Networks on Chips (NoC) –

Keeping up with Rent's Rule and Moore's Law

Avi Kolodny

Technion – Israel Institute of Technology

International Workshop on System Level Interconnect Prediction (SLIP)

March 2007

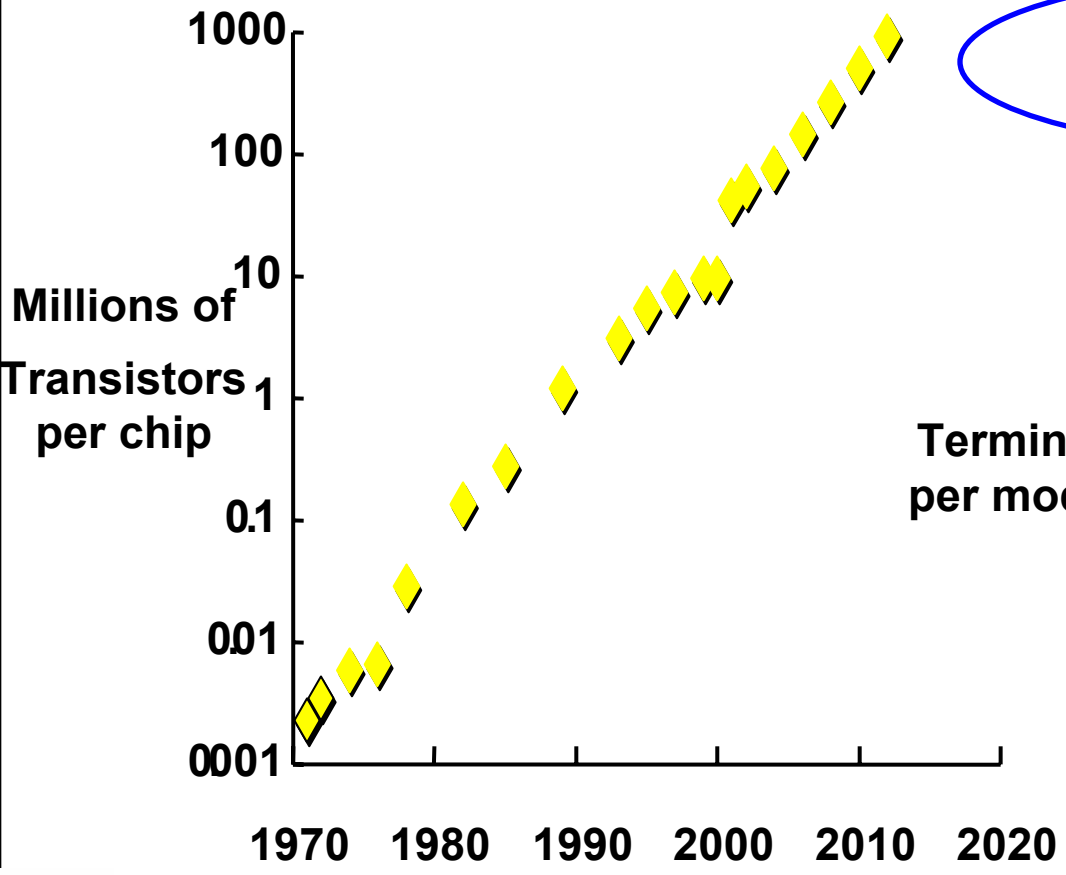
Acknowledgements

- Research colleagues
  Israel Cidon,
  Ran Ginosar,
  Idit Keidar,
  Uri Weiser

- Students:
  Evgeny Bolotin,
  Reuven Dobkin,
  Roman Gindin,
  Zvika Guz,
  Tomer Morad,
  Arkadiy Morgenshtein,
  Zigi Walter

- Sponsors

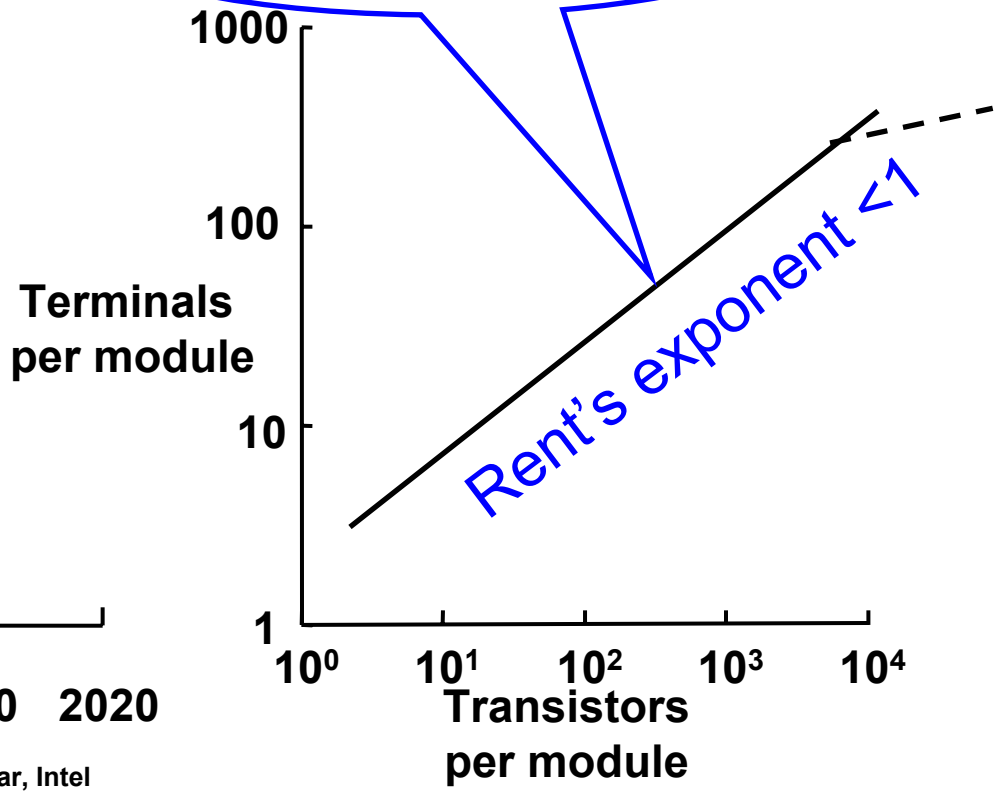
Keeping up with Moore's law:



Source: S. Borkar, Intel

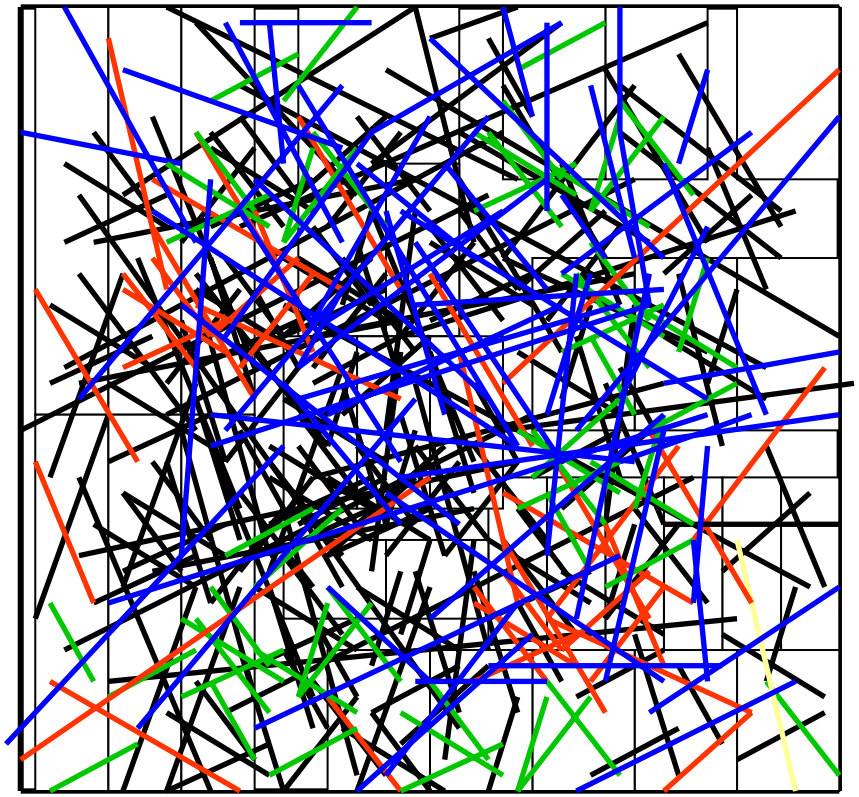
Principles for dealing with complexity:

- Abstraction
- Hierarchy
- Regularity
- Design Methodology

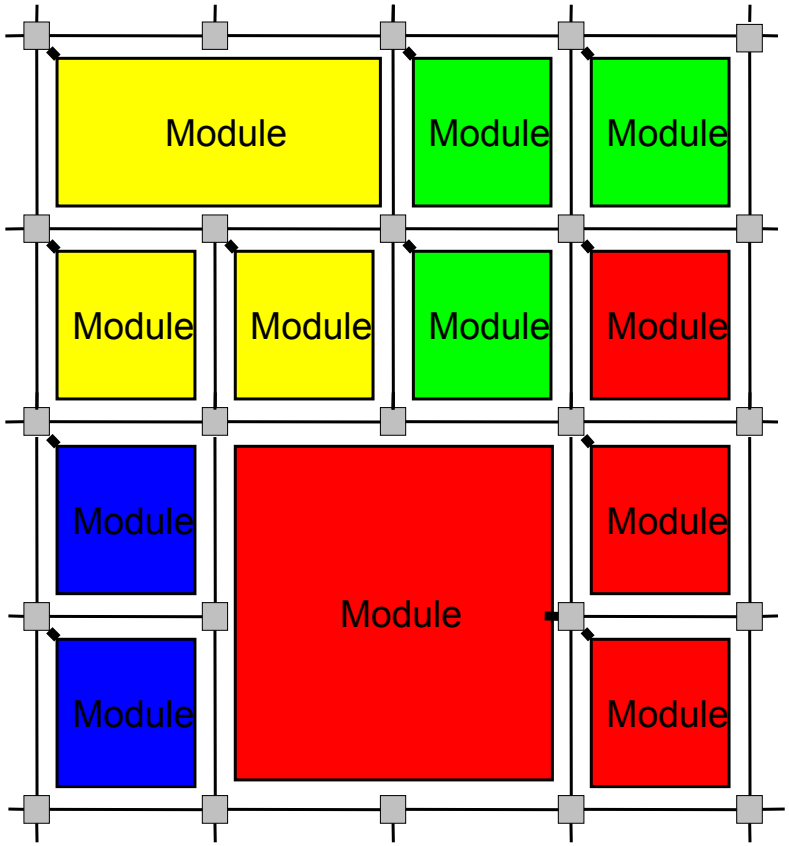


NoC = More Regularity and Higher Abstraction

From: **Dedicated signal wires**



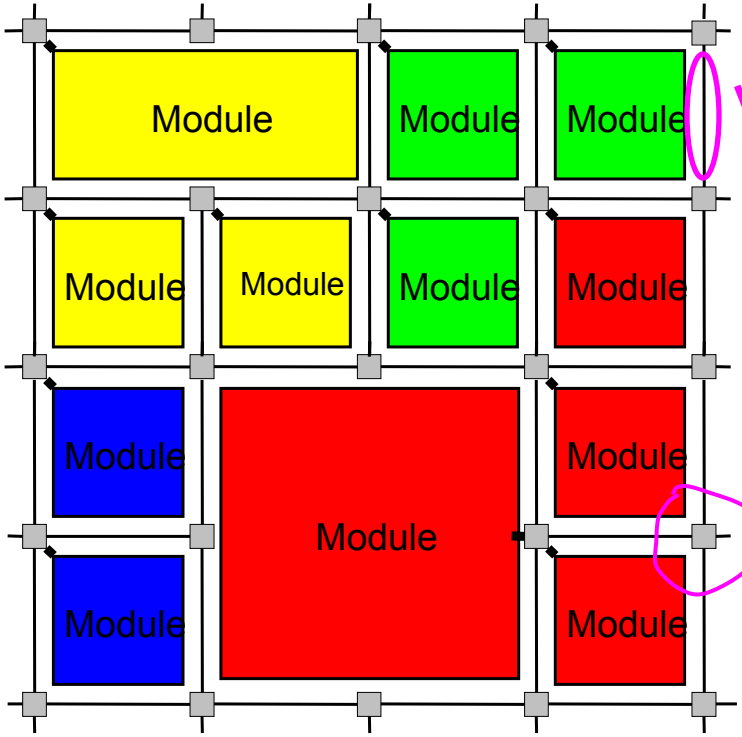
To: **Shared network**



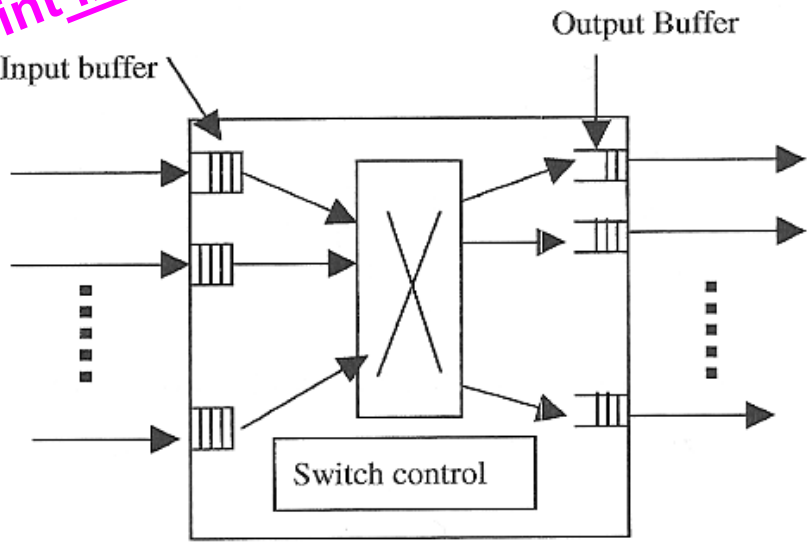
Similar to:
- Road system
- Telephone system

Computing Module **Network switch** **Network link**

NoC essentials



point-to-point link



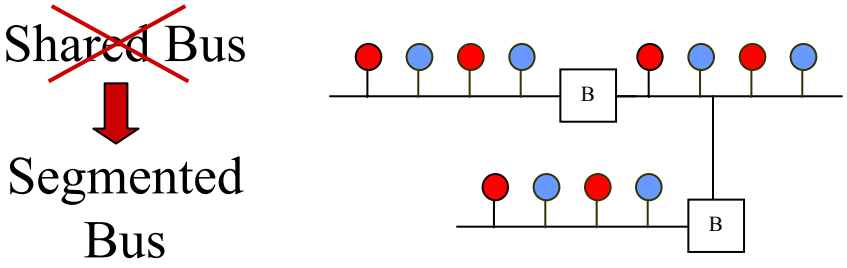
Switch (a.k.a. Router)

- Communication by packets of bits
- Routing of packets through several hops, via switches
- Parallelism
- Efficient sharing of wires

Origins of the NoC concept

- **The idea was talked about in the 90's, but actual research came in the new Millenium.**
- **Some well-known early publications:**
  - Guerrier and Greiner (2000)
    – *"A generic architecture for on-chip packet-switched interconnections"*
  - Hemani et al. (2000)
    – *"Network on chip: An architecture for billion transistor era"*
  - Dally and Towles (2001)
    – *"Route packets, not wires: on-chip interconnection networks"*
  - Wingard (2001)
    – *"MicroNetwork-based integration of SoCs"*
  - Rijpkema, Goossens and Wielage (2001)
    – "*A router architecture for networks on silicon"*
  - Kumar et al. (2002)
    – "*A Network on chip architecture and design methodology"*
  - De Micheli and Benini (2002)
    – *"Networks on chip: A new paradigm for systems on chip design"*

From buses to networks

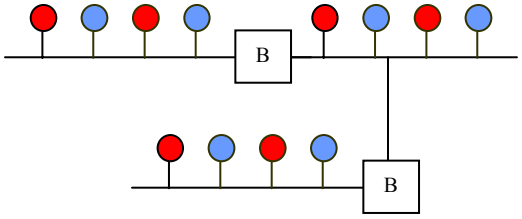


Original bus features:

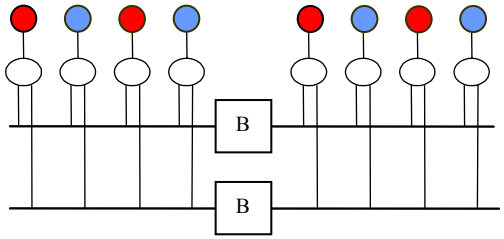
- One transaction at a time
- Central Arbiter
- Limited bandwidth
- Synchronous
- Low cost

Advanced bus

Segmented Bus



Multi-Level Segmented Bus



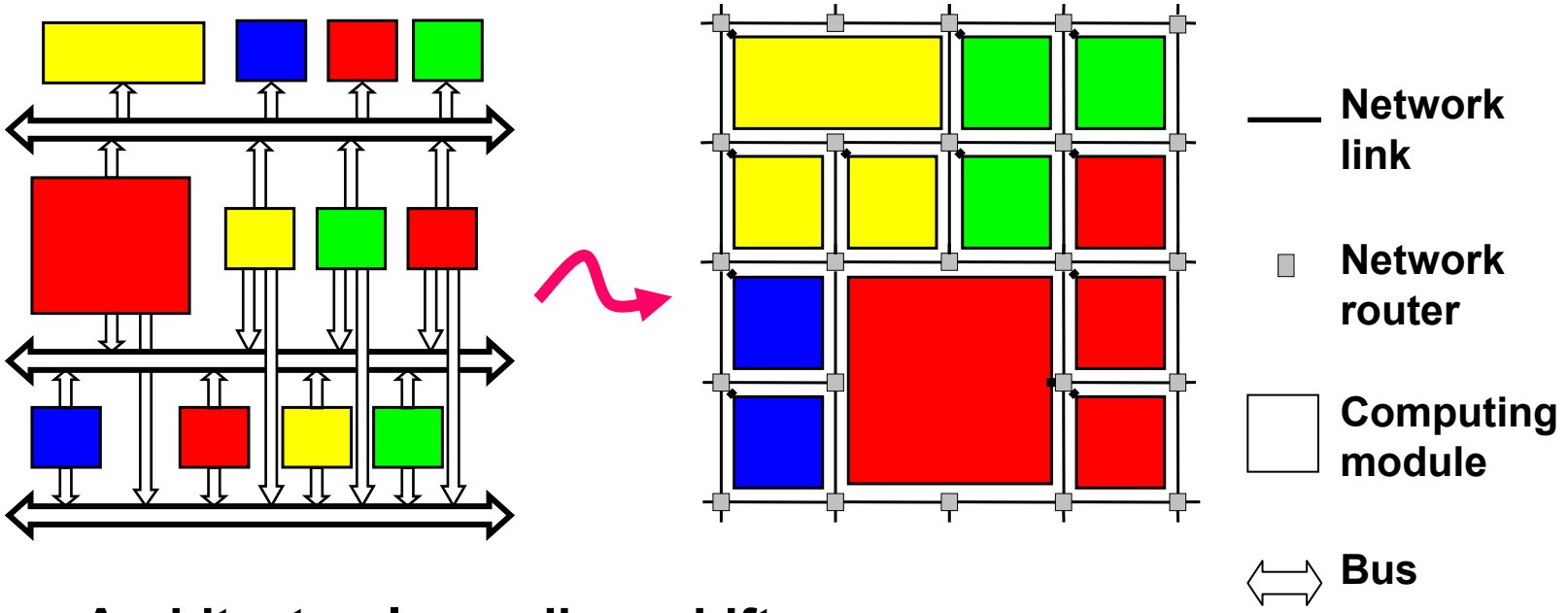
Original bus features:

- One transaction at a time
- Central Arbiter
- Limited bandwidth
- Synchronous
- Low cost

New features:

- Versatile bus architectures
- Pipelining capability
- Burst transfer
- Split transactions
- Overlapped arbitration
- Transaction preemption and resumption
- Transaction reordering…

Evolution or Paradigm Shift?

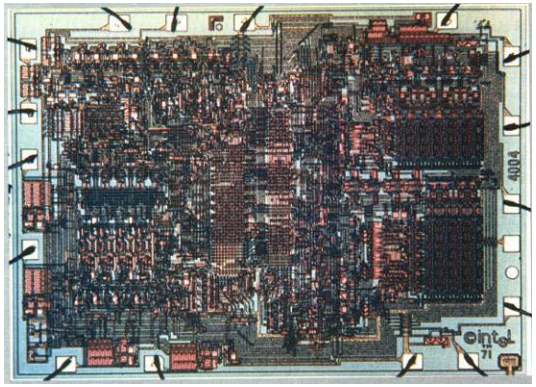


- Architectural paradigm shift
  - Replace the wire spaghetti by a network
- Usage paradigm shift
  - Pack everything in packets
- Organizational paradigm shift
  - Confiscate communications from logic designers
  - Create a new discipline, a new infrastructure responsibility
    - Already done for power grid, clock grid, …

Past examples of paradigm shifts in VLSI

The Microprocessor

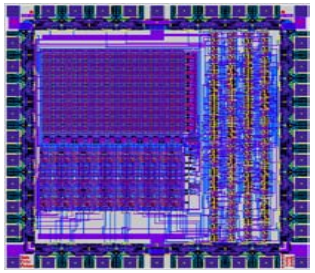
From: Hard-wired state machines
To: Programmable chips



- Created a new computer industry

Logic Synthesis

From: Schematic entry
To: HDLs and Cell libraries



- Logic designers became programmers
- Enabled ASIC industry and Fab-less companies
- "System-on-Chip"

Characteristics of a successful paradigm shift

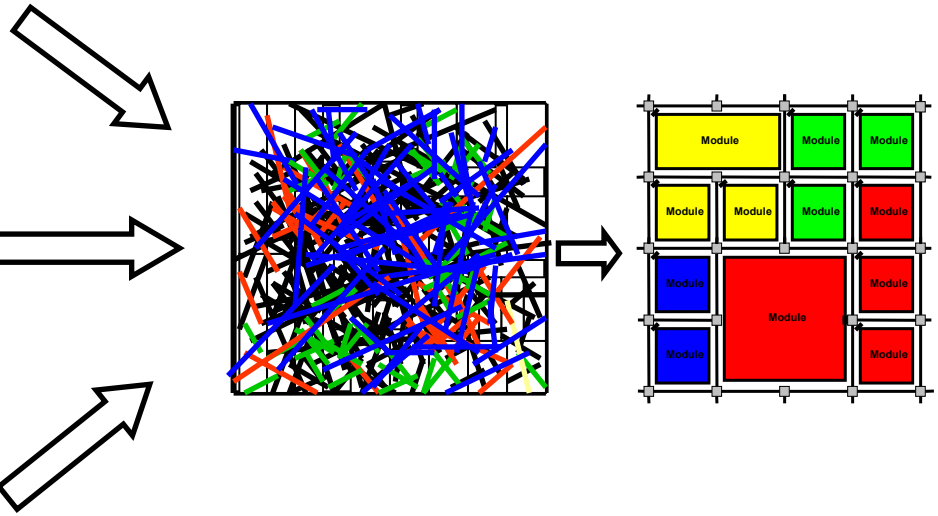
- Solves a critical problem (or several problems)
- Step-up in abstraction
- Design is affected:
  - Design becomes more restricted
  - New tools
  - The changes enable higher complexity and capacity
  - Jump in design productivity
- Initially: skepticism. Finally: change of mindset!

Let's look at the ***problems addressed by NoC***

Critical problems addressed by NoC

1) Global interconnect design problem:
delay, power, noise, scalability, reliability

2) System integration
productivity problem



3) Chip Multi Processors

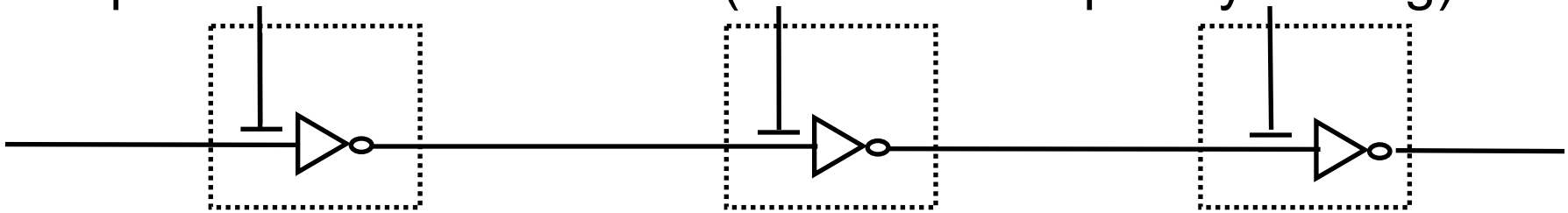
(key to power-efficient computing)

1(a): NoC and Global wire delay

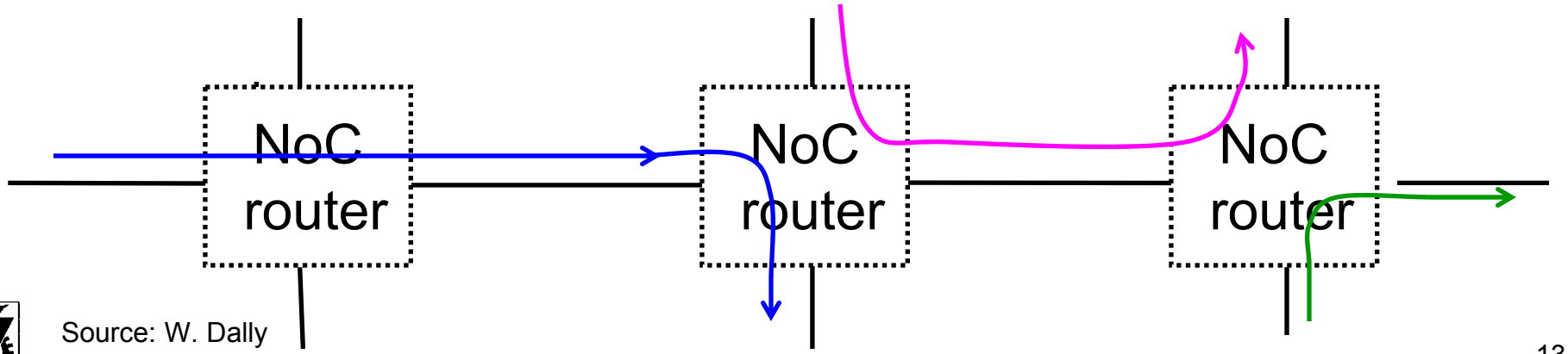
Long wire delay is dominated by Resistance

Add repeaters

Repeaters become latches (with clock frequency scaling)

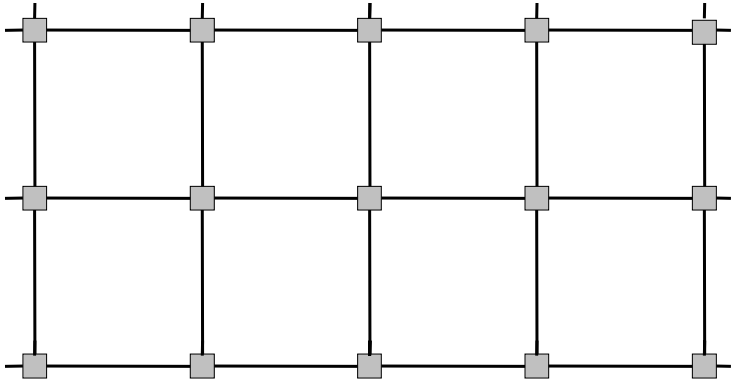


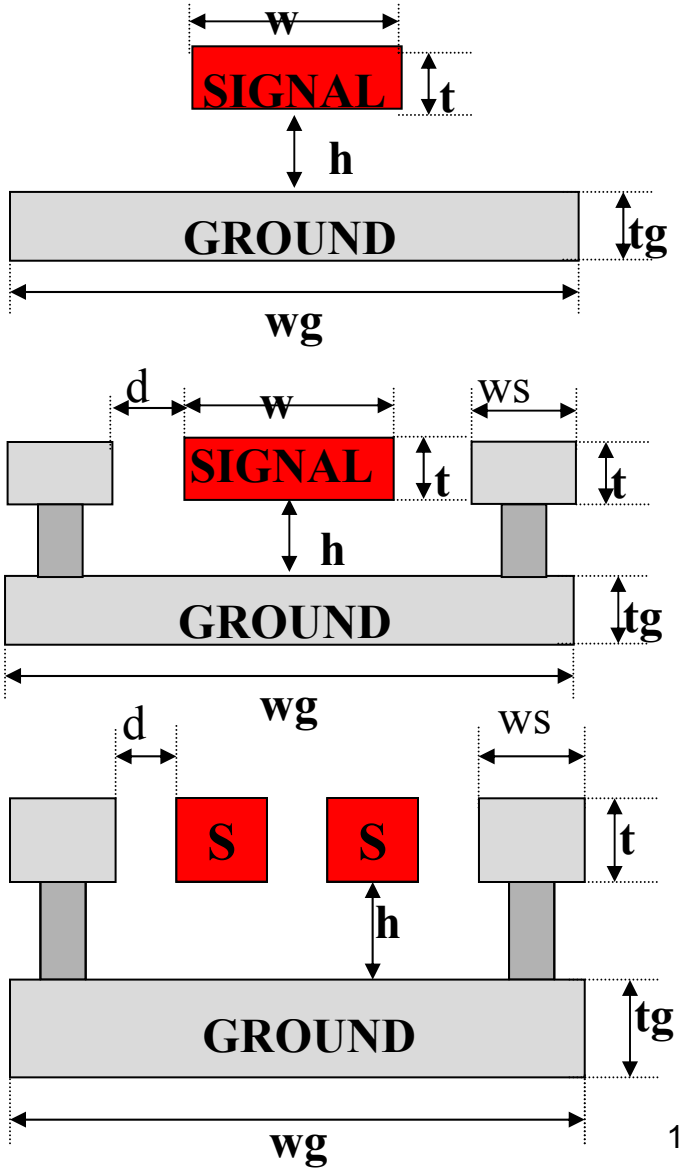
Latches evolve to NoC routers



Source: W. Dally

1(b): Wire Design for NoC



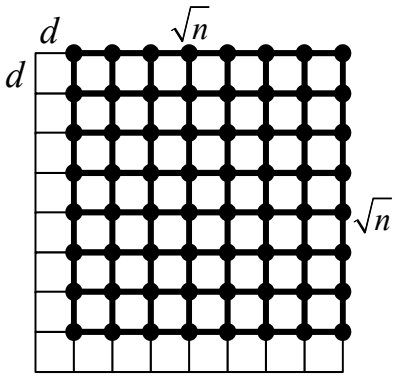


- NoC links:
  - Regular
  - Point-to-point (no fanout tree)
  - Can use transmission-line layout
  - Well-defined current return path
- Can be optimized for <u>noise / speed / power</u>
  - Low swing, current mode, ….

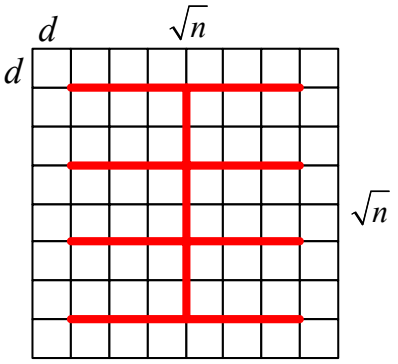
1(c): NoC Scalability

For Same Performance, compare the wire-area cost of:

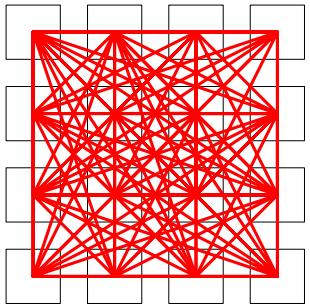
NoC: $O(n)$



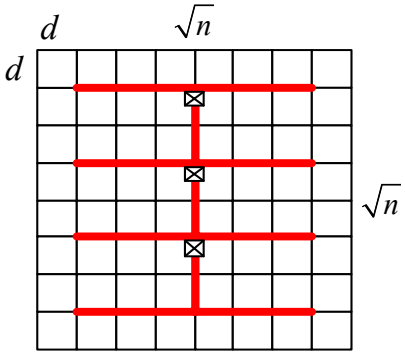
Simple Bus: $O(n^3\sqrt{n})$



Point-to-Point: $O(n^2\sqrt{n})$

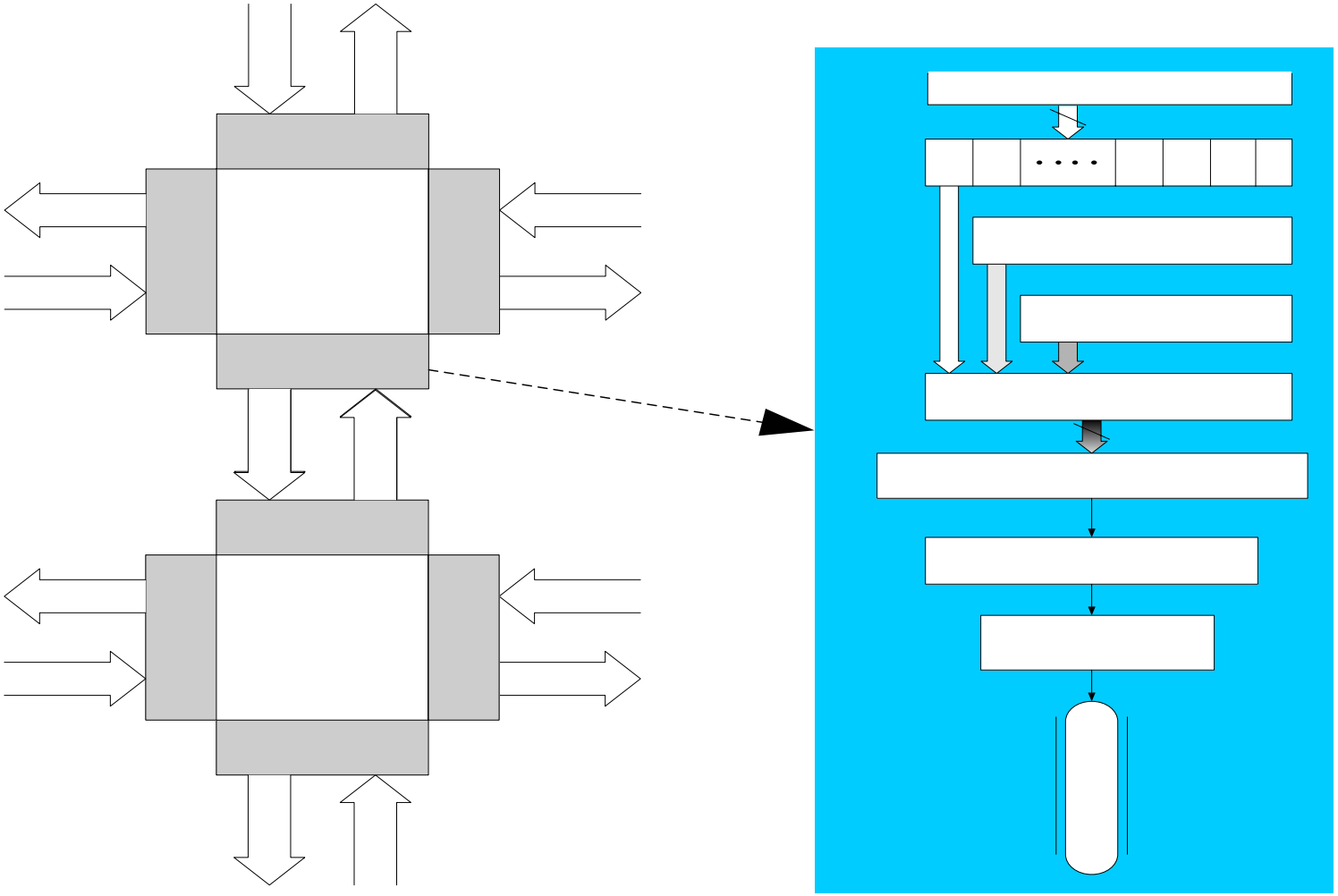


Segmented Bus: $O(n^2\sqrt{n})$



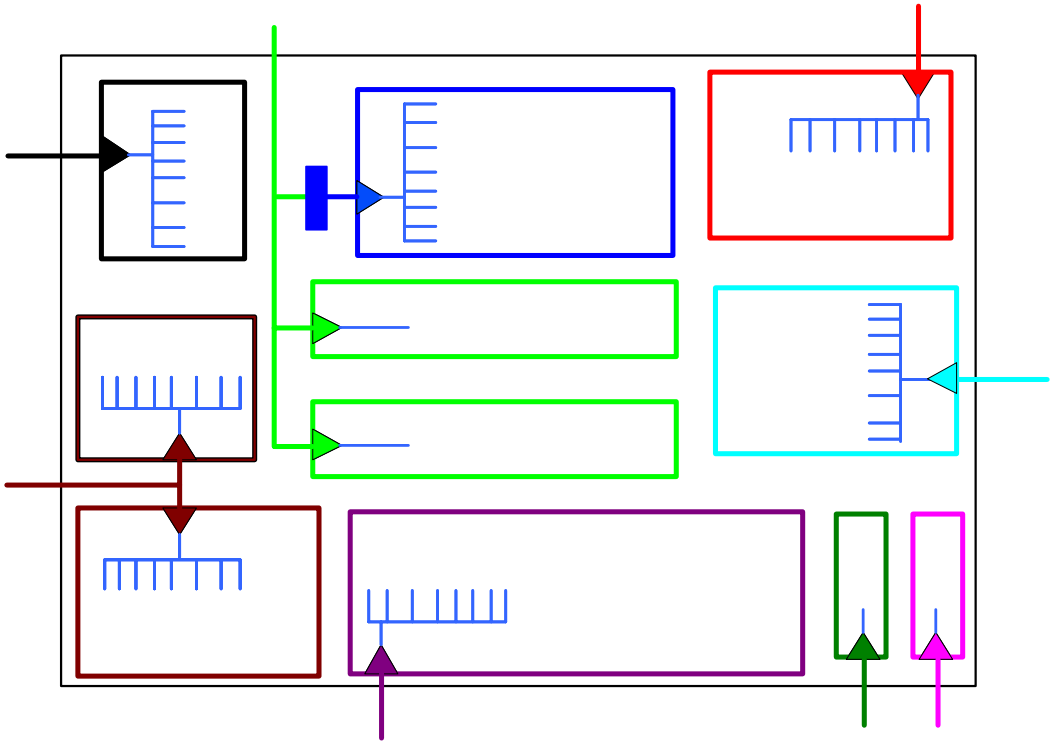
1(d): NoC and communication reliability

- **Fault tolerance and error correction**



1(e): NoC and GALS

- System modules may use different clocks
  - May use different voltages
- NoC can take care of synchronization
- NoC design may be asynchronous
  - No waste of power when the links and routers are idle



2: NoC and engineering productivity

- **NoC eliminates ad-hoc global wire engineering**
- **NoC separates computation from communication**
  - NoC supports modularity and reuse of cores
- **NoC is a platform for system integration, debugging and testing**

DATE 07

Call for Participation

DATE 2007 Friday Workshop on

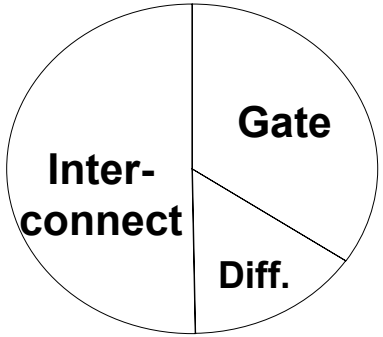
Diagnostic Services in Network-on-Chips

— Test, Debug, and On-Line Monitoring —

Palais des Congrès Acropolis — Nice, France
Friday April 20, 2007

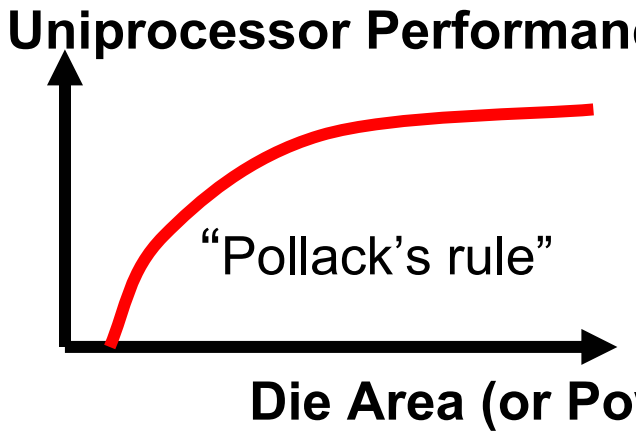
3: NoC and CMP

- **Uniprocessors cannot provide Power-efficient performance growth**
  - Interconnect dominates dynamic power
  - Global wire delay doesn't scale
  - Instruction-level parallelism is limited



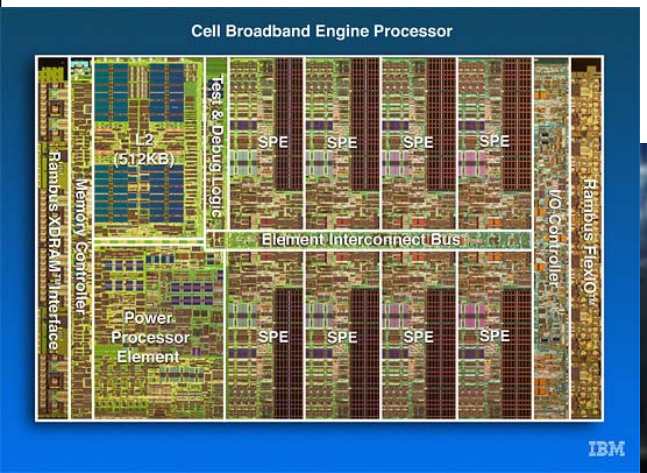
Uniprocessor dynamic power
(Magen et al., SLIP 2004)

- **Power-efficiency requires many parallel local computations**
  - Chip Multi Processors (CMP)
  - Thread-Level Parallelism (TLP)



(F. Pollack. Micro 32, 1999)

- **Network is a natural choice for CMP!**

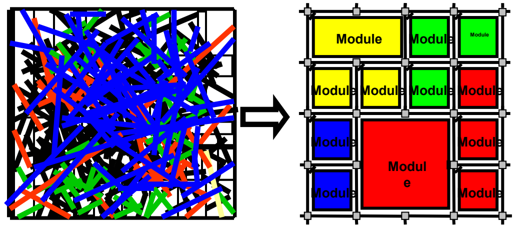


Why Now is the time for NoC?

Difficulty of DSM wire design

Productivity pressure

CMPs



successful

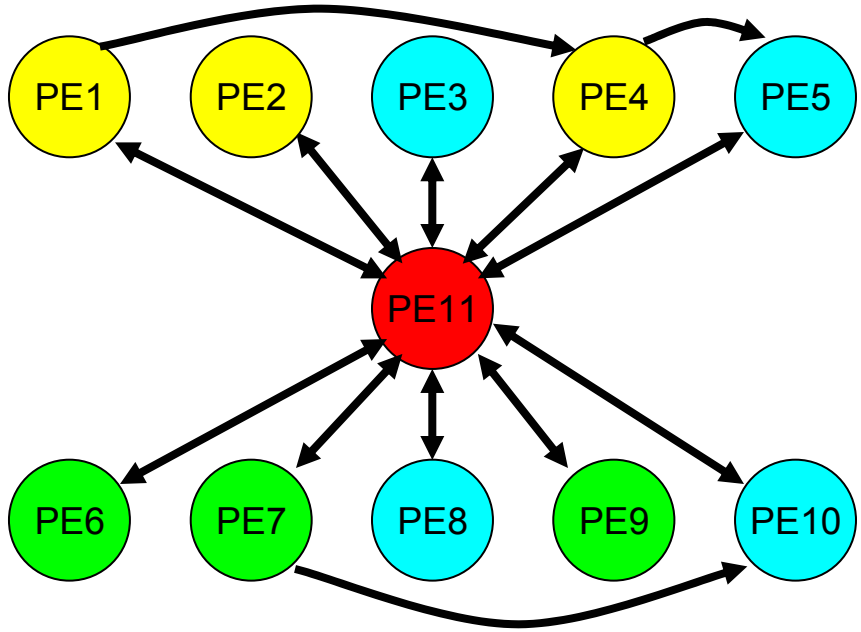
Characteristics of a paradigm shift

- Solves a critical problem (or several problems)
- Step-up in abstraction
- Design is affected:
  - Design becomes more restricted
  - New tools
  - The changes enable higher complexity and capacity
  - Jump in design productivity
- Initially: skepticism. Finally: change of mindset!

Now, let's look at the **Abstraction** provided by NoC

Traffic model abstraction

Flow	BW	Packet Size	Latency
1⇨4	500Kb/s	1Kb	5nsec
4⇨5	1.5Mb/s	3Kb	12nsec
7⇨10	200Kb/s	2Kb	5nsec
1⇔11	50Kb/s	2Kb	15nsec
2⇔11	50Kb/s	1Kb	22nsec
3⇔11	300Kb/s	3Kb	15nsec
4⇔11	1.5Mb/s	5Kb	22nsec
5⇔11	50Kb/s	1Kb	12nsec
6⇔11	300Kb/s	1Kb	22nsec
7⇔11	1.5Mb/s	5Kb	5nsec
8⇔11	50Kb/s	1.5Kb	12nsec
9⇔11	300Kb/s	2Kb	15nsec
10⇔11	1.5Mb/s	3Kb	12nsec



- Traffic model may be captured from actual traces of functional simulation
- A statistical distribution is often assumed for messages

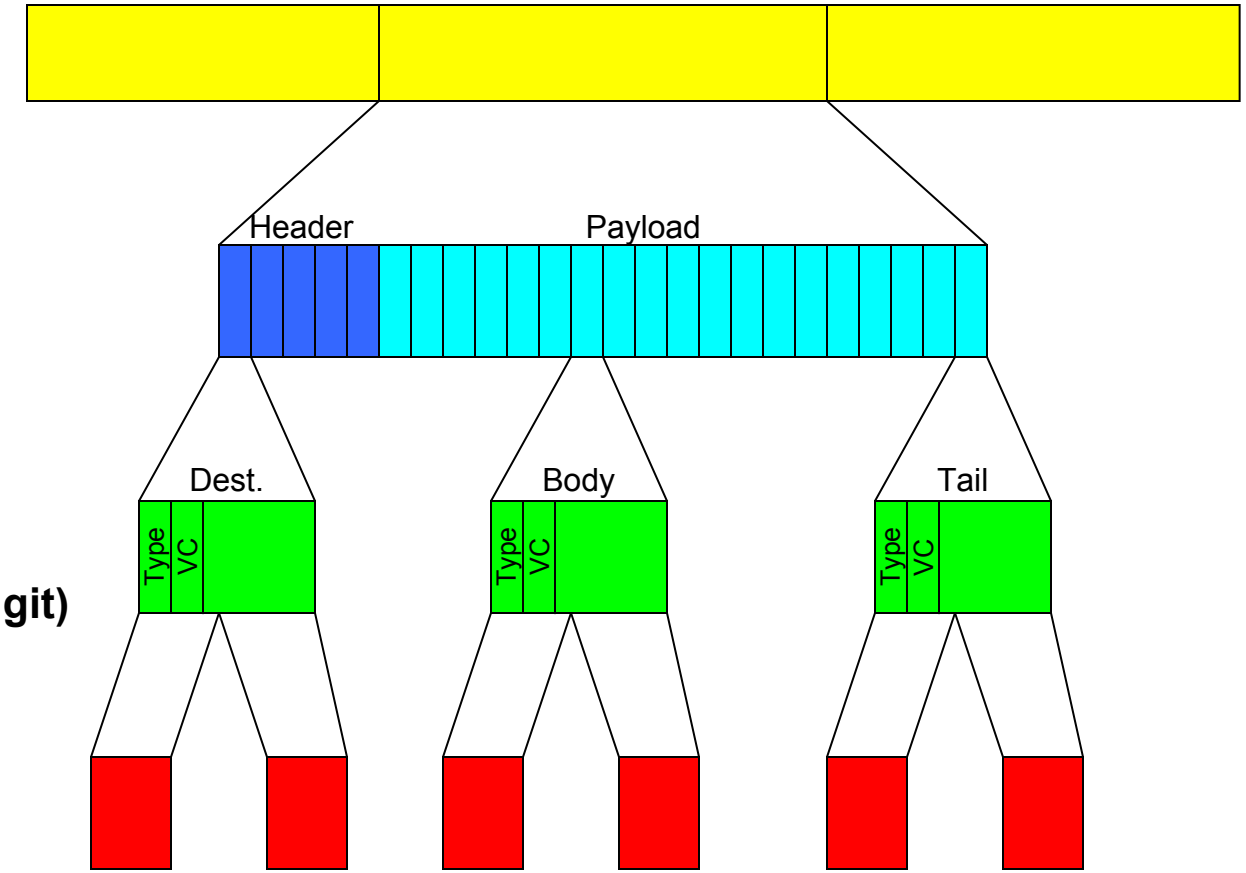
Data abstraction

Message

Packet

Flit
(Flow control digit)

Phit
(Physical unit)

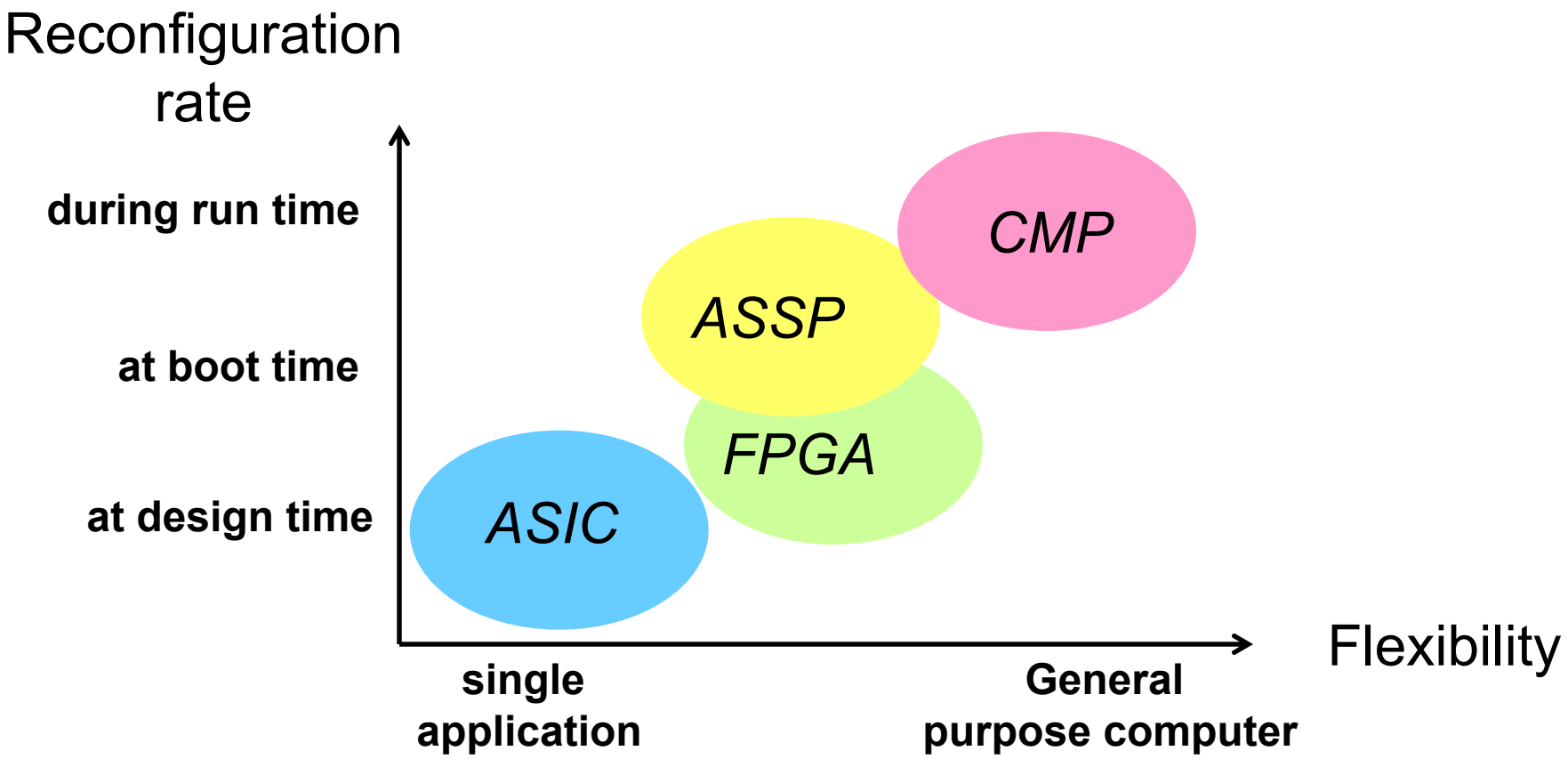


Layers of Abstraction in Network Modeling

- **Software layers**
  - O/S, application
- **Network and transport layers**
  - Network topology e.g. crossbar, ring, mesh, torus, fat tree,…
  - Switching Circuit / packet switching: SAF, VCT, wormhole
  - Addressing Logical/physical, source/destination, flow, transactio
  - Routing Static/dynamic, distributed/source, deadlock avoidance
  - Quality of Service e.g. guaranteed-throughput, best-effort
  - Congestion control, end-to-end flow control
- **Data link layer**
  - Flow control (handshake)
  - Handling of contention
  - Correction of transmission errors
- **Physical layer**
  - Wires, drivers, receivers, repeaters, signaling, circuits,..

Let's skip a tutorial here, and look at an **example**

Architectural choices depend on system needs

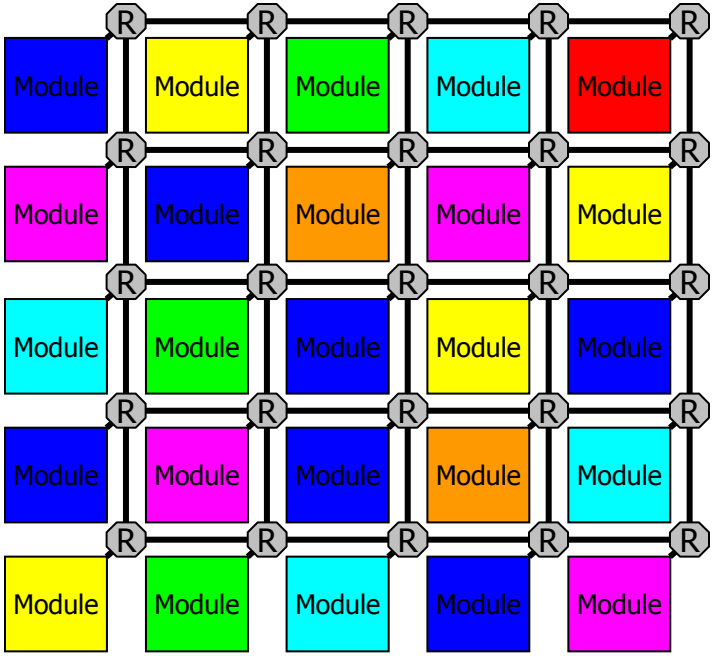


- **A large design space for NoCs!**

Example: QNoC

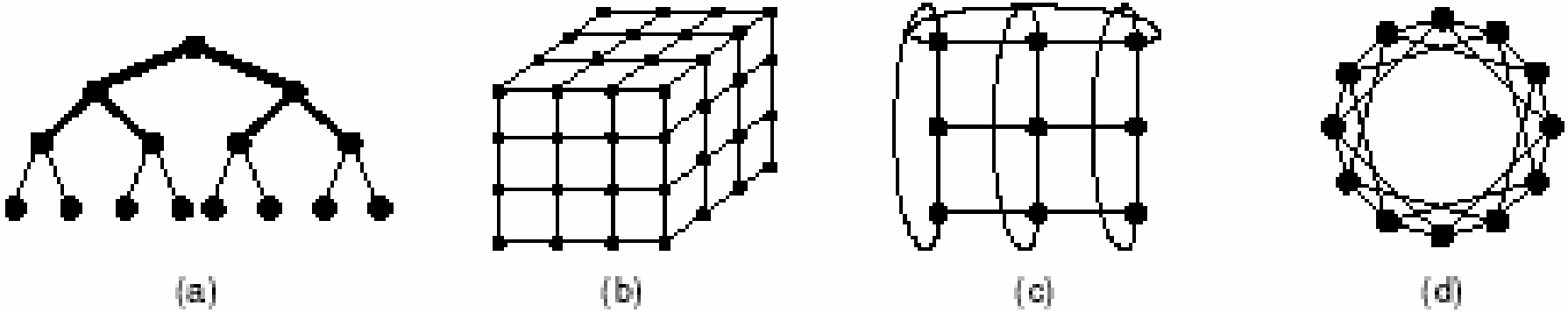
Technion's Quality-of-service NoC architecture

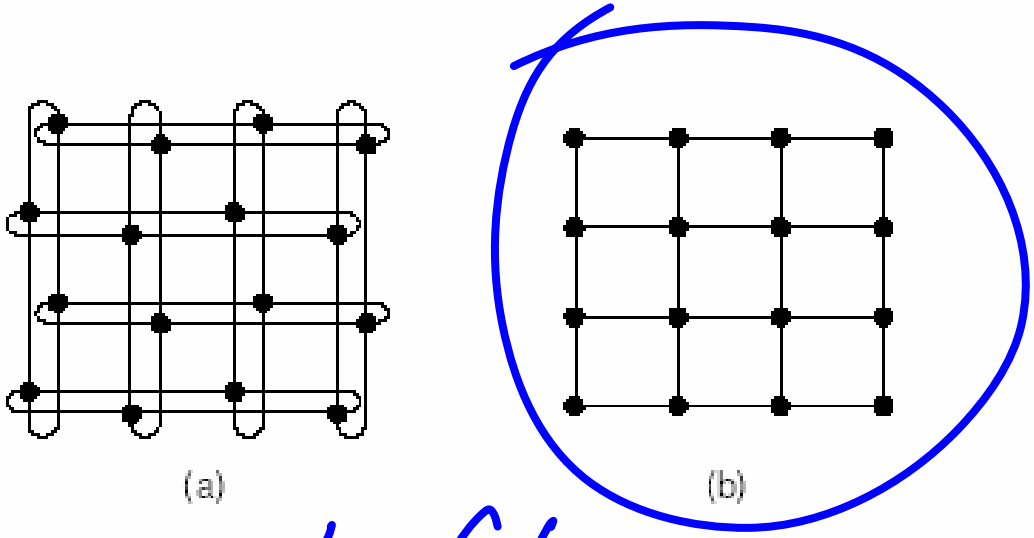
- **Application-Specific system (ASIC) assumed**
  - **~10 to 100 IP cores**
  - **Traffic requirements are known a-priori**
- **Overall approach**
  - **Packet switching**
  - **Best effort ("statistical guarantee")**
  - **Quality of Service (priorities)**



* E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny., **"QNoC: QoS architecture and design process for Network on Chip"**, ***JSA*** special issue on NoC, 2004.

Choice of generic network topology

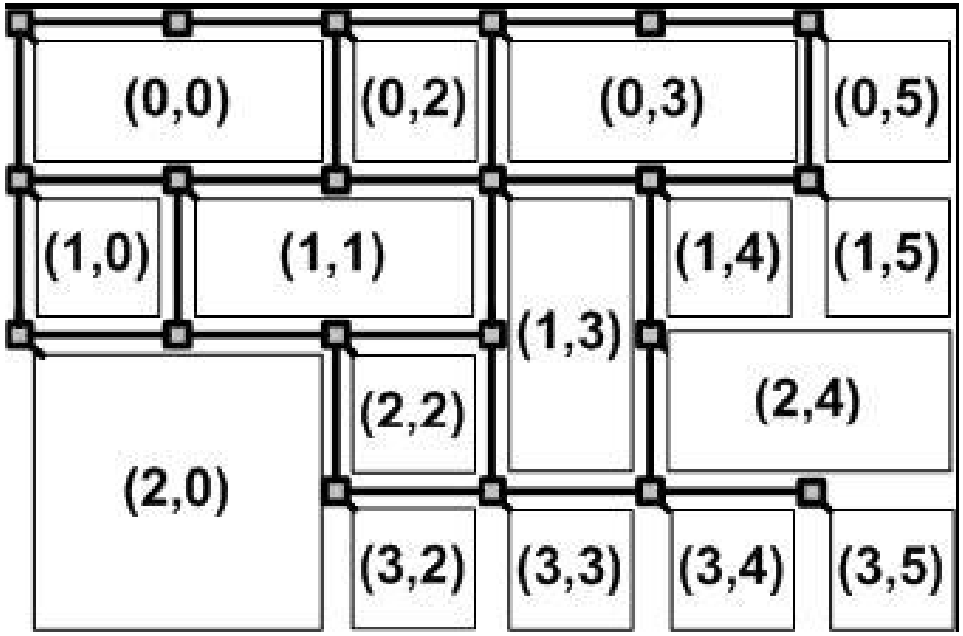




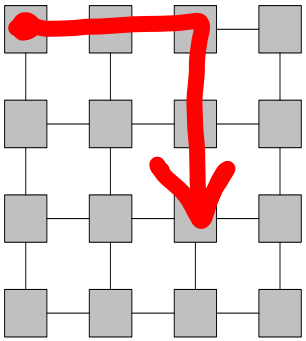
- Simple mesh fits planar chip
- Short links

Topology customization

- **Irregular mesh**
  - **Address = coordinates in the basic grid**



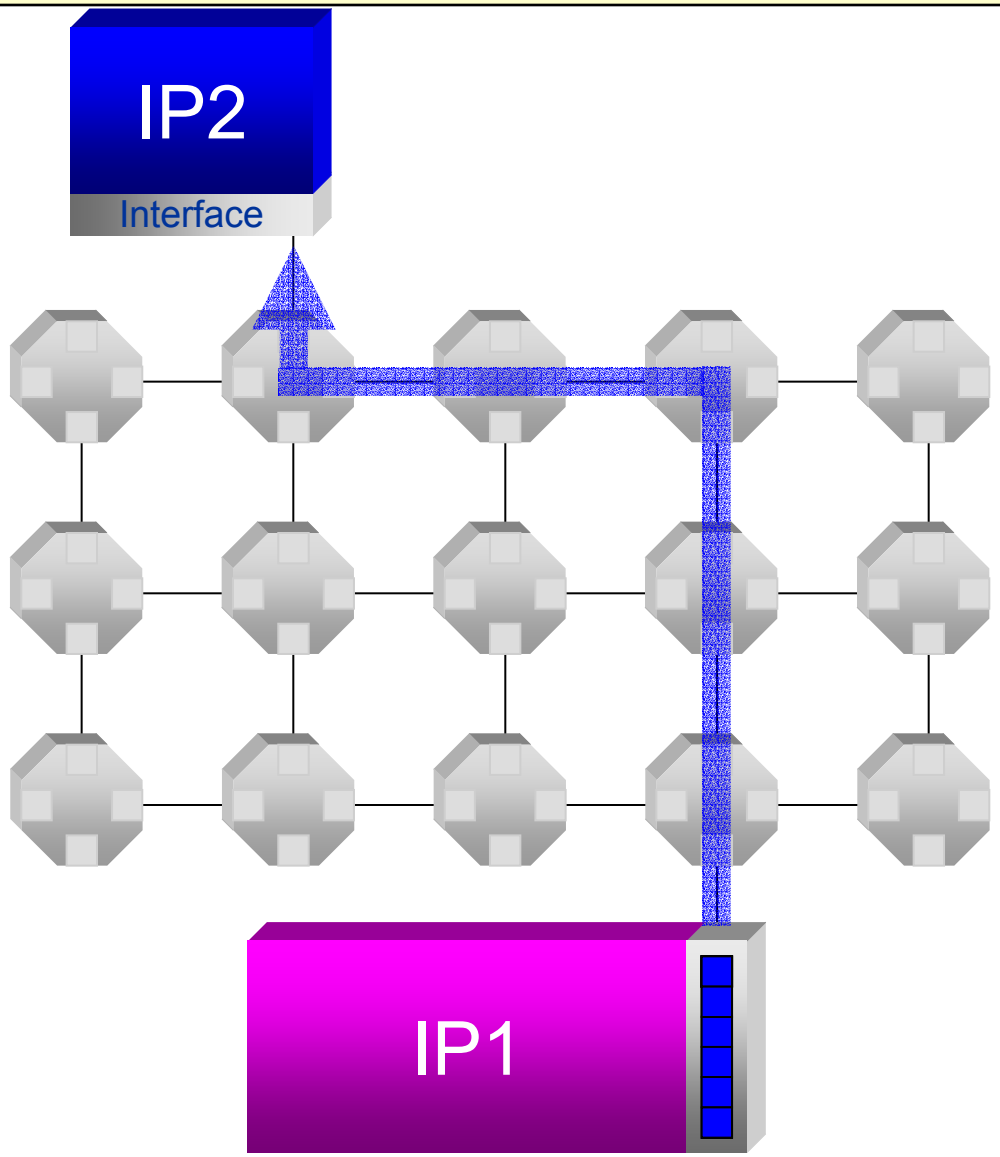
Message routing path



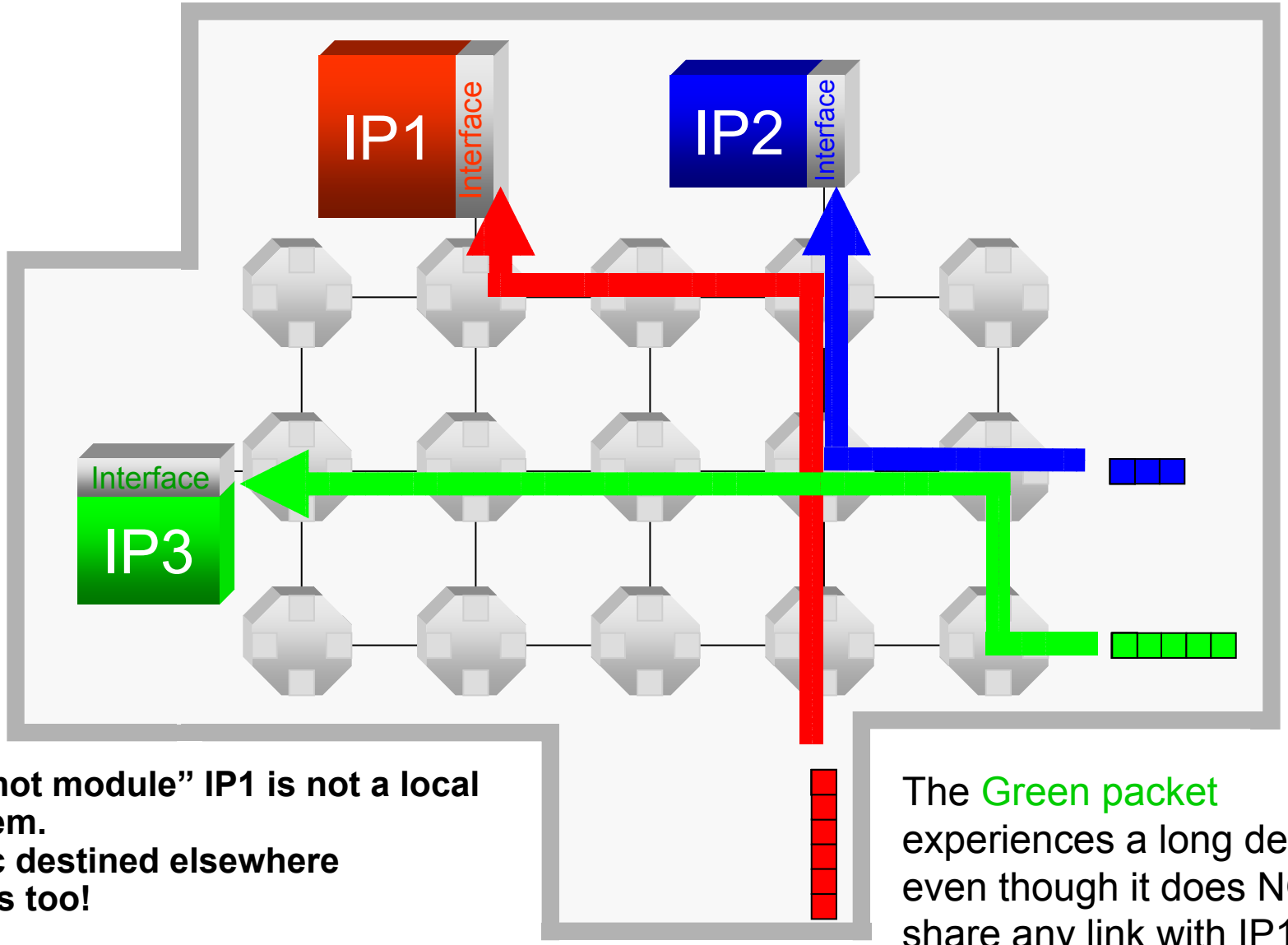
- **Fixed shortest-path routing (X-Y)**
  - ✓ **Simple Router**
  - ✓ **No deadlock scenario**
  - ✓ **No retransmission**
  - ✓ **No reordering of messages**
  - ✓ **Power-efficient**

Wormhole Switching

- Small number of buffers
- Low latency



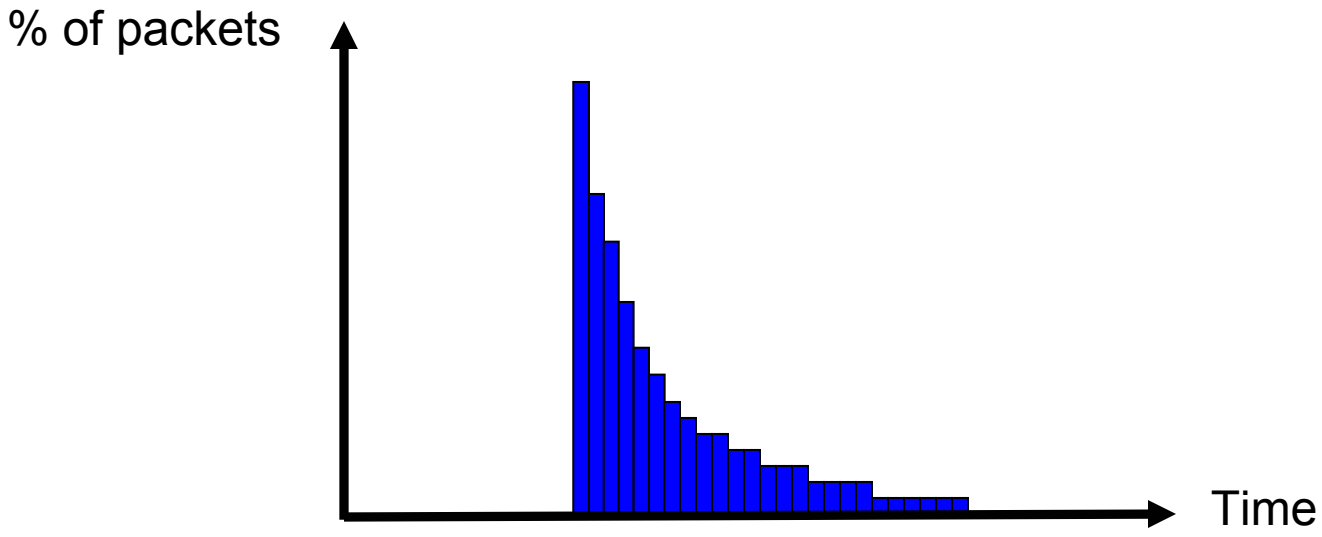
Blocking issue



- **The “hot module” IP1 is not a local problem.**
  Traffic destined elsewhere suffers too!

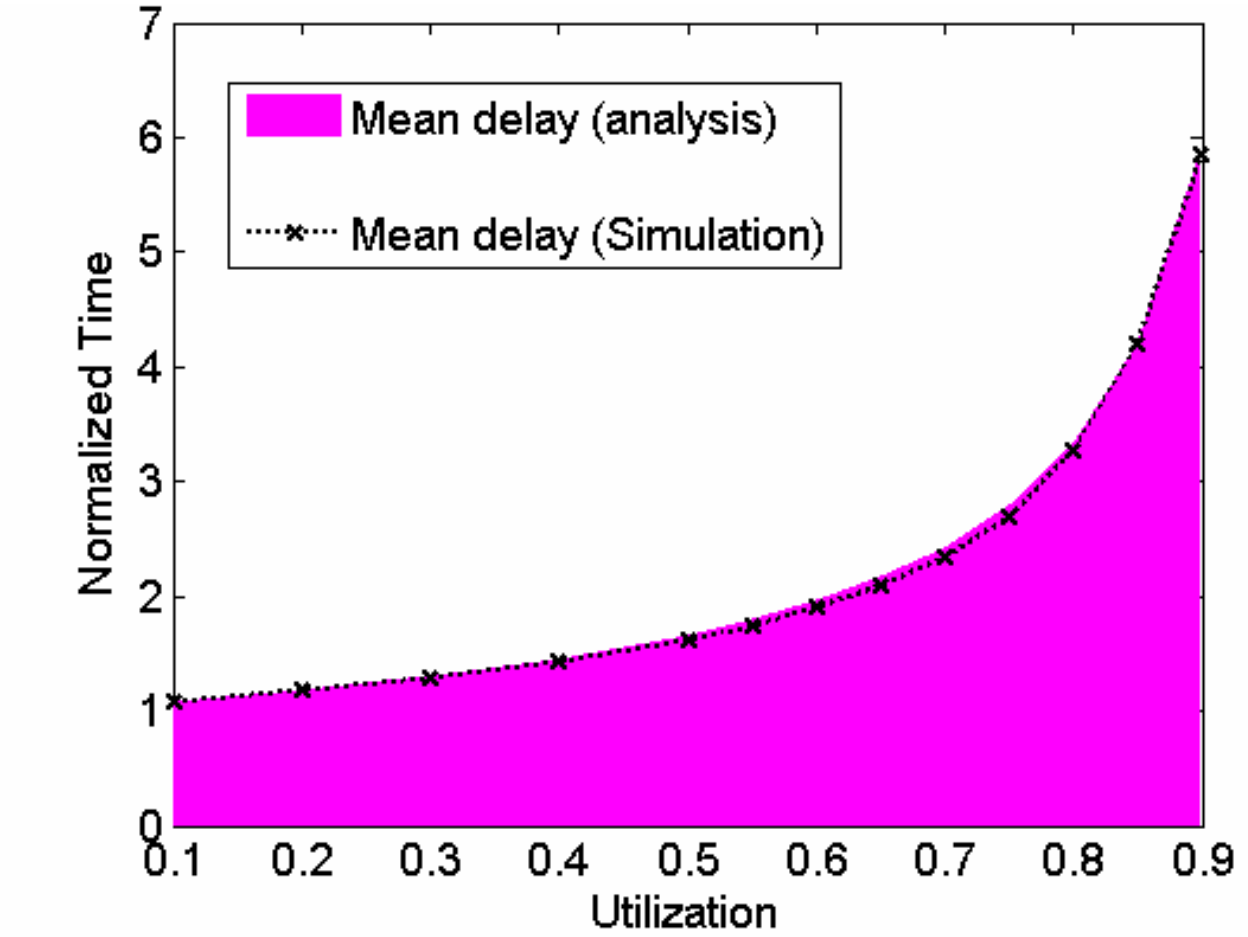
The Green packet experiences a long delay even though it does NOT share any link with IP1 traffic

Statistical network delay



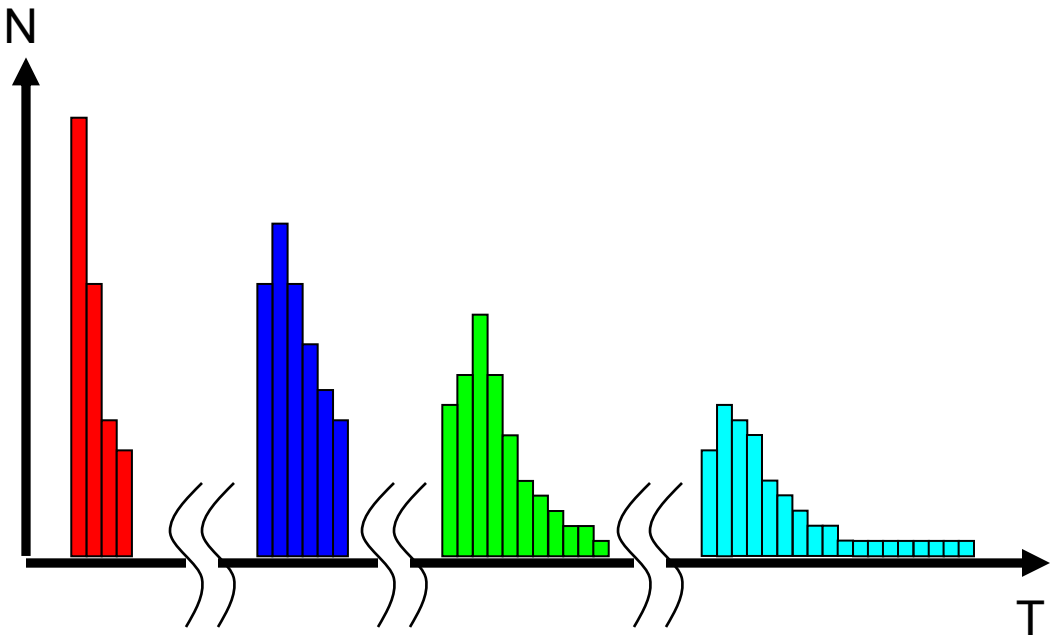
- **Some packets get more delay than others, because of blocking**

Average delay depends on load



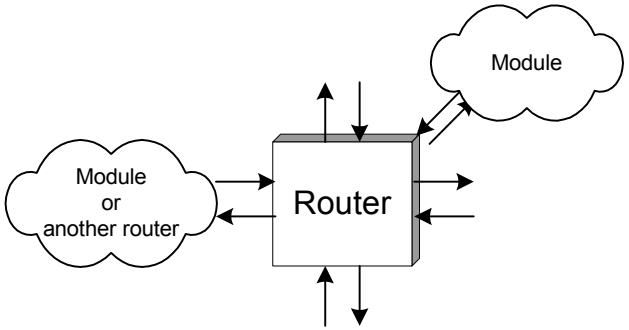
Quality-of-Service in QNoC

- **Multiple priority (service) levels**
  - **Define latency / throughput**
  - **Example:**
    - **Signaling**
    - **Real Time Stream**
    - **Read-Write**
    - **DMA Block Transfer**
  - **Preemptive**
- **Best effort performance**
  - **E.g. 0.01% arrive later then required**

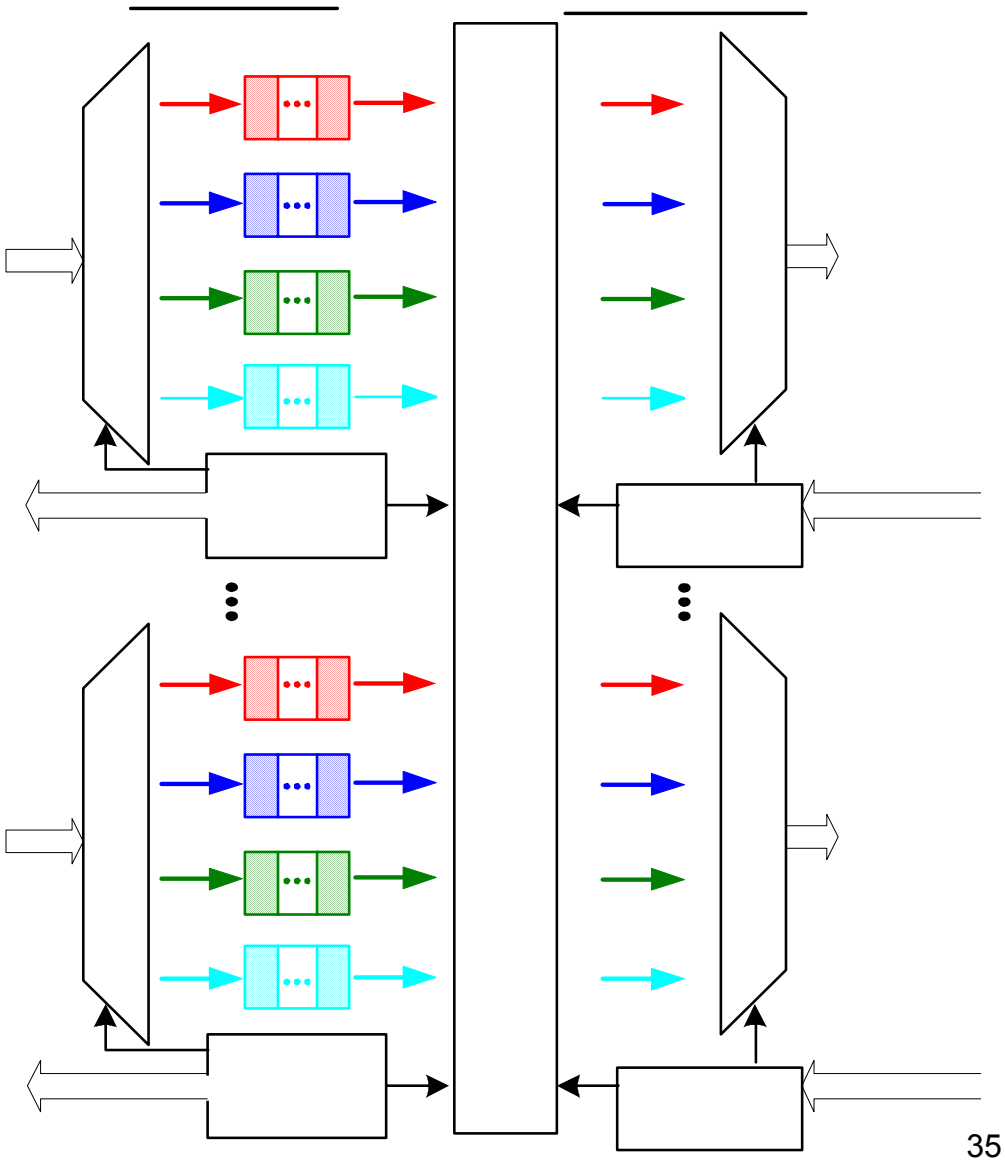


* E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny., **"QNoC: QoS architecture and design process for Network on Chip",** JSA special issue on NOC, 2004.

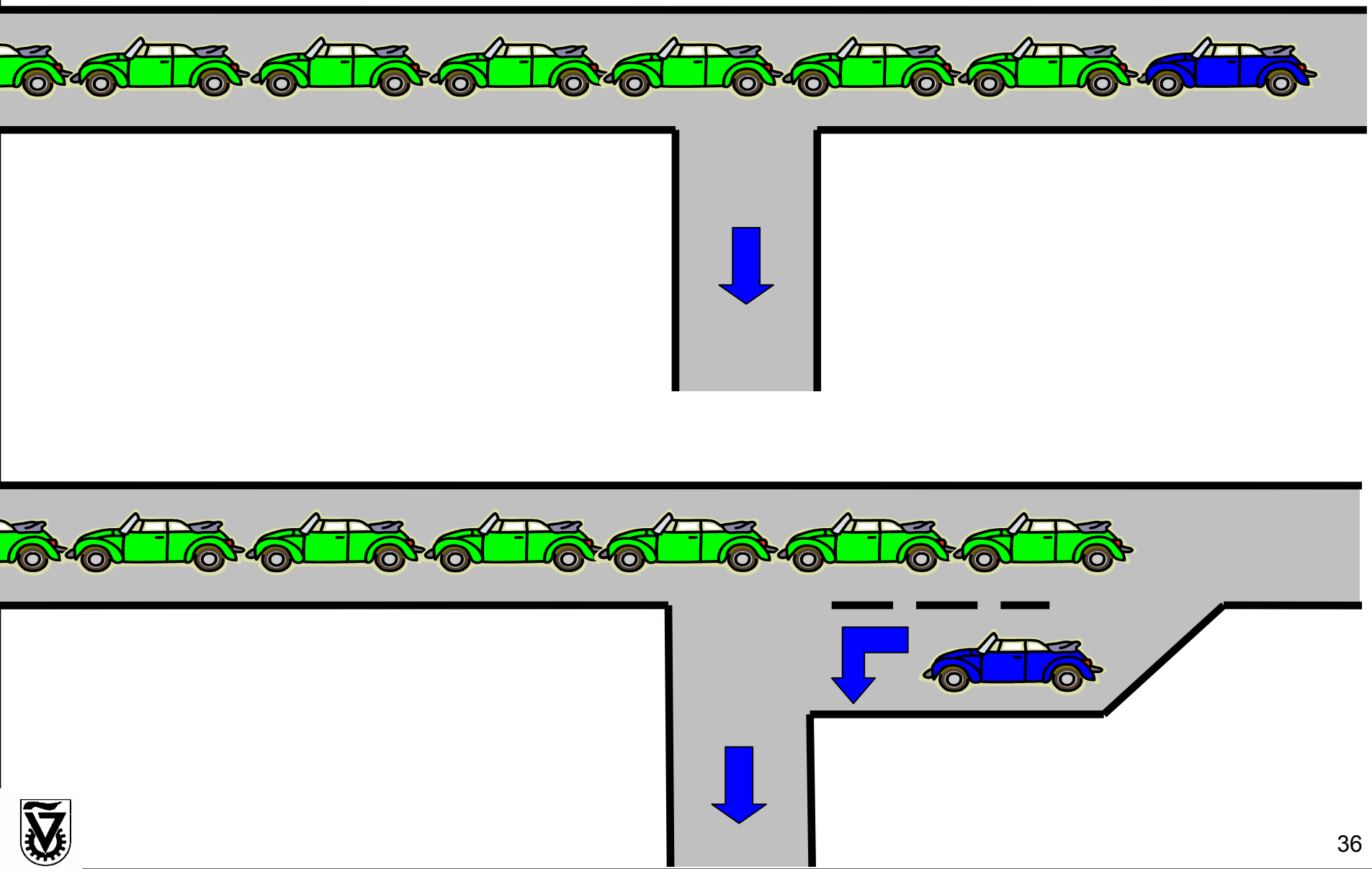
Router structure



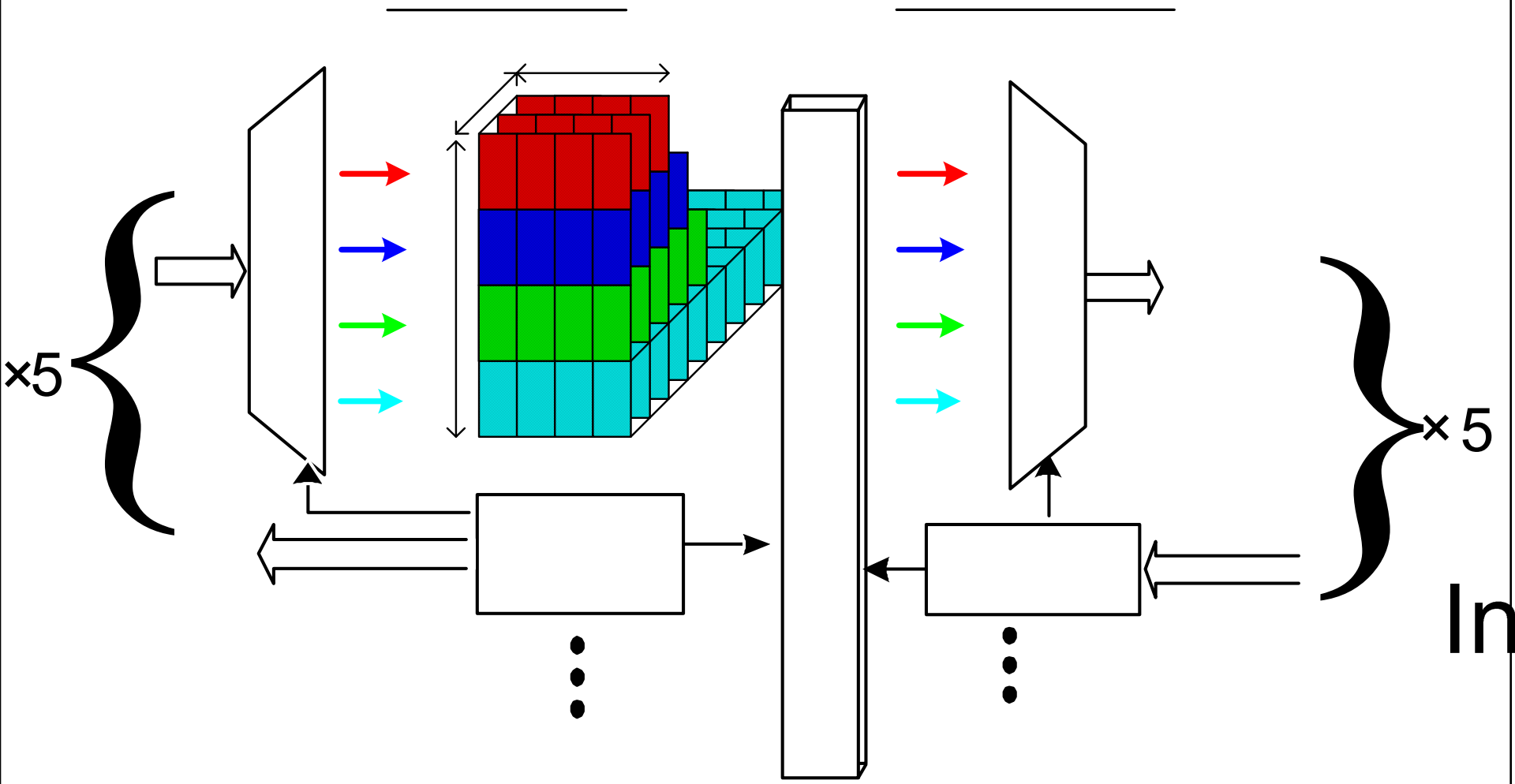
- Flits stored in input ports
- Output port schedules transmission of pending flits according to:
  - Priority (*Service Level*)
  - Buffer space in next router
  - Round-Robin on input ports of same SL
  - Preempt lower priority packets



Virtual Channels

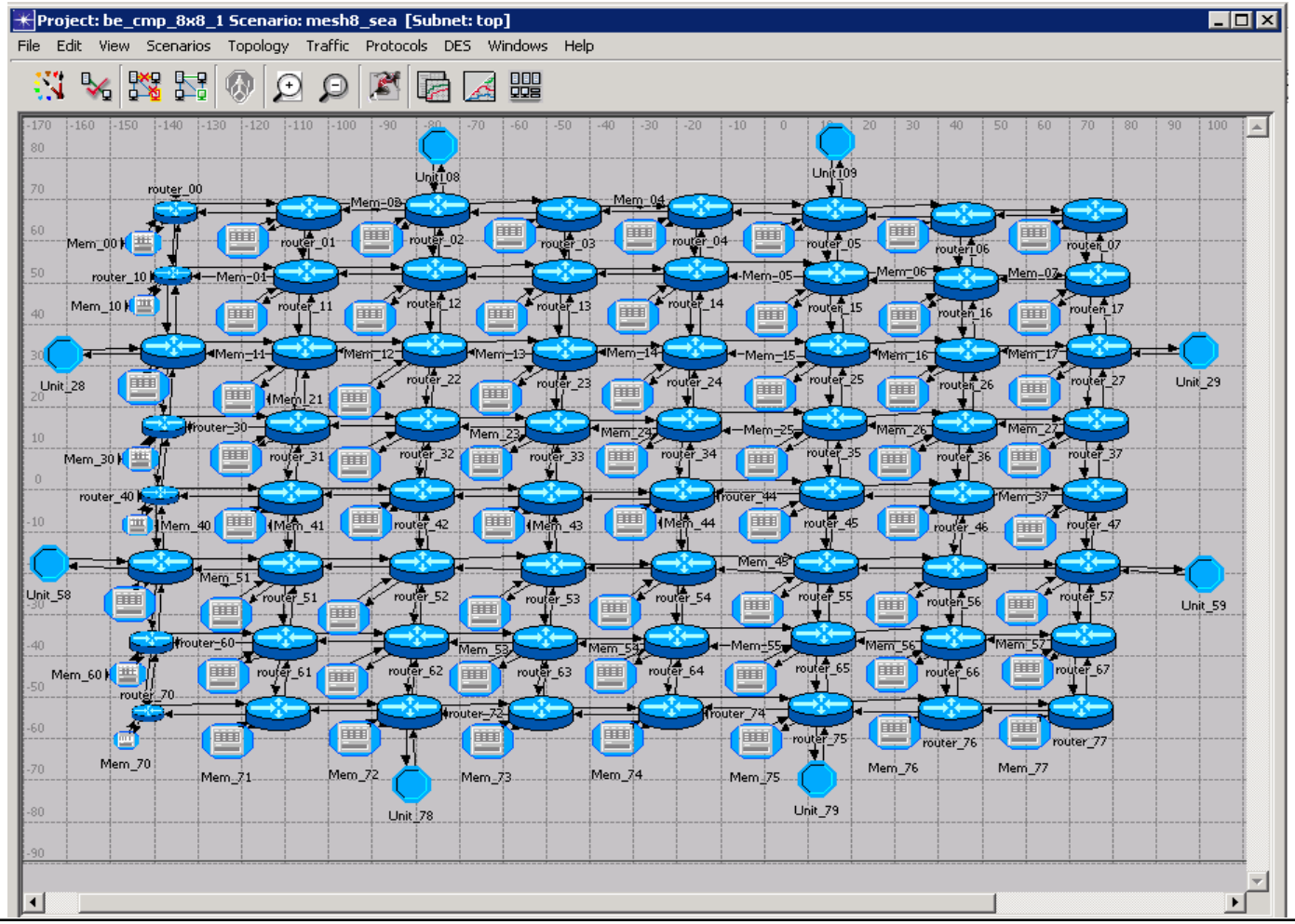


QNoC router with multiple Virtual Channels



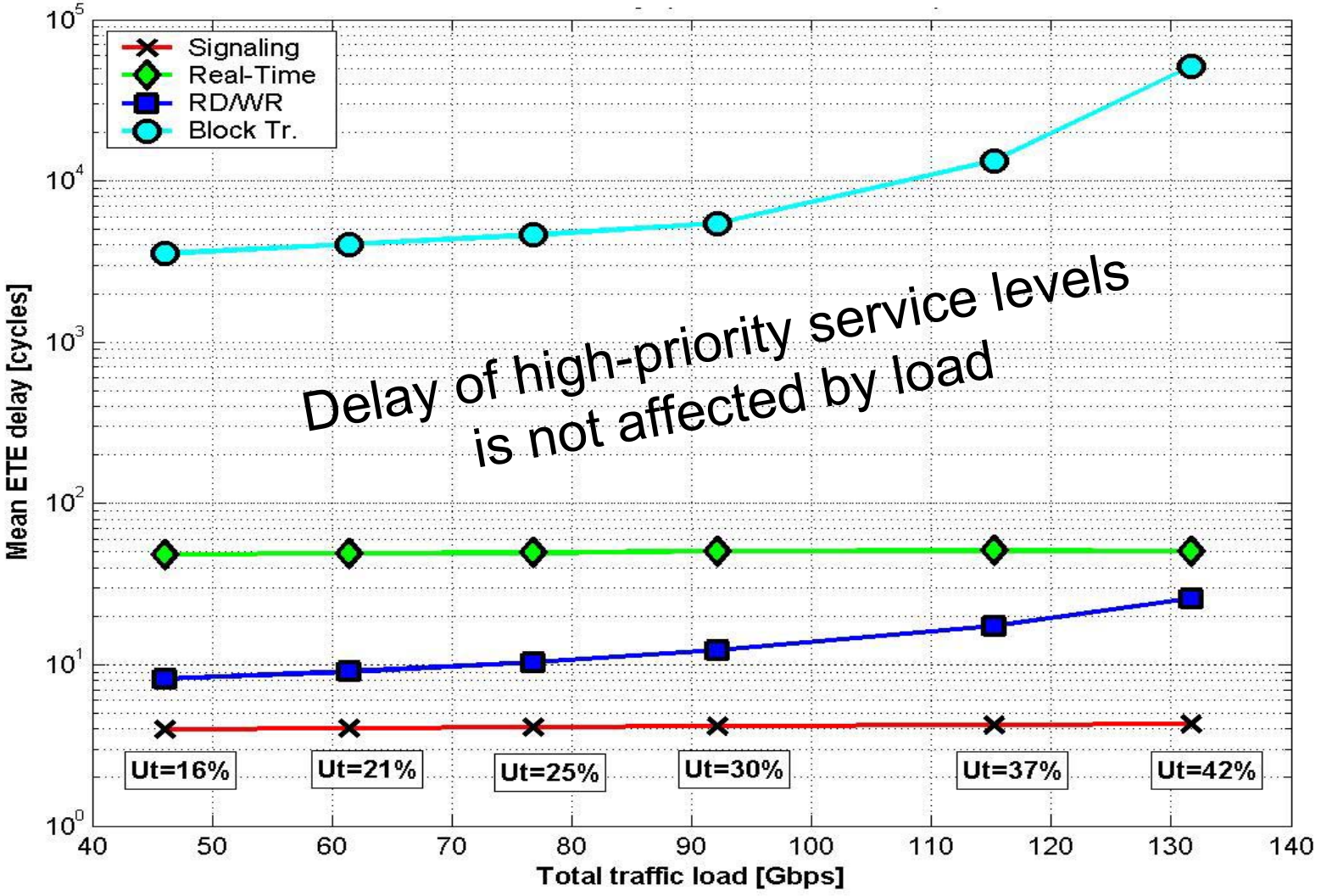
Simulation Model

- OPNET Models for QNoC
- Any topology and traffic load
- Statistical or trace-based traffic generation at source nodes



Simulation Results

- **Flit-accurate simulations**



Perspective 1: NoC vs. Bus

NoC

- Aggregate bandwidth grows
- Link speed unaffected by N
- Concurrent spatial reuse
- Pipelining is built-in
- Distributed arbitration
- Separate abstraction layers

However:

- No performance guarantee
- Extra delay in routers
- Area and power overhead?
- Modules need network interface
- Unfamiliar methodology

Bus

- Bandwidth is limited, shared
- Speed goes down as N grows
- No concurrency
- Pipelining is tough
- Central arbitration
- No layers of abstraction (communication and computation are coupled)

However:

- Fairly simple and familiar

Perspective 2: NoC vs. Off-chip Networks

NoC

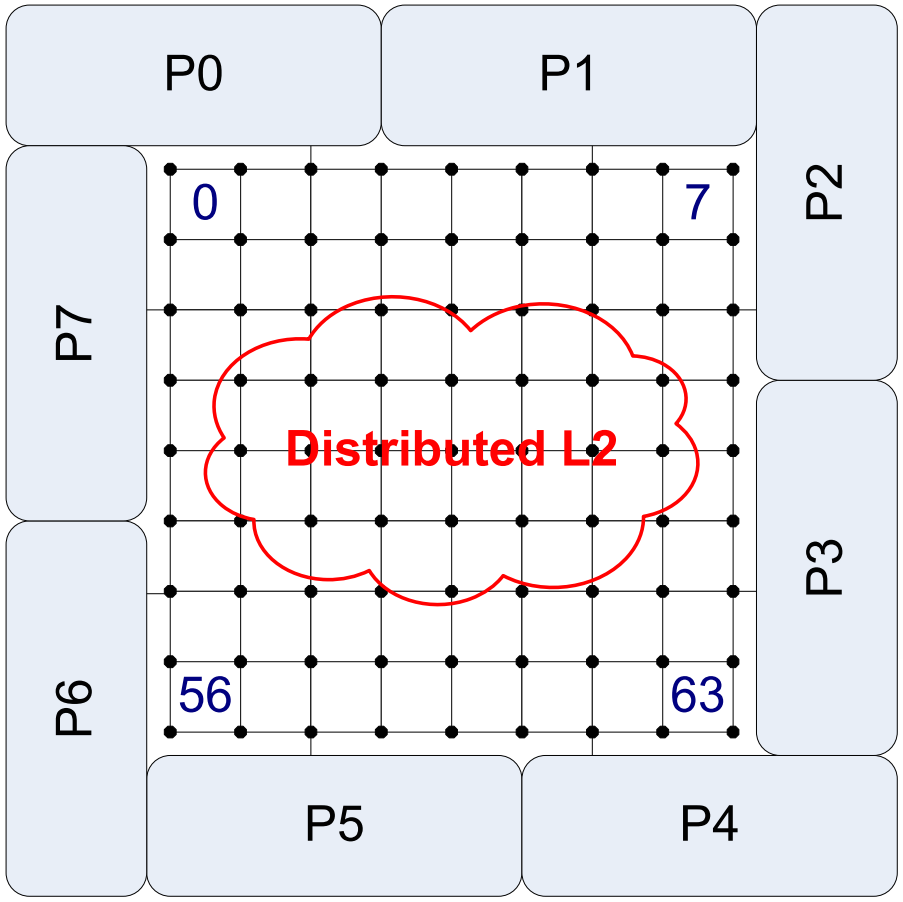
- Sensitive to cost:
  - area
  - power
- Wires are relatively cheap
- Latency is critical
- Traffic may be known a-priori
- Design time specialization
- Custom NoCs are possible

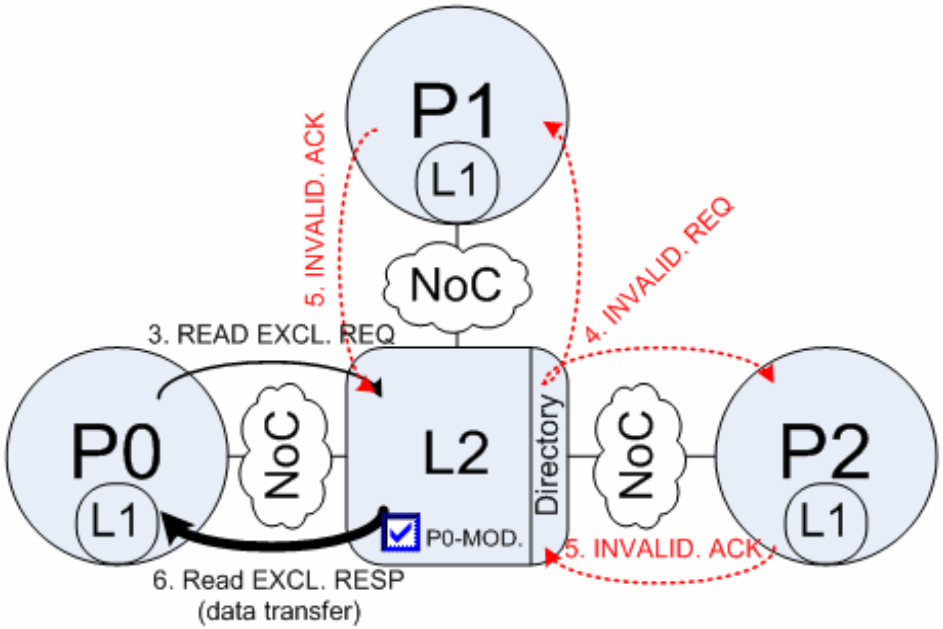
Off-Chip Networks

- Cost is in the links
- Latency is tolerable
- Traffic/applications unknown
- Changes at runtime
- Adherence to networking standards

NoC can provide system services

Example: Distributed CMP cache





* *E.Bolotin, Z. Guz, I.Cidon, R. Ginosar and A. Kolodny,* "**The Power of Priority: NoC based Distributed Cache Coherency**", ***NoCs 2007***.

Characteristics of a successful paradigm shift

- Solves a critical problem (or several problems)
- Step-up in abstraction
- Design is affected:
  - Design becomes more restricted
  - New tools
  - The changes enable higher complexity and capacity
  - Jump in design productivity
- Initially: skepticism. Finally: change of mindset!

OK, what's the impact of NoC on chip **design**?

VLSI CAD problems

- Application mapping
- Floorplanning / placement
- Routing
- Buffer sizing
- Timing closure
- Simulation
- Testing

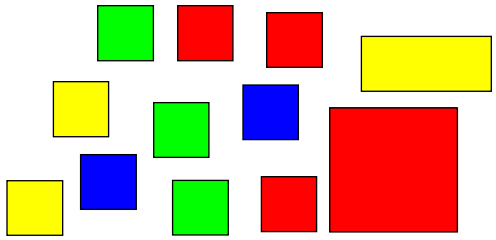
VLSI CAD problems reframed for NoC

- Application mapping (map tasks to cores)
- Floorplanning / placement (within the network)
- Routing (of messages)
- Buffer sizing (size of FIFO queues in the routers)
- Timing closure (Link bandwidth capacity allocation)
- Simulation (Network simulation, traffic/delay/power modeling)
- Testing

- … combined with problems of designing the NoC itself
  (topology synthesis, switching, virtual channels, arbitration, flow control,……)

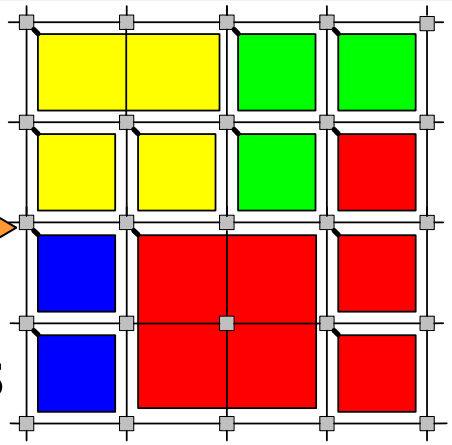
Let's see a NoC-based **design flow example**

QNoC-based SoC design flow

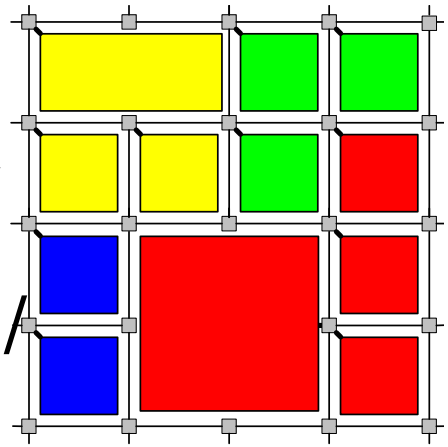


Inter-module Traffic model

Place Modules

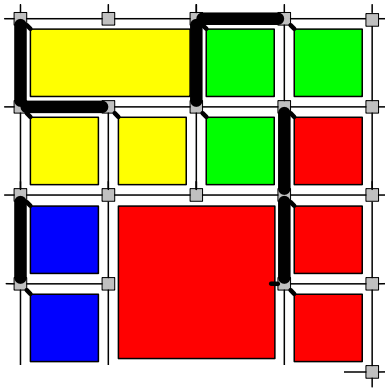


Trim routers / ports / links



Determine routing

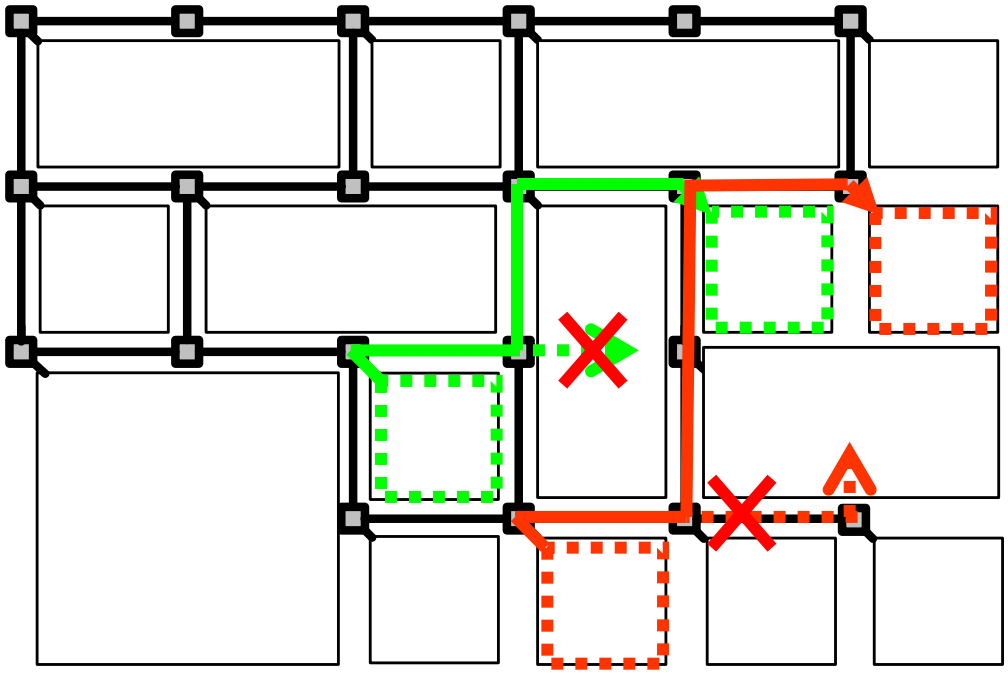
Adjust link capacities



Routing on Irregular Mesh

✓ Around the Block

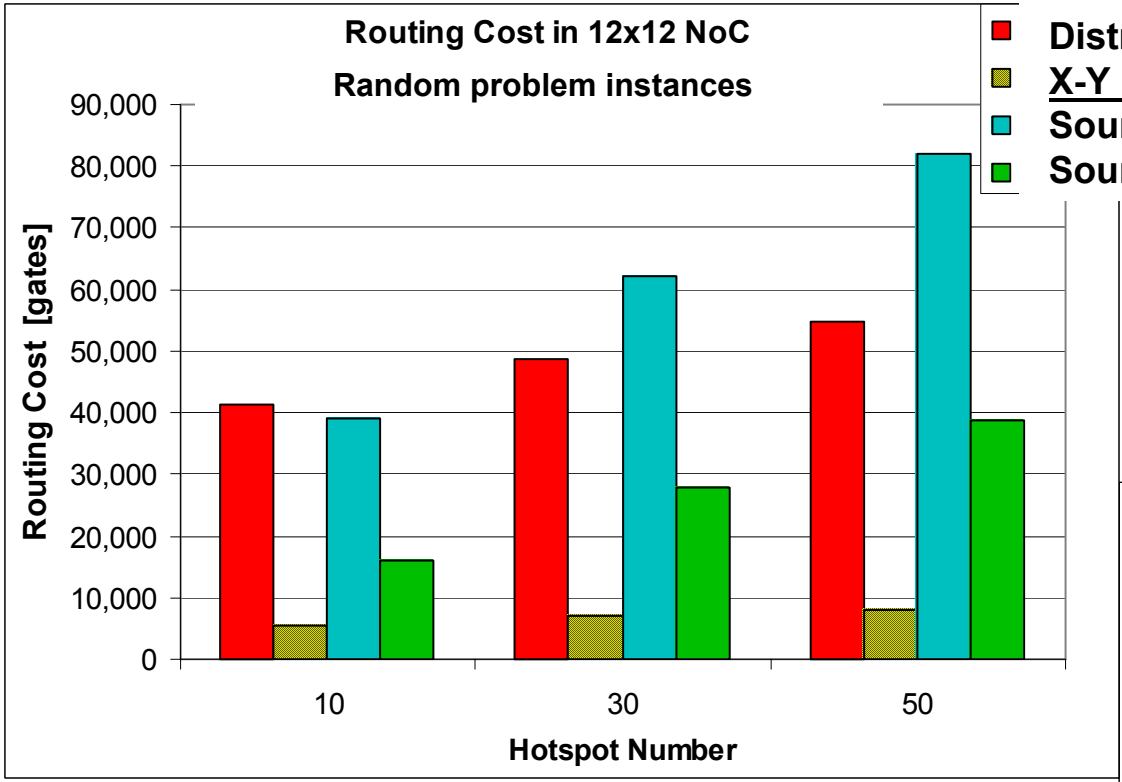
✓ Dead End

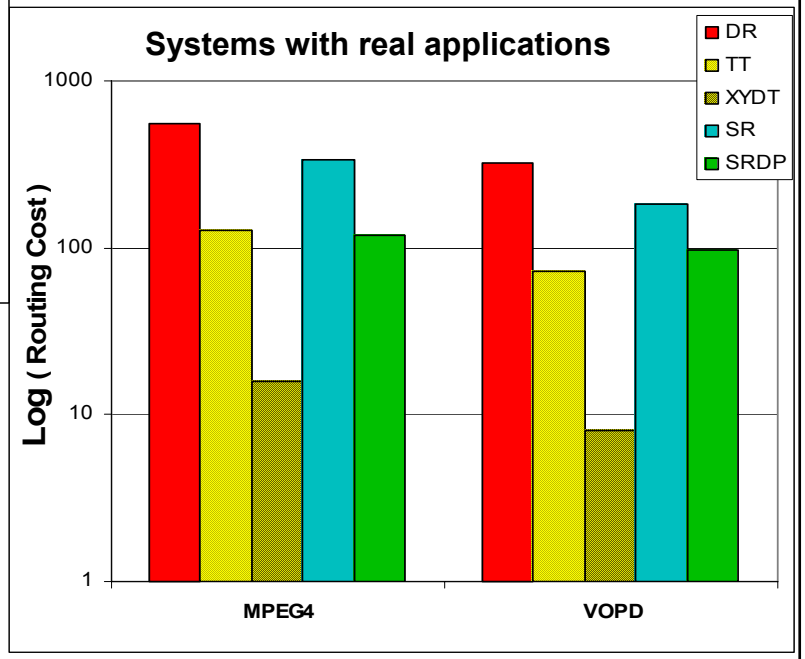


Goal: Minimize the total size of routing tables required in the switches

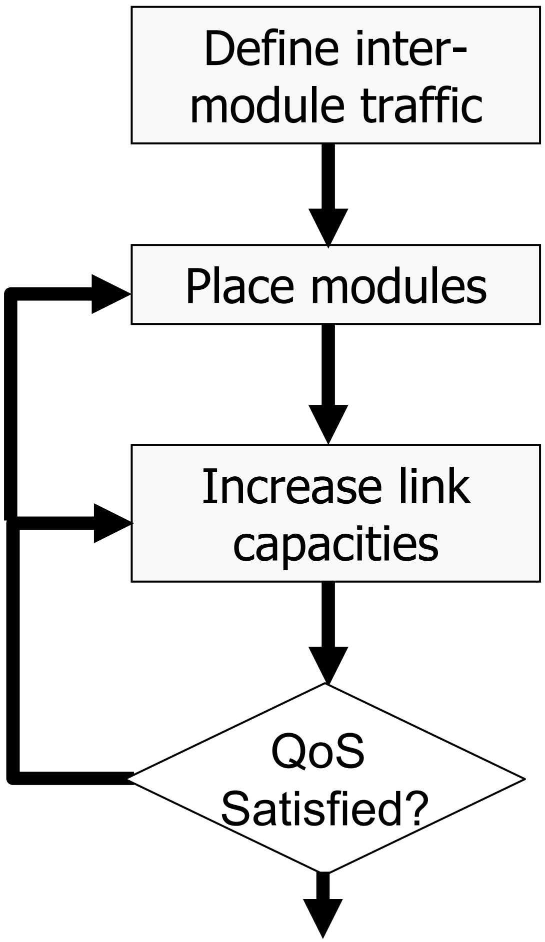
E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, ***"Routing Table Minimization for Irregular Mesh NoCs", DATE 2007***.

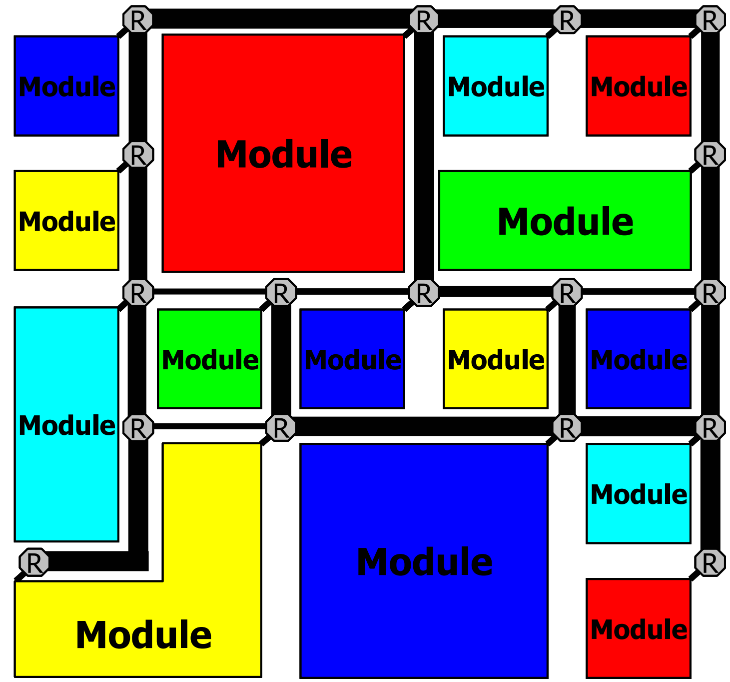
Routing Heuristics for Irregular Mesh





Timing closure in NoC





- **Too low capacity results in poor QoS**
- **Too high capacity wastes power/area**
- **Uniform link capacities are a waste in application-specific systems!**

Network Delay Modeling

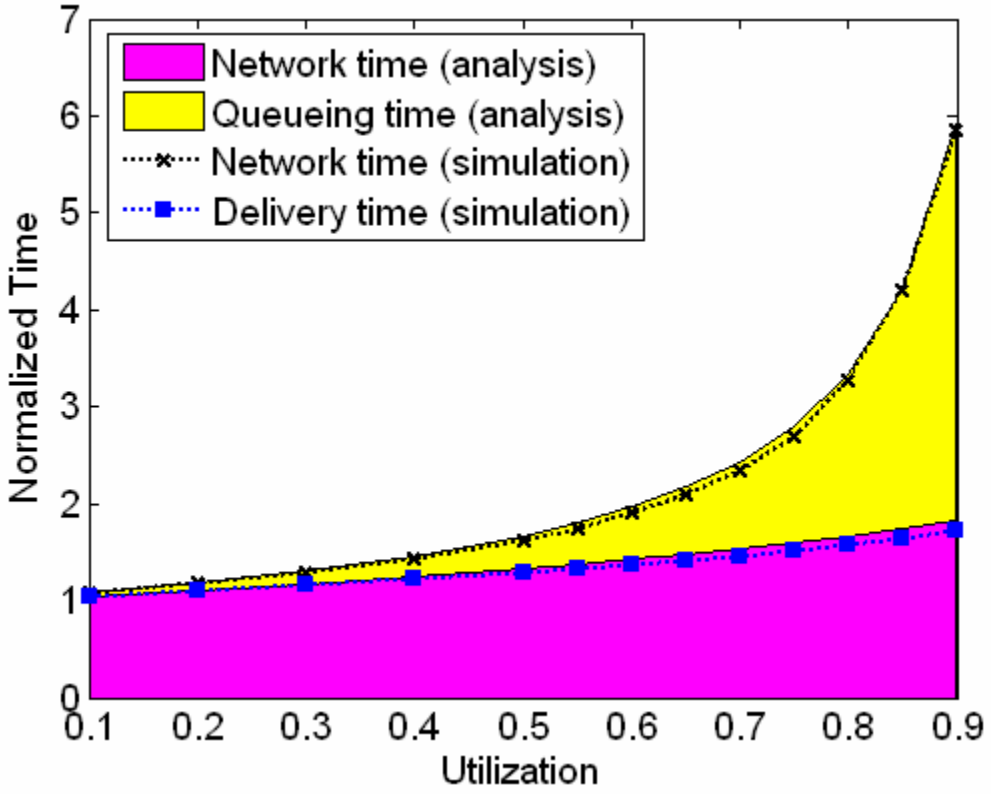
- **Analysis of mean packet delay in wormhole network**
  - **Multiple Virtual-Channels**
  - **Different link capacities**
  - **Different communication demands**

Queuing delay:

$$Q^i = \frac{1}{2 \cdot (\frac{1}{T^i_{network}} - \lambda^i)} - \frac{T^i_{network}}{2}$$

Flit interleaving delay approximation:

$$t^i_j = \frac{l}{C_j - l \cdot \sum_{f \mid j \in \pi^f \wedge f \neq i} \lambda^f \cdot m^f}$$



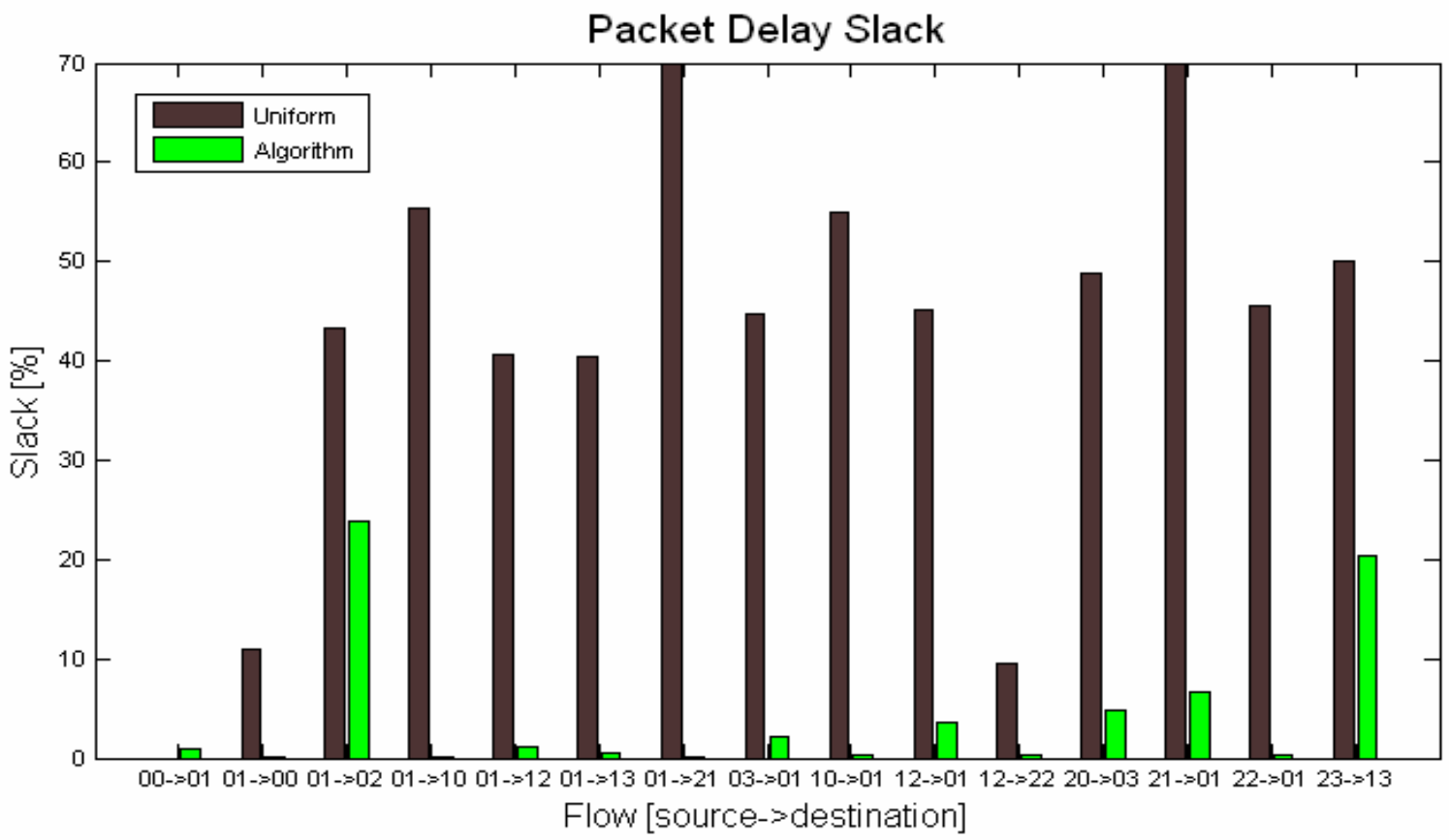
Capacity Allocation Problem

- **Given:**
  - **system topology and routing**
  - **Each flow's bandwidth (f^i) and delay bound (T^i_{REQ})**
- **Minimize total link capacity** $\left(\sum_{e \in E} C_e\right)$

- **Such that:**

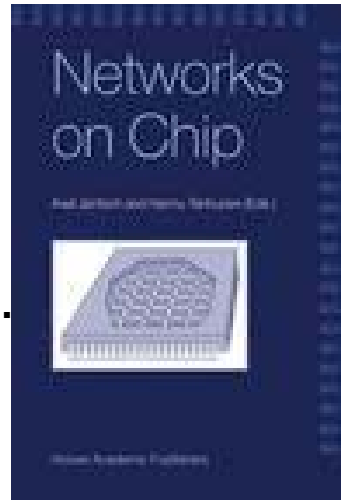
$$\forall link\ e: \quad \sum_{i \mid e \in path(i)} f^i < C_e$$

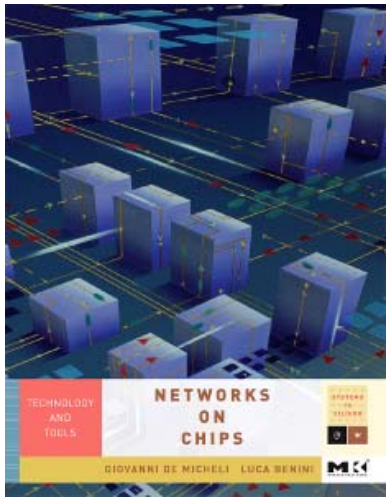
$$\forall flow\ i: \quad T^i \leq T^i_{REQ}$$



State of the art:
NoC is already here!

- **> 50 different NoC architecture proposals in the literature; 2 books; hundreds of papers since 2000**
- **Companies use (try) it**
  - **Freescale, Philips, ST, Infineon, IBM, Intel, …**
- **Companies sell it**
  - **Sonics (USA), Arteris (France), Silistix (UK), …**





- **1st IEEE Conference: NOCS 2007**
  - **102 papers submitted**

NoC research community

- Academe and industry
- VLSI / CAD people
- Computer system architects
- Interconnect experts
- Asynchronous circuit experts
- Networking/Telecomm experts

Possible impact:
Expect new forms of Rent's Rule?

- **View interconnection as *transmission of messages over virtual wires* (through the NoC)**

- **Model system interconnections among blocks in terms of required bandwidth and timing**
  - **Dependence on NoC topology**
  - **Dependence on the S/W application (in a CMP)**
  - **Usage for prediction of hop-lengths, router design, ….**

* *D. Greenfield, A. Banerjee, J. Lee and S. Moore,* **"Implications of Rent's Rule for NoC Design and Its Fault-Tolerance", NoCS 2007 (to appear)**

Summary

- NoC is a scalable platform for billion-transistor chips
- Several driving forces behind it
- Many open research questions
- May change the way we structure and model VLSI systems