DA support – cannot be done manually
  - Spatial+temporal optimization problem (place/route/schedule)
- **Variability challenge**: Applications, components and environment are “dynamic”
  - Nothing is fully known at design time
  - Must handle unreliable fabrics

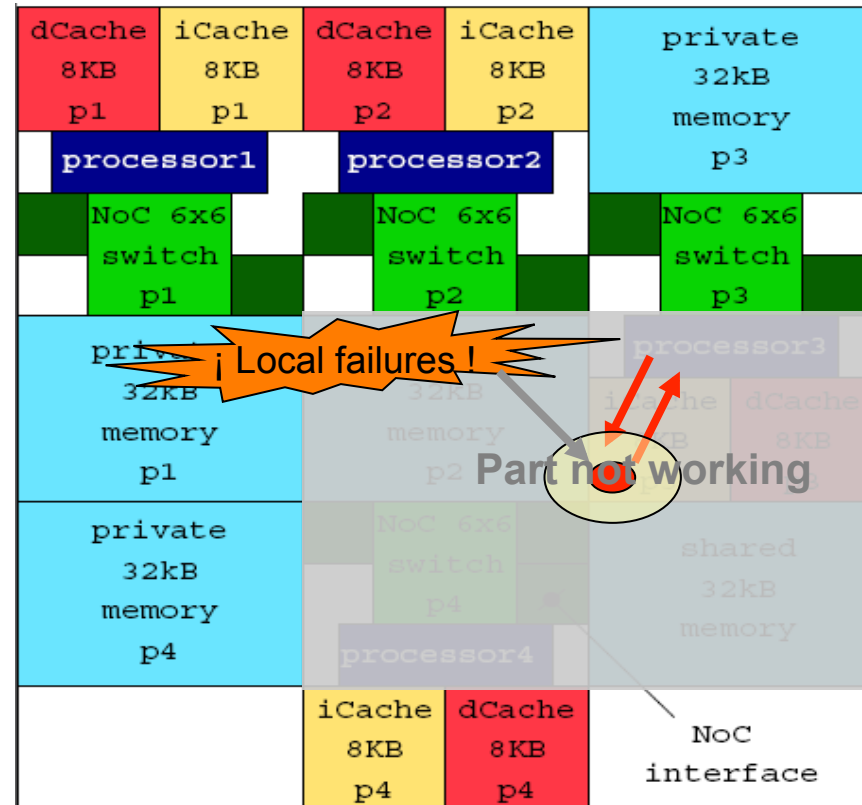
# Challenges

... on the path to 1000 cores



# Variability Challenge

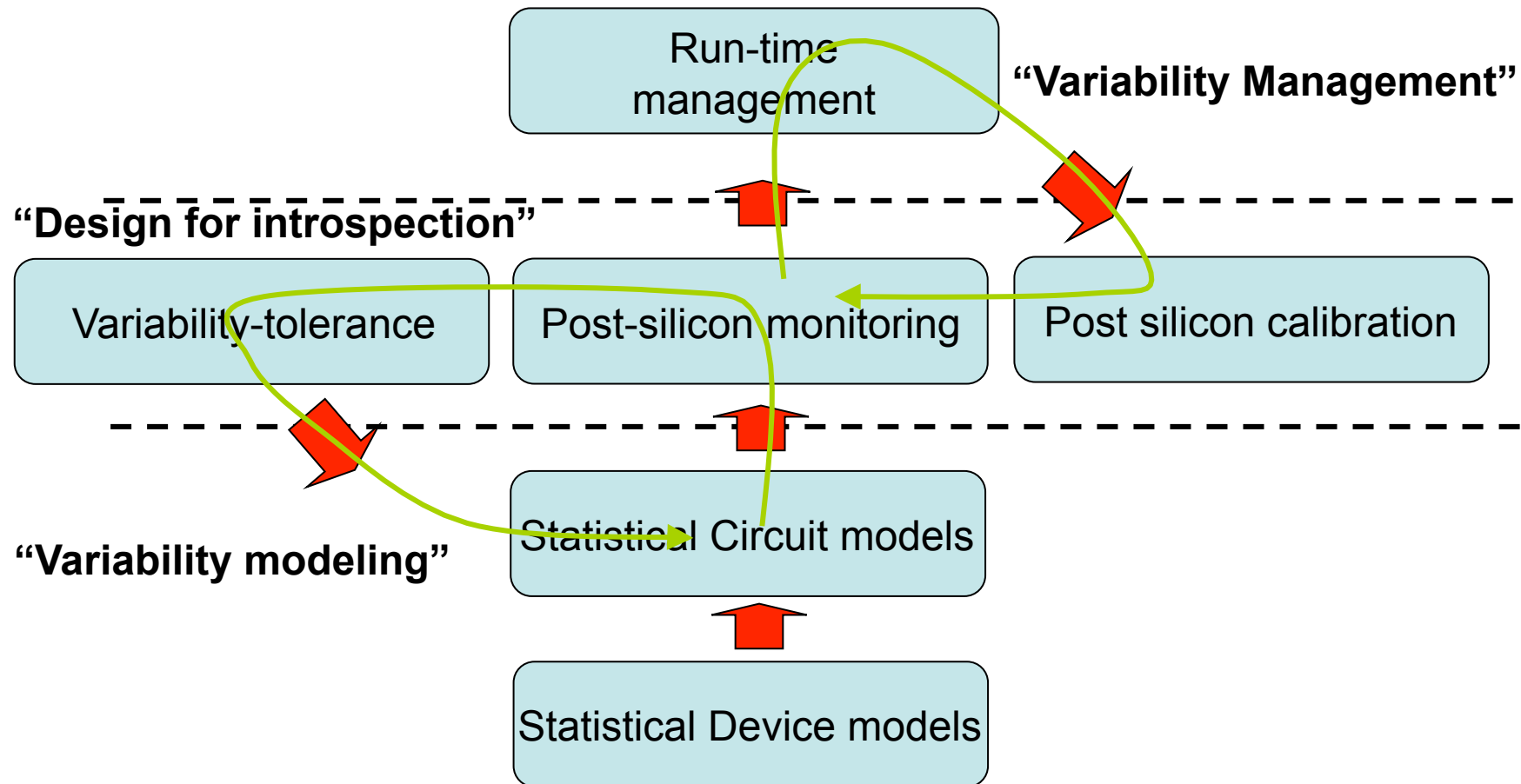
- Complex SoCs in nm-integration:
  - Systematic and random variations (static variability)
- Non uniform workloads, high operating temperature and hot spots
  - Non-uniform ageing (dynamic variability)
- Low voltage operation of scaled transistors
  - Low-signal-to-noise ratios (run-time errors)



**Static and dynamic variations:  
How can we manage these effects?**

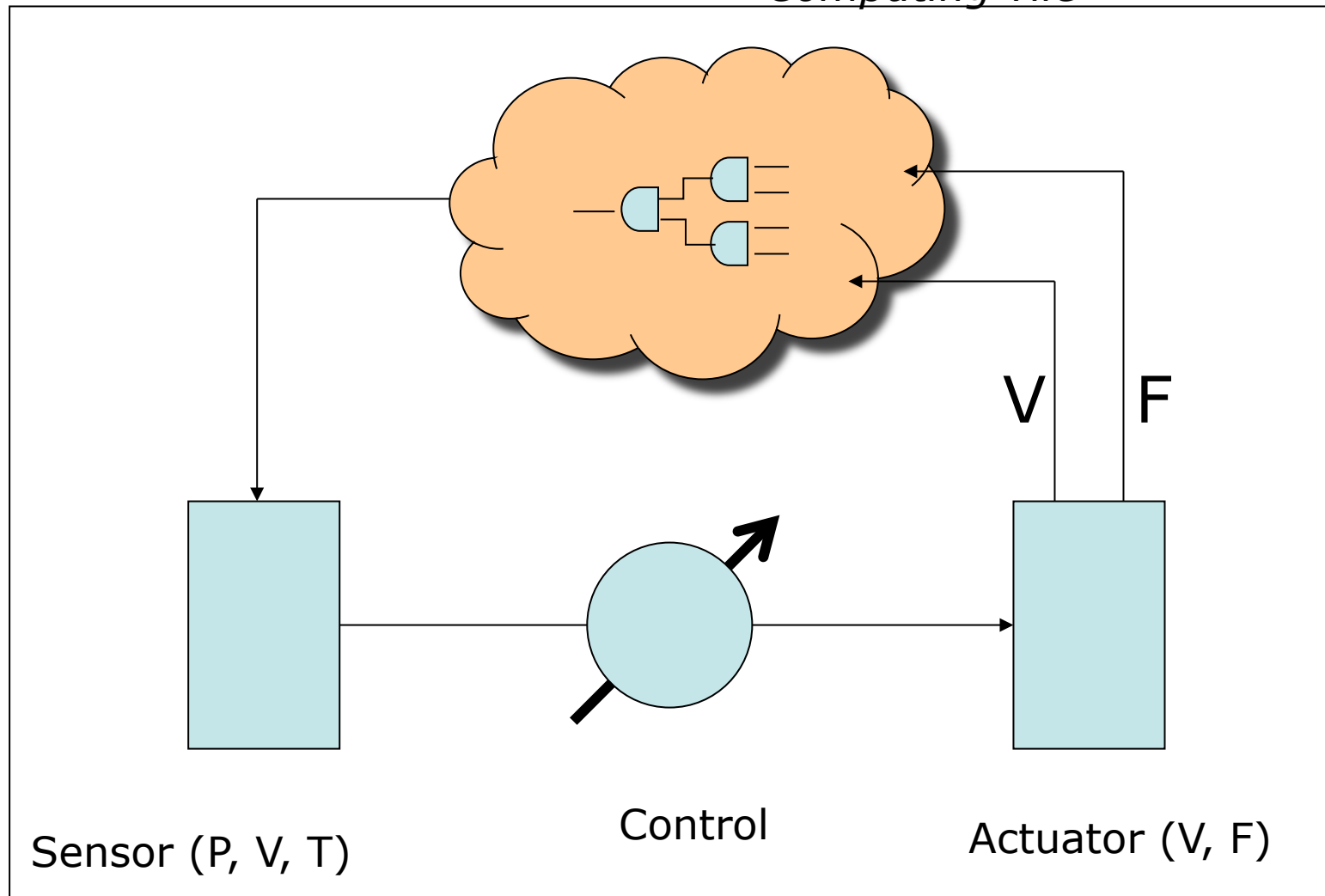
# Robust design

## Introspective platforms



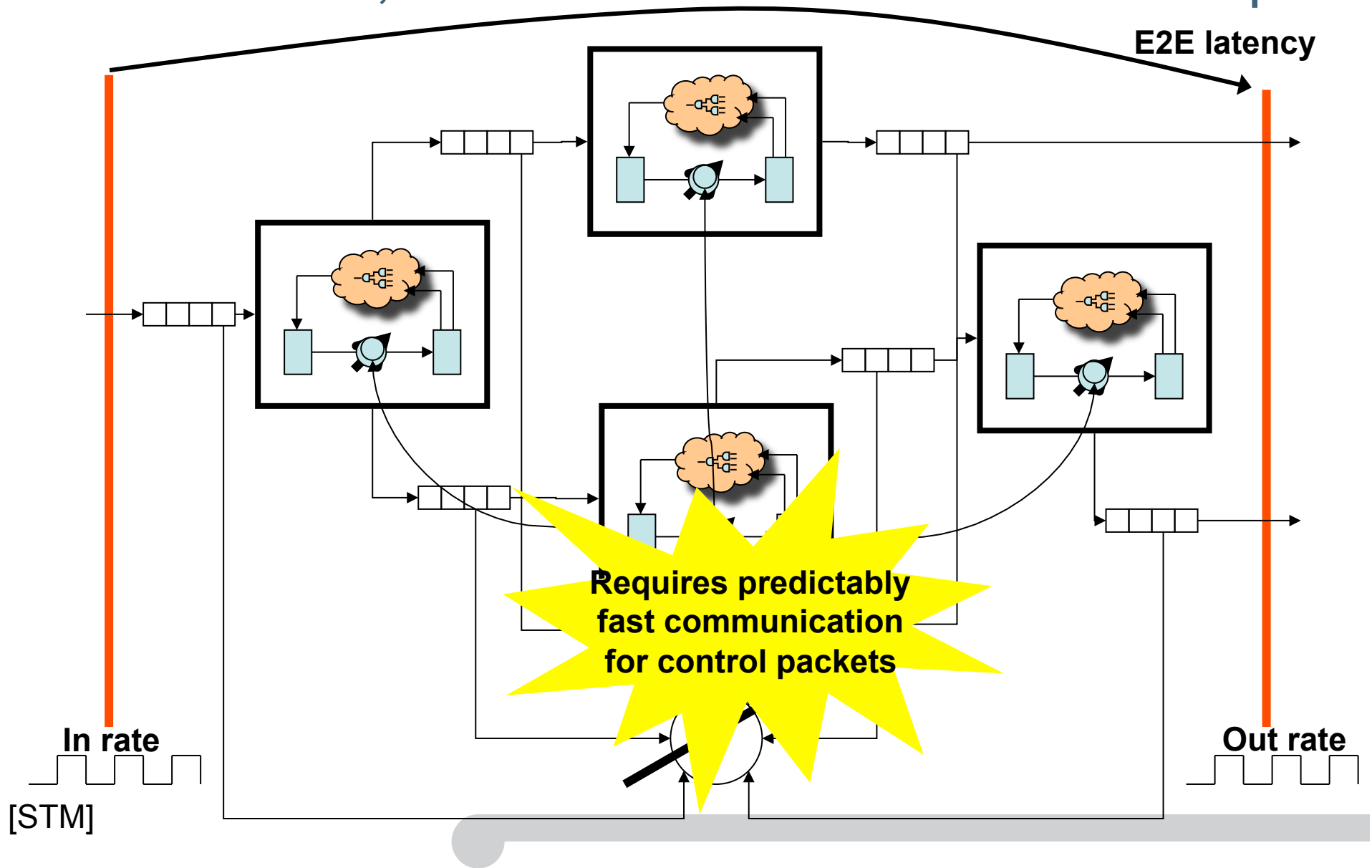
# Local Closed Loop Control at Tile Level

*Computing Tile*



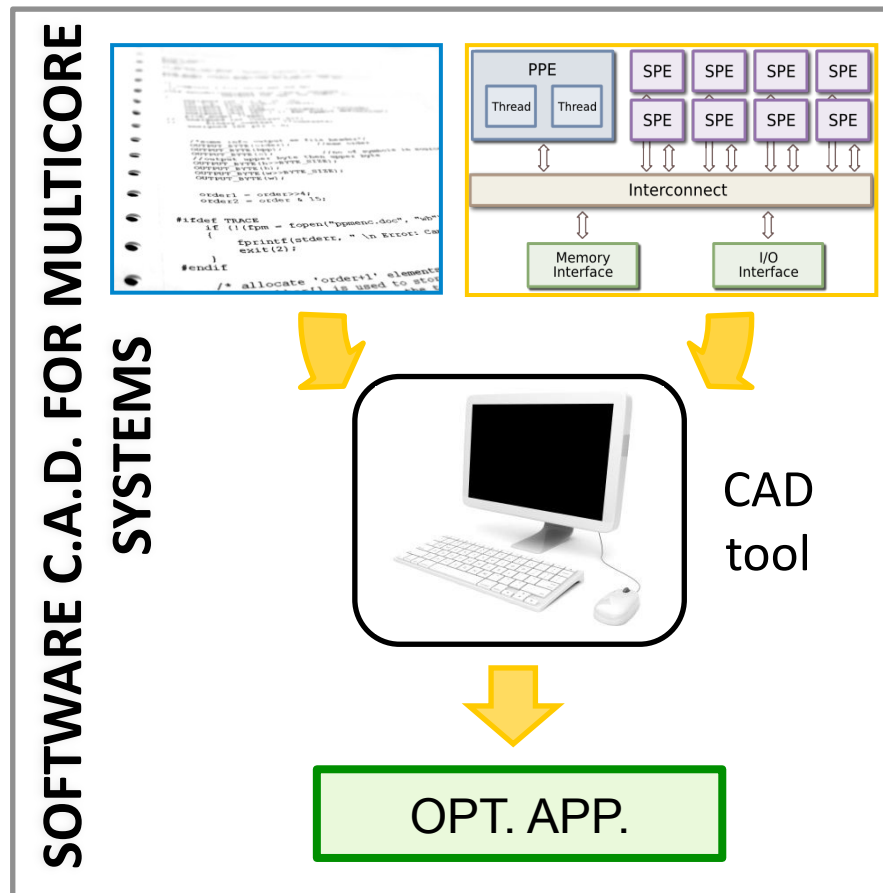
[STM]

# Distributed, hierarchical Global Control Loop



## Mapping Challenge

Predictable resource allocation & scheduling of parallel applications on MPSoC platforms for real time systems



Given:

- App. Description (Task Graph)
- Durations as BCET/WCET
- Platform description

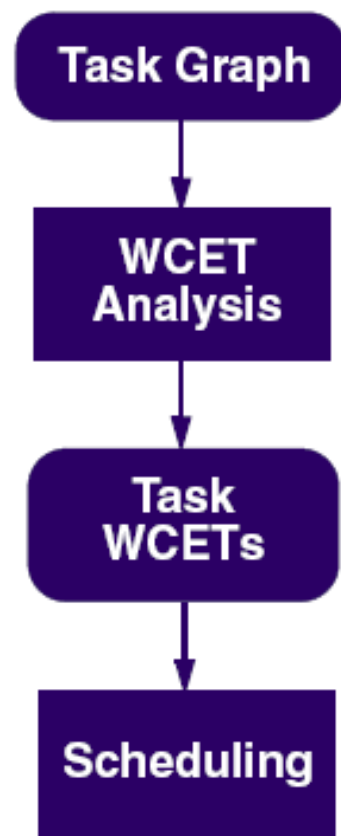
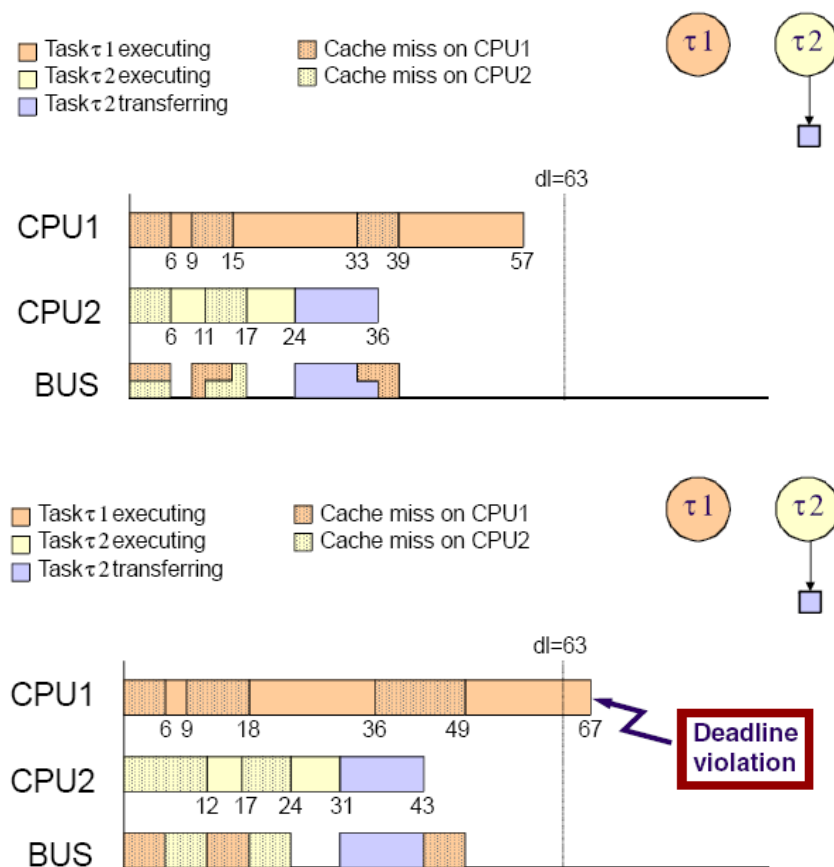
**Compute allocation & schedule**

Guaranteed to meet a global deadline constraint for every execution scenario

➔ Lothar's talk (later today)

# The problem with Sharing (interconnect, cache)

[Eles09]



**Circular Dependency!**

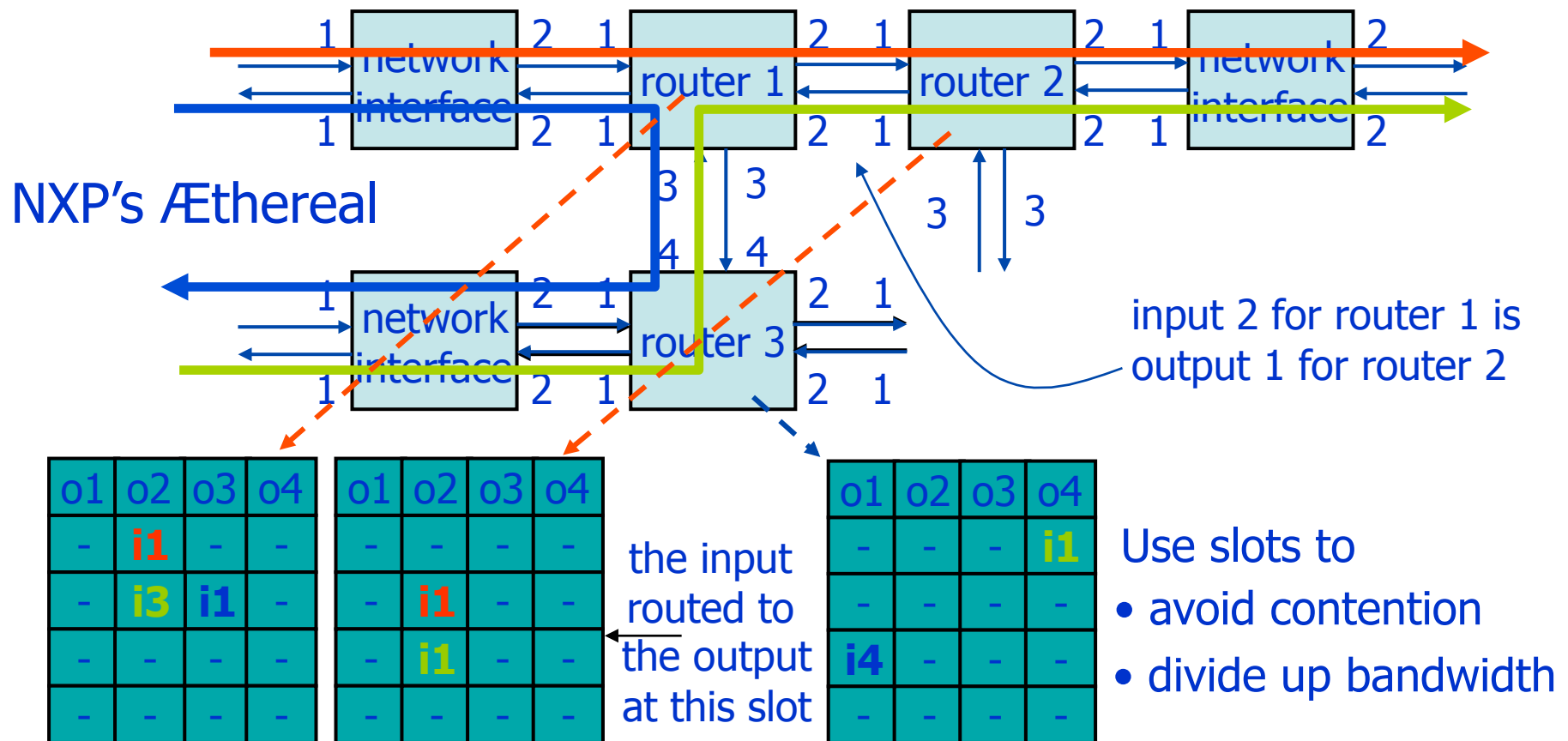
- The WCETs depend on the schedule (interference due to bus conflicts).
- The schedule depends on the WCETs.

## Sharing challenge

- Communication fabric is shared
  - We need **Predictable and scalable** communication
- We know (...almost) how to make a shared bus predictable
  - Field buses: CAN, Flexray, TTTbus, ...many standards
  - On-chip buses: Sonics
- NoCs are harder: **distributed, and must be efficient**
  - Circuit switching (virtual, physical, hybrid) e.g. Aethereal
  - Priorities and preemption, e.g. QNoC
  - Best effort with boundary traffic regulation, e.g. Xpipes
- Predictability proof is non-trivial in all cases
  - Requires hard guarantees on bandwidth & latency

# Contention-Free Routing with Pipelined TDMA

- Emulate circuits → **Schedule packet injection**

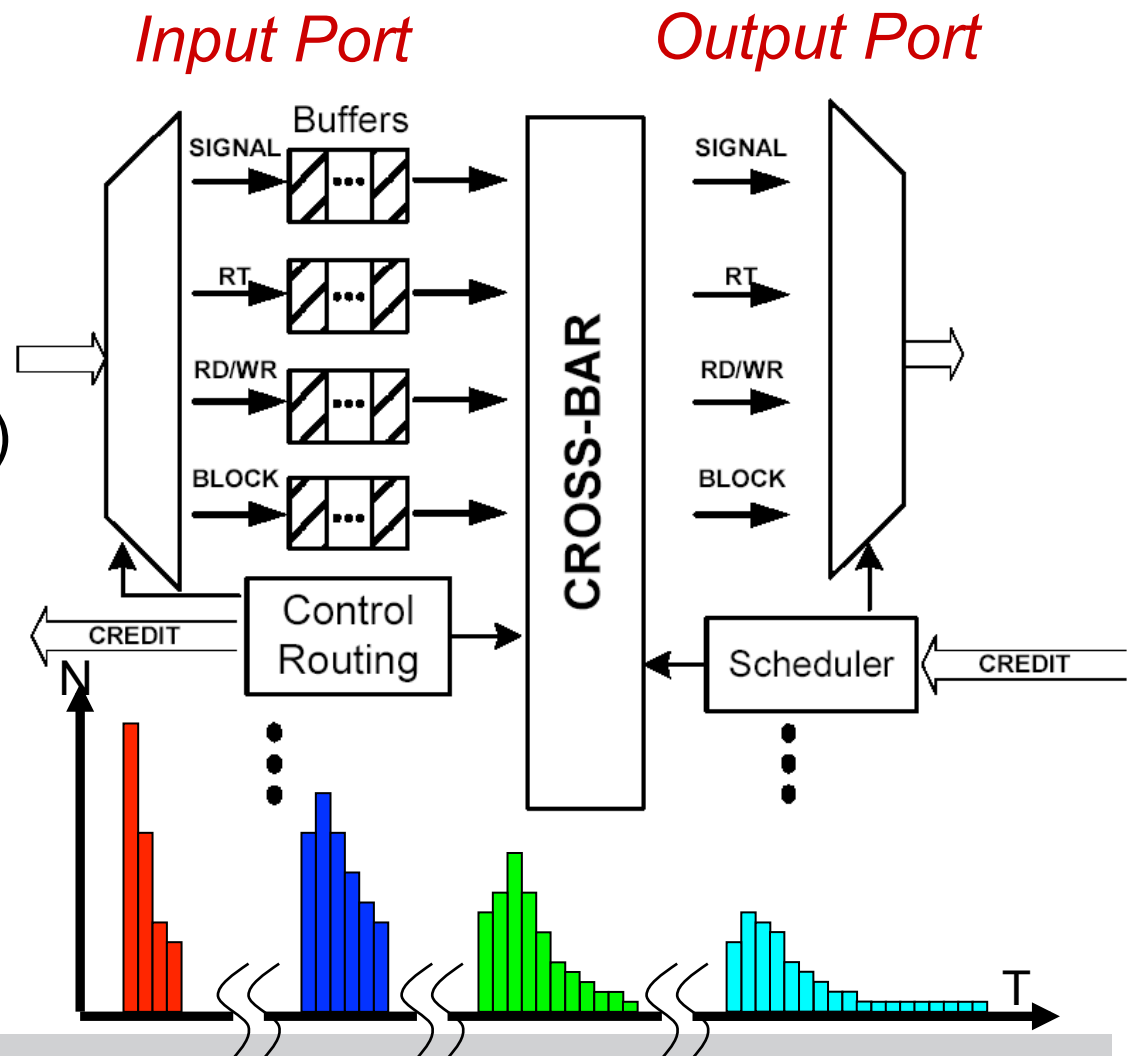


- Alternative resource reservation schemes are possible!

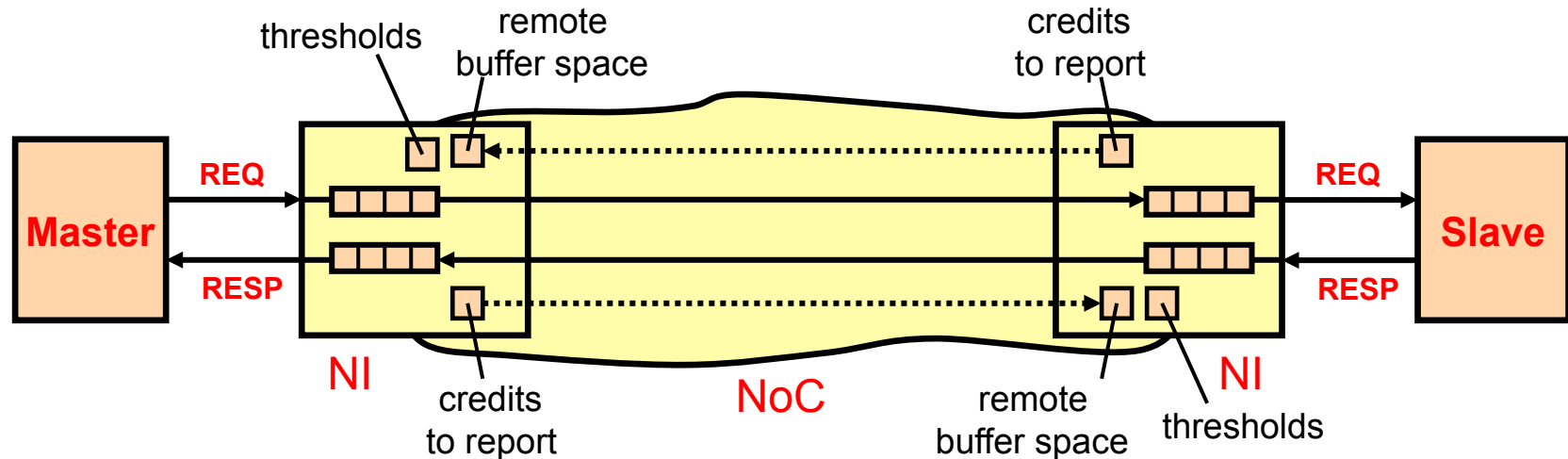
# Packet priorities with preemption

QNoC: [Bolotin04]

- Multiple priority classes
  - **Signaling**
  - **Real Time Stream**
  - **Read-Write**
  - **DMA Block Transfer**
- Requires as many VC as priorities (!)
- Statistical “guarantees” (!)
  - E.g. <0.01% arrive later then required
- **[Shi08] present solid WC analysis**
  - Bounds are quite conservative
  - Jitter is an issue



## Predictable Connections?



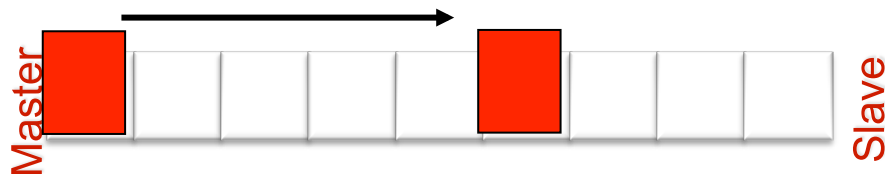
- Best-effort connection setup & teardown
  - Bounds are not defined for this
- Connection management required
  - Credit-based E2E flow control to prevent buffer overflow at consumer
- Predictability of E2E connection depends on target behaviour
  - E2E bandwidth + latency analysis is non-trivial!

An alternative view: E2E connection control is needed anyway. Why not demanding QoS entirely to it?

# Bounding D, BW for RR wormhole NoCs

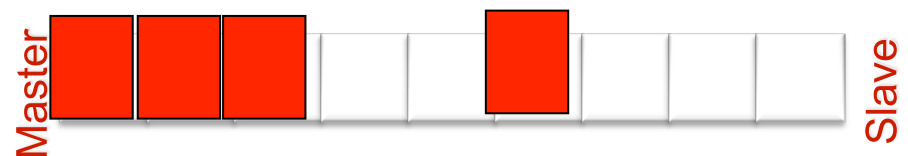
## Method RTB-LL

- Real Time Bound - Low Latency
- Suitable for the applications with flows that have **low worst-case latency** Demands
- Must inject packets at pre-defined intervals (**requires traffic regulators**) to avoid self-interference

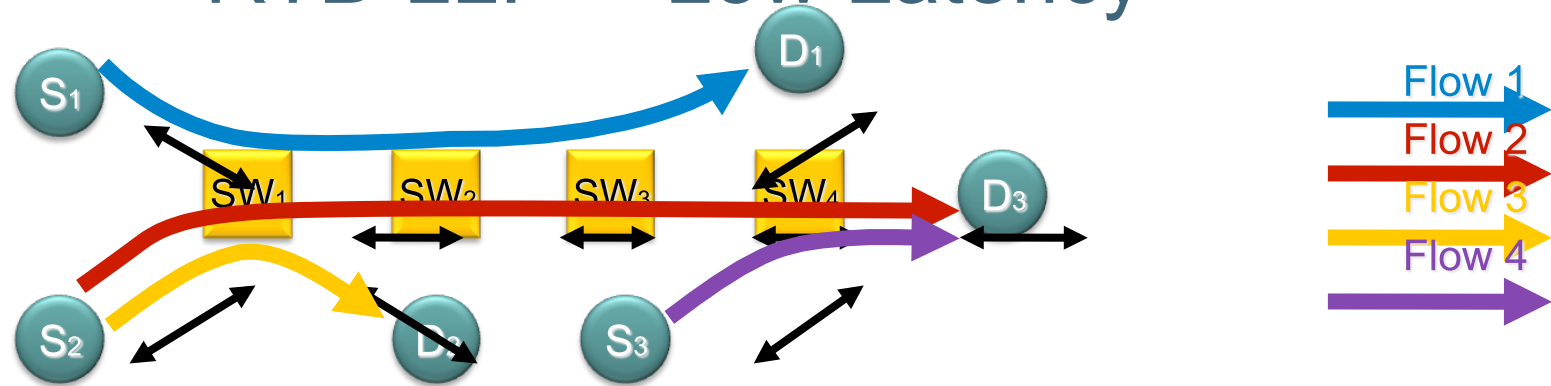


## Method RTB-HB

- Real Time Bound – High Bandwidth
- Suitable for applications with flows that have **high average bandwidth** and **moderate worst-case latency** Demands
- No restrictions on packet injection (**unmodified hardware**), traffic regulated only by backpressure



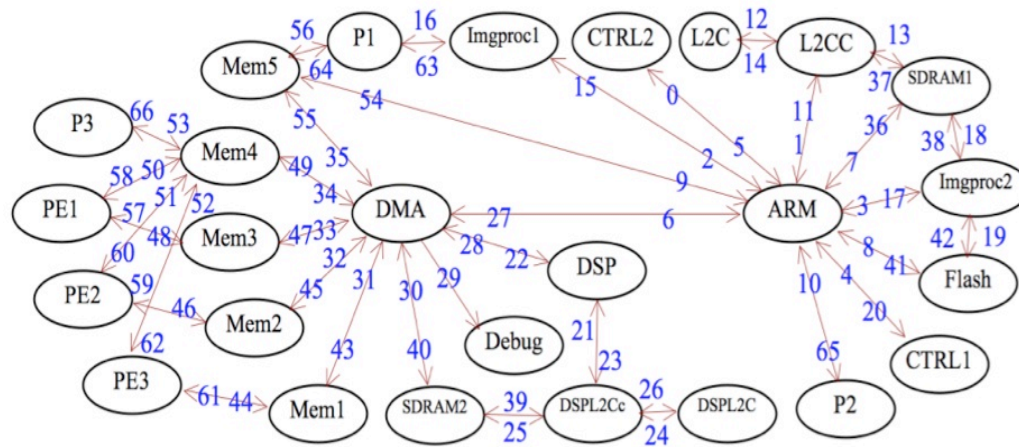
## RTB-LL: Low Latency



Calculation example:

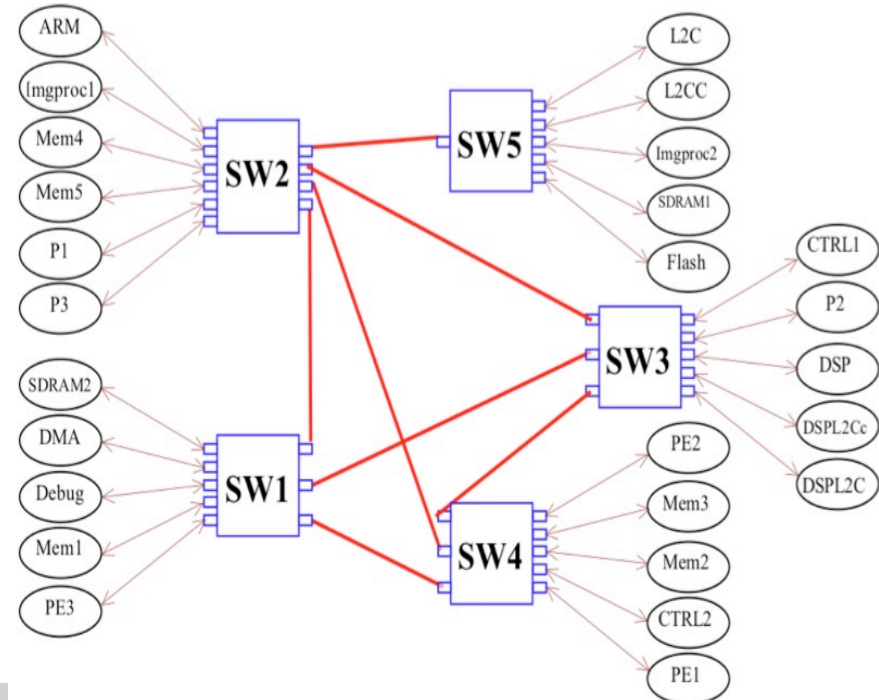
- Upper Bound Delay for flow 1 :  $UB1 = \text{Zero\_Load\_Delay} + L + u_1(1) + u_1(2) + u_1(3)$
- $u_1(1)$  : blocking time of p1 (of flow 1) in switch 1 = direct contention in sw1 + indirect contention in switch 4 =  $L + L$
- $u_1(2) = u_1(3) = 0$
- $u_k(j)$ : the blocking time of flow  $k$  at  $SW_j$  because of direct or indirect contentions with other flows
- Considering  $(a=1, b=1, L=4)$  :  $UB1 = 4*a + 3*b + 3*L = 19 \text{ cycles}$
- $ml1 = L + u_1(1) + u_1(2) + u_1(3) = 3*L = 12 \text{ cycles}$
- Considering (Flit\_Width=1 byte, Clock\_Freq=300 MHz):
- $MBW1 = L * \text{Flit\_Width} / ml1 * \text{Clock\_Freq} = 100 \text{ MByte/sec}$

## A sample application

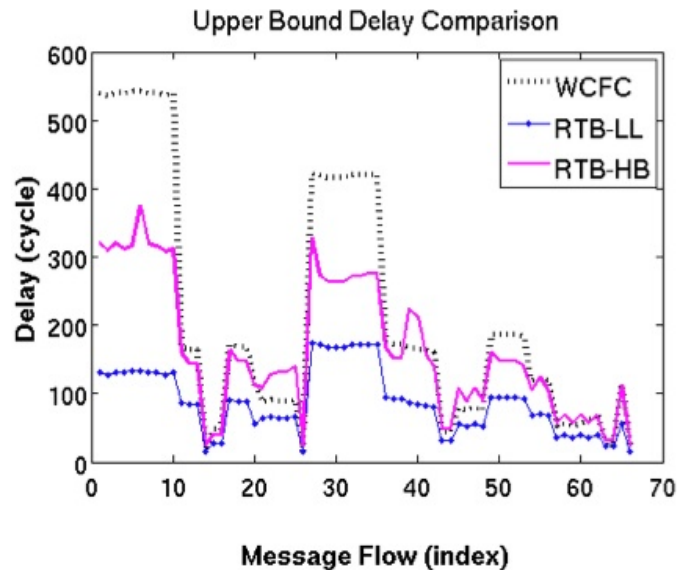


A real-time application for  
NoC with 26 Ip-Cores and  
67 traffic flows

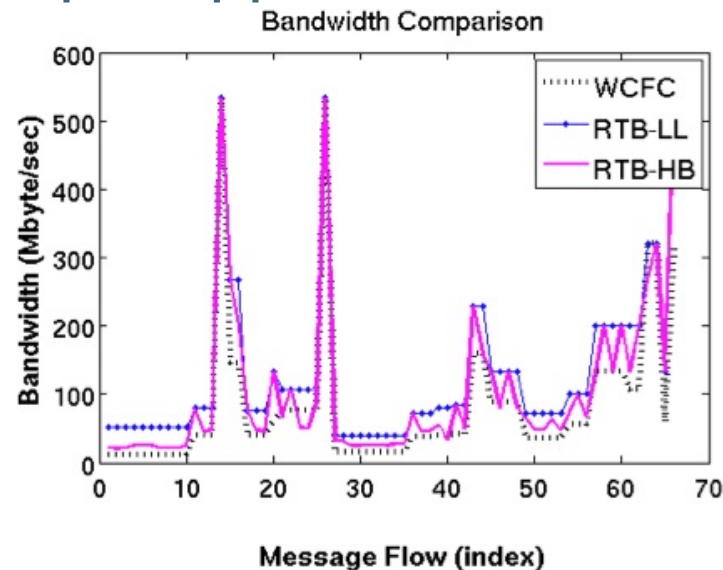
Mapping the  
application on a 5-  
switches network



## Results for the sample application



UB (cycles)	Minimum	average	maximum
RTB-LL	17	91	139
RTB-HB	24	174	392
WCFC	17	237	545




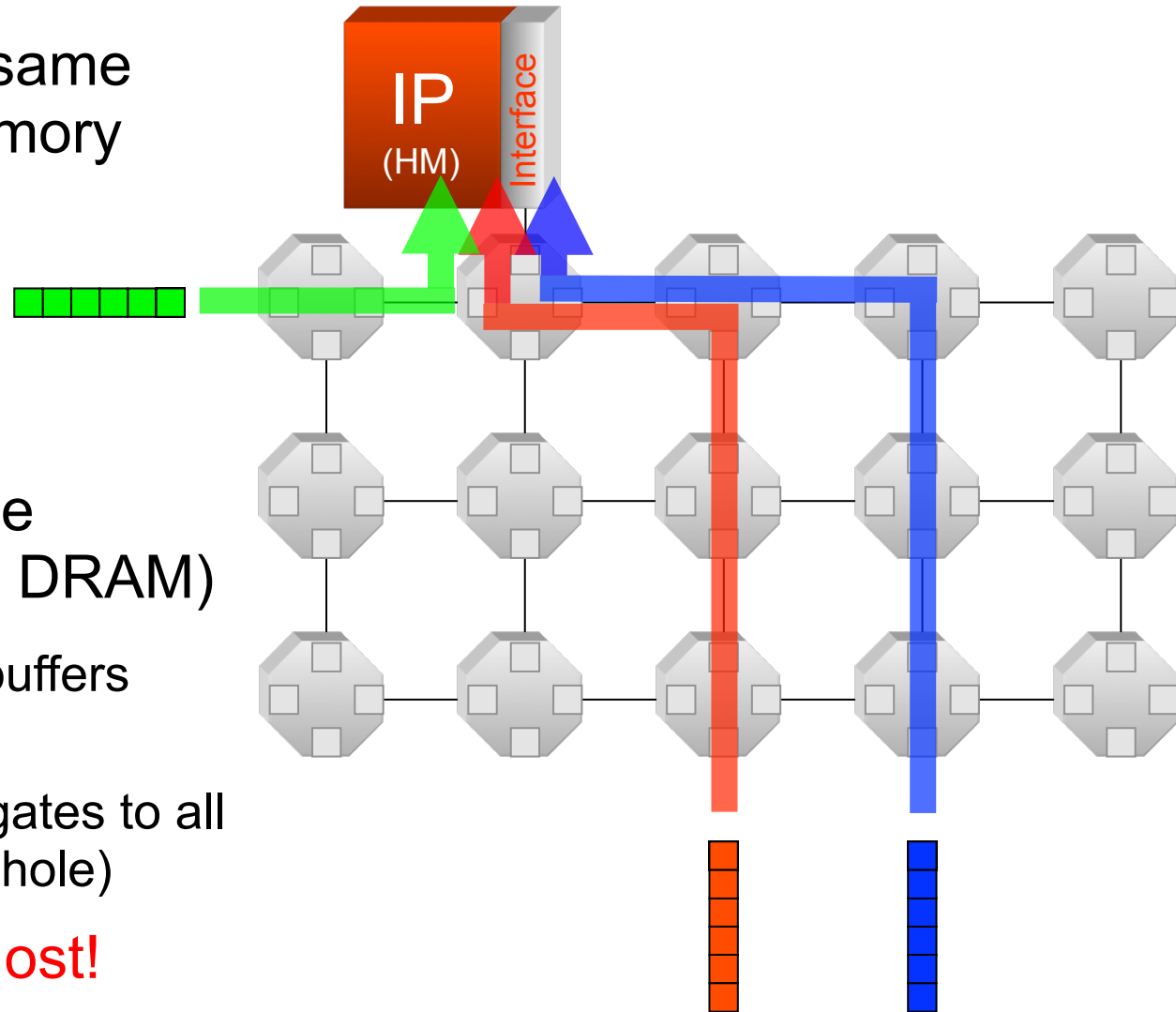
BW (MB/s)	minimum	average	maximum
RTB-LL	36	124	533
RTB-HB	21	101	533
WCFC	12	81	533

- Analysis is very fast, but conservative (competitive with network calculus)
- Assuming no backpressure from destination nodes

**NoC-level QoS support may give better control on latency+bandwidth**

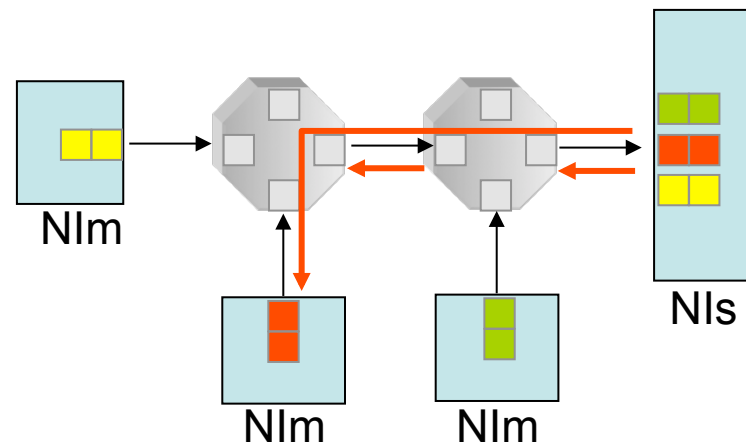
# A Plumbing Problem: Hot Module

- Many flows to the same end-point (e.g. memory controller)  

- Module with variable response time (e.g. DRAM)
  - Request channel buffers may fill-up
  - Congestion propagates to all the network (wormhole)
- All guarantees are lost!



# Addressing the Hot Module Problem

- E2E control [WalterNOCS06,AkessonCODES07]
  - Memory controllers supports fair memory allocation
  - Backpressure at the initiator NIs (prevent clogging)
- Distributed control [YooDATE09,JangDAC09]
  - Packets are prioritized according to MEM sequencing
  - Network buffers exploited to reduce MemCTRL buffers

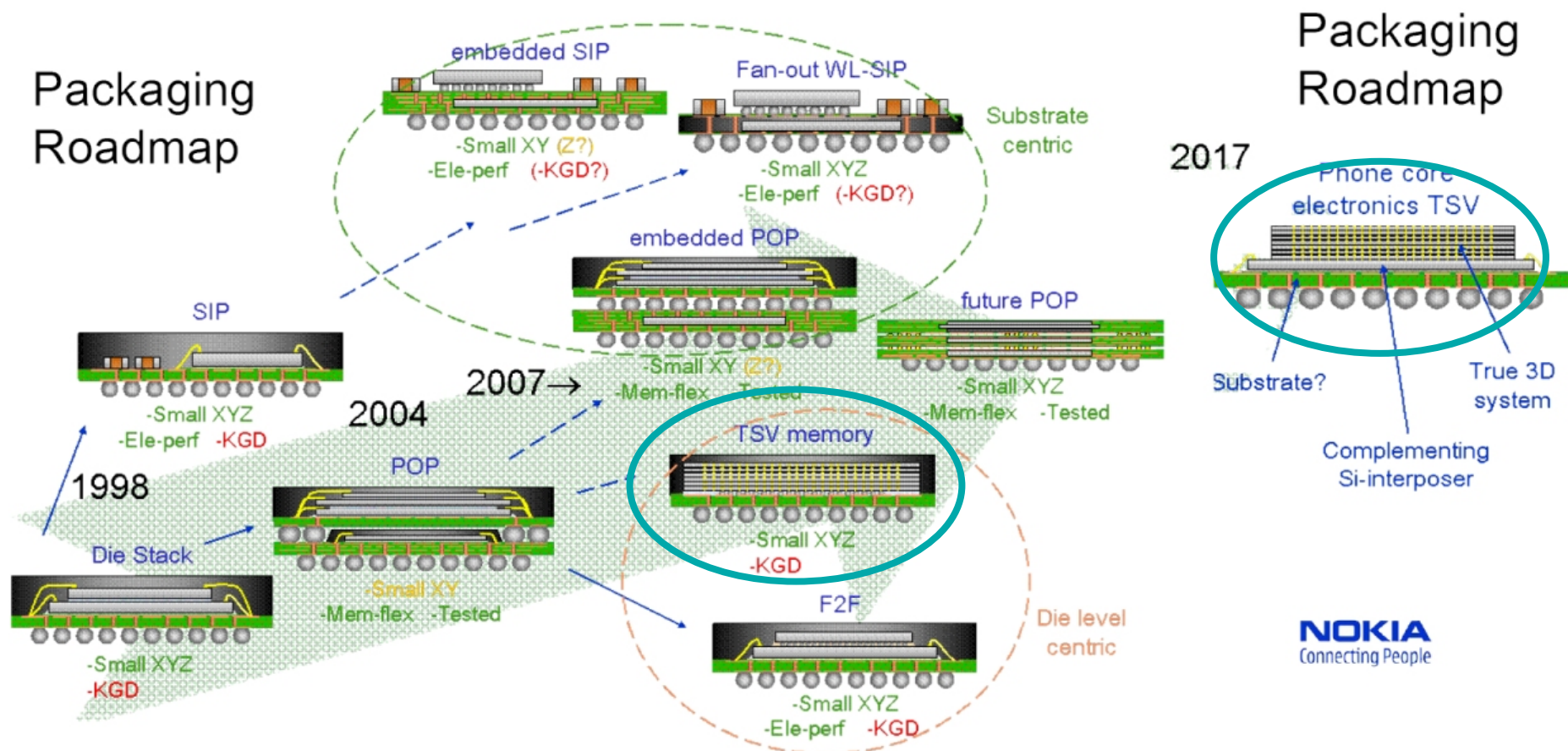


## Summing up

- Predictable NoCs do exist
  - Several mechanisms are available
- Predictable NoC abstraction
  - Given a set of flows  $\rightarrow$  for each flow provides max delivery time and min bandwidth guarantees
  - Requires global analysis  $\rightarrow$  takes time – issues in doing it on-line
  - Assumes well-behaved initiators **and** targets
  - Resource underutilization is a price to be paid
- Predictability is a “max-min” property: as strong as its weakest component!
  - Currently external (DRAM) memory interface is the weakest component

# Help from technology

Bandwidth hungry, even more so for predictability

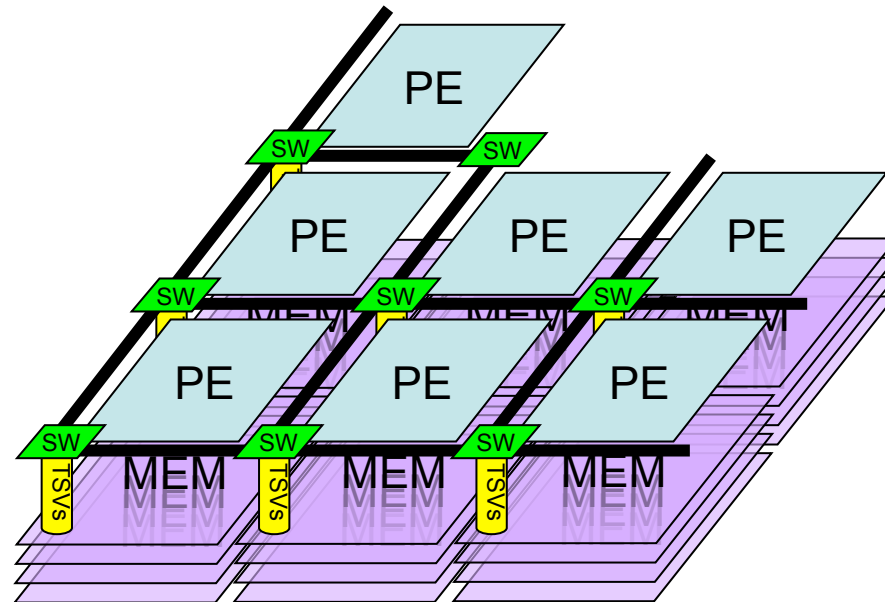


Through-silicon vias are at the technology bleeding edge today  
Industry interest is growing: <http://www.emc3d.org/>

# Scalable & predictable 3D-platform

## 3D-Network on-chip

- Packet-based communication with **QoS support** (TDMA/priorities/regulated traffic)
- **Architecturally scalable**: more nodes, more bandwidth
- **Physically scalable**: segmented P2P links



## Vertically Integrated main memory

- TSV main-memory communication from 10pJ/bit to 10fJ/bit
- $10^5$  interconnect density increase
- Priority/Bandwidth reservation (mainly for low-latency memory neighborhood)

# STMicroelectronics 2012: Architecture Template

- Synchronous Computing Domain. Redundant Grain
- SMP Cluster
- Voltage & Frequency Island. Isolatable

- Decoupled Domains
- Data Flow Programming Model

- Large Stacked Memory Multi Way Access (one per main) using TSV

- Packet Based (NoC) regular Communication Infrastructure

[STM, CEA]

- Synchronous Computing Domain. Redundant Grain
- SMP Cluster
- Voltage & Frequency Island. Isolatable

- Decoupled Domains

- Data Flow Programming Model

- Packet Based (NoC) regular Communication Infrastructure

Large Stacked Memory  
Multi Way Access (one per  
main) using TSV

[STM, CEA]

## “Reconciling” scalability & predictability

- Sharing cannot be fully avoided
  - But it should be made explicit (existence & cost) → you can access foreign memory neighborhoods, but with an explicit (and deterministically bounded) cost
  - Strict access policies for shared resources can be enforced (e.g. no starvation) at a modest hardware cost
- Scalability is required for predictability
  - Bottlenecks kill average **and** worst-case performance
  - Scalability implies some “*marginal over-design*”, but it pays off
- Complete design-time knowledge is not required
  - Safe assumptions at design time + slack reclamation at run time
  - This goes hand-in-hand with (possibly significant) over-design



Thank you!

**ACK: FP7 Predator, Genesys, Share**

A thick, light gray horizontal bar with a rounded left end, positioned at the bottom of the slide.