QNoC: A Quality-of-Service Network-on-Chip Architecture

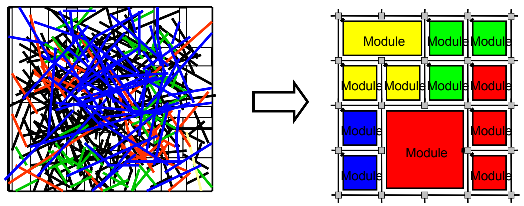
Avinoam Kolodny

VLSI Research Center
Department of Electrical Engineering
Technion—Israel Institute of Technology

Princeton University, 28 September 2005

Power and Area Efficient Network on Chip (NoC):

- Network layer architecture
  - ✓ Topology
  - ✓ Routing
  - ✓ Congestion control
  - ✓ Specialized features for CMPs
- Data link and Physical layers
  - ✓ Fast/power-efficient on-chip communication links
- Circuit design for NoC components



The Team

- **Faculty:**
  Israel Cidon, Ran Ginosar, Idit Keidar, Avinoam Kolodny

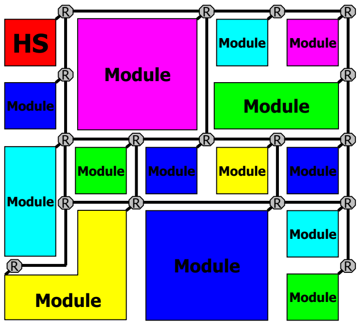
- **Graduate Students:**
  Evgeny Bolotin, Zvika Guz, Zigi Walter,
  Arkadiy Morgenshtein, Reuven Dobkin,
  Tomer Morad, Avshalom Elyada

Grants

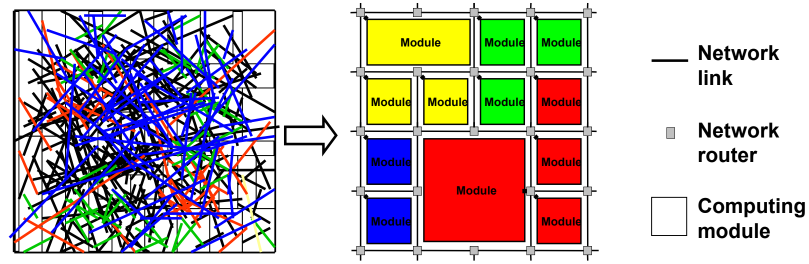
- Semiconductors Research Corporation
- ISRC consortium – Israel Government
- Intel Corp.

Outline

- Research motivation
- QNoC Architecture principles
- System Design flow with QNoC
- Specific topics:
  - Wormhole delay model
  - Hot Spots
  - Fast serial asynchronous links
  - Routing in an irregular mesh



A possible paradigm shift in VLSI

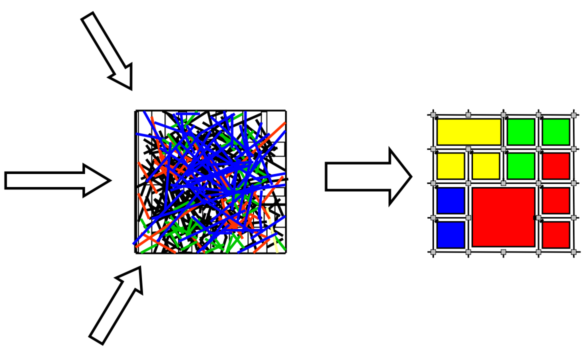


- Efficient sharing of wires by packet switching
- Lower cost / lower risk / faster design
- Scalable with system size
- NoC is an infrastructure (e.g. power, clock)
- NoC is customized for each chip

Why Now?

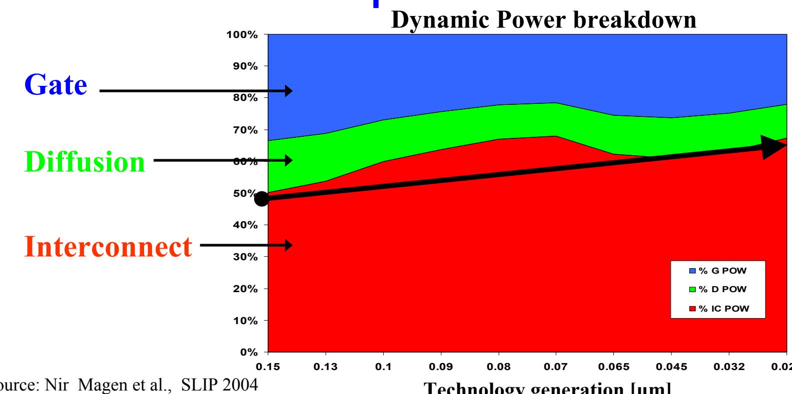
3) Chip Multi-Processors

2) Full-chip
productivity crisis



1) Sub-micron physical effects :
Global interconnect delay, power, noise

Interconnect power problem in a uni-processor



Source: Nir Magen et al., SLIP 2004
ITRS 2001 Edition adapted data

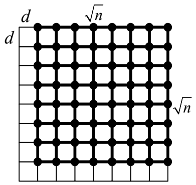
Interconnect power grows to 65%-80% within 5 years

(using optimistic interconnect scaling assumptions for a

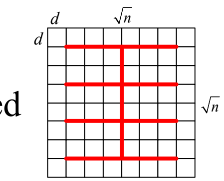
NoC scalability vs. alternatives

For Same Performance, compare the cost of:

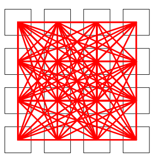
NoC:



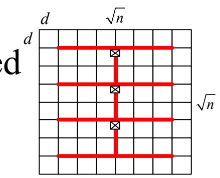
Non-Segmented Bus:



Point-to-Point:



Segmented Bus:



Asymptotic cost scalability

Power and Area required to provide same bandwidth versus number of system modules n

Arch	Total Area	Power Dissipation
NS-Bus	$O(n^3\sqrt{n})$	$O(n\sqrt{n})$
S-Bus	$O(n^2\sqrt{n})$	$O(n\sqrt{n})$
NoC	$O(n)$	$O(n)$
PTP	$O(n^2\sqrt{n})$	$O(n\sqrt{n})$

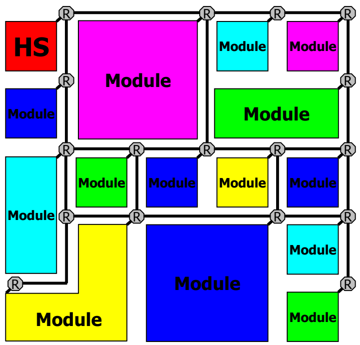
* E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "Cost Considerations in Network-on-Chip", INTEGRATION – the VLSI journal, 2004)

Practical NoC Challenges

- Low cost:
  - Area (routers, interfaces and links)
  - Power (dynamic, leakage)
- Flexible standard interface
- Multiple levels of service (QoS)
- Low design effort

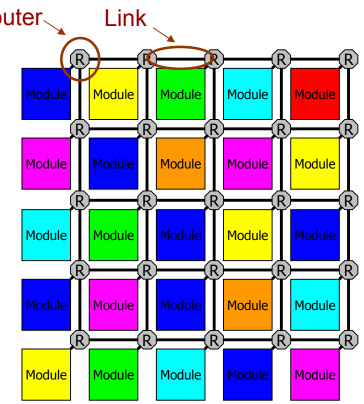
Outline

- Research motivation
- **QNoC Architecture principles**
- System Design flow with QNoC
- Specific topics:
  - Wormhole delay model
  - Hot Spots
  - Fast serial asynchronous links
  - Routing in an irregular mesh



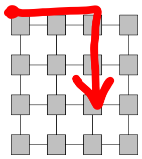
QNoC: Quality-of-service NoC architecture

- Grid topology
- Packet-switched
- XY Routing
- Service-levels
- Wormhole hop-to-hop flow-control



* 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.

QNoC topology and routing



- Grid topology matches planar technology
  - ✓ Variable capacity links!
  - ✓ Virtual channels
  - irregular mesh
- Fixed shortest path routing (X-Y)
  - ✓ Simple Router (no tables, simple logic)
  - ✓ No deadlock scenario
  - ✓ No retransmission
  - ✓ No reordering of messages
  - ✓ Power-efficient

QNoC Quality-of-service

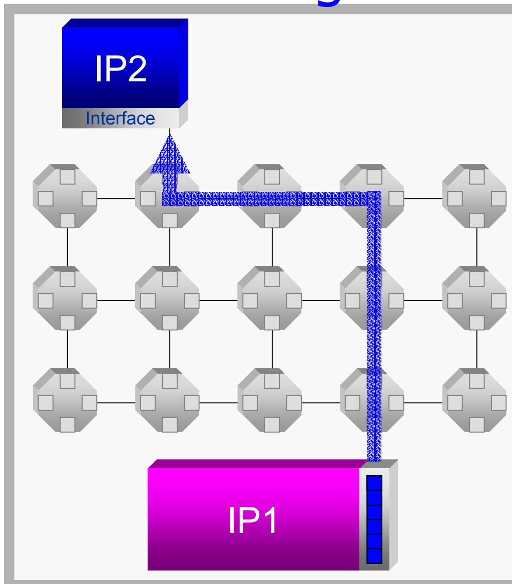
Define <u>Service Levels</u> like:

- *Signaling – interrupts, signals.*
- *Real-Time - audio, video.*
- *Read/Write (RD/WR) – bus semantics*
- *Block-Transfer – DMA semantics*

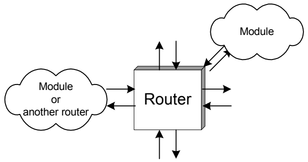
✓ Different QoS **(delay characteristics)** for each Service Level

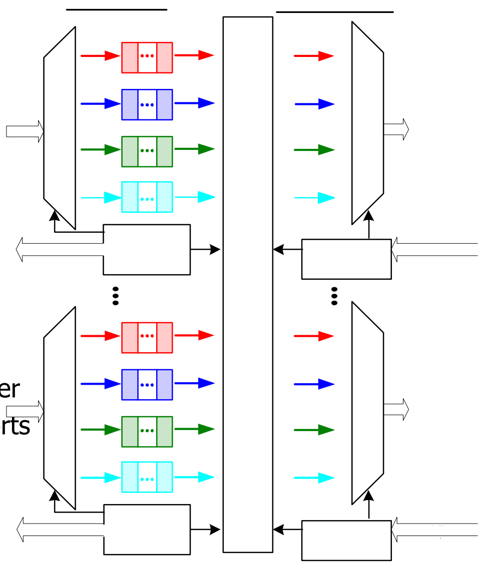
Wormhole Routing

- Small number of buffers
- Low latency
- Virtual Channels for concurrent flits transmission on the same link



Router structure

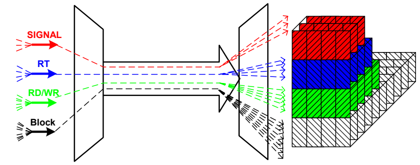




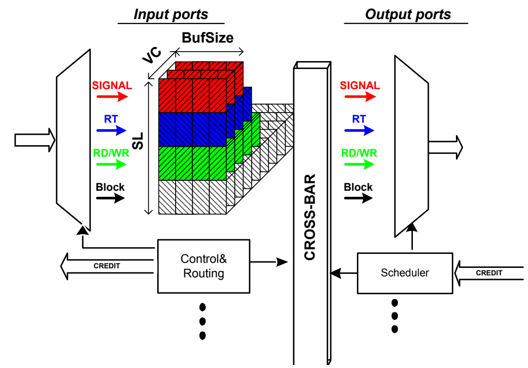
- Flits stored in input ports
- Output port schedules transmission of pending flits according to:
  - Priority (*Service Level*)
  - Buffer space in next router
  - Round-Robin on input ports of same SL
  - Preempt lower priority packets

QNoC router with multiple Virtual Channels

Multiple VCs link:



The QNoC Router:



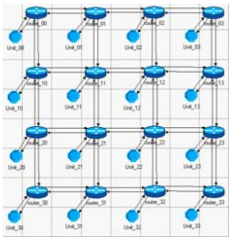
RT

Simulation Model

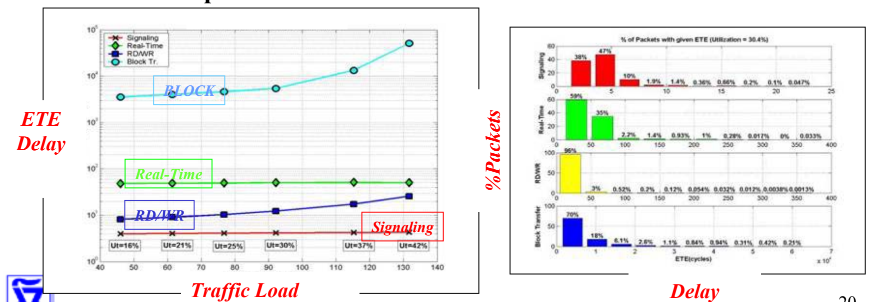
- OPNET Models for QNoC:
- Any topology and traffic load
- Statistical traffic generation at source nodes
- Flit level simulations

Simulation example

QNoC Example:

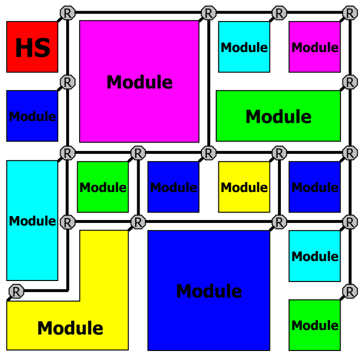


Results Example:

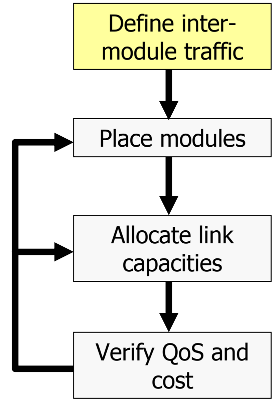


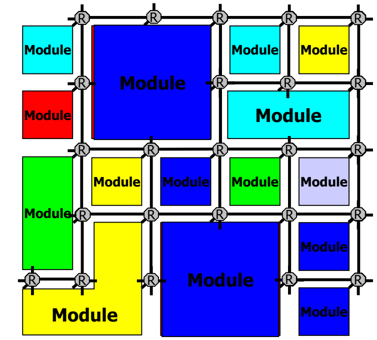
Outline

- Research motivation
- QNoC Architecture principles
- **System Design flow with QNoC**
- Specific topics:
  - Wormhole delay model
  - Hot Spots
  - Fast serial asynchronous links
  - Routing in an irregular mesh

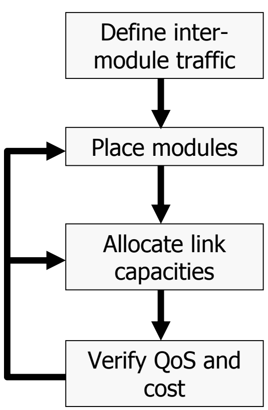


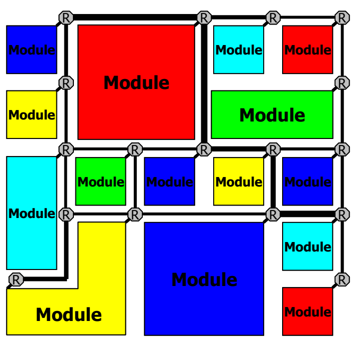
QNoC-based system Design Flow





QNoC Design Flow





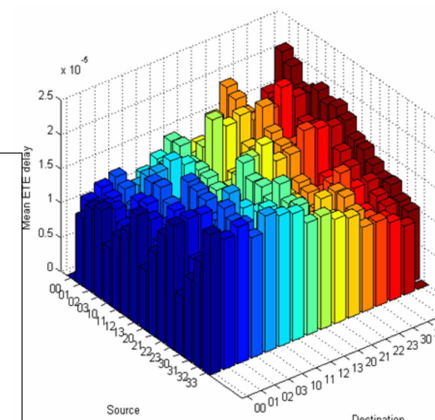
- Too low capacity results in poor QoS
- Too high capacity wastes power

Link 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:

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

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



Simulated mean packet delays in a 4-by-4 unoptimized network (uniform capacity in all links)

Capacity Allocation Algorithm

- Greedy, iterative algorithm

For each source destination pair:
- ✓Use delay model to identify most sensitive link
- ✓increase its capacity
- ✓Repeat until delay requirements are met

```
/*assign initial capacities*/
foreach link e:
  C_e <- SUM_{f in F : e in pi^f} lambda^f · m^f · l
end foreach

foreach flow f in F :
  /*evaluate current transit delay*/
  T^f <- Delay_Model (C, f)
  while (T^f > T_REQ^f )
    /*look for most sensitive link*/
    foreach e in pi^f :
      for all j != e : C~_j = C_j
      C~_e = C_e + delta
      T_e^f <- Delay_Model (C~, f)
    end foreach
    /*get most sensitive link*/
    e' = argmin_e (T_e^f)
    /*increase its capacity*/
    C_e' = C_e' + delta
  end while
end foreach
```

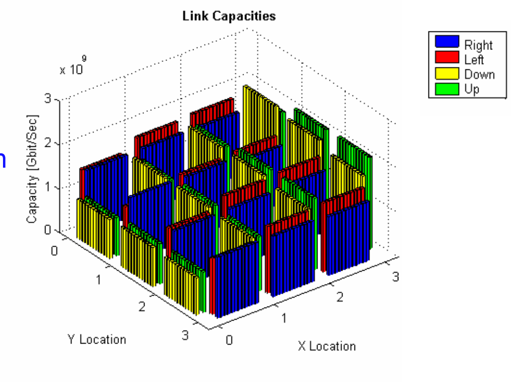
Figure 3: capacity allocation algorithm.

Capacity Allocation – Example#1

- A simple 4-by-4 system with uniform traffic pattern and uniform requirements
- "Classic" design: 74.4Gbit/sec
- Using the delay model and algorithm: 69Gbit/sec
- Total capacity reduced by 7%

Before optimization

After optimization



Capacity Allocation – Example#2

A SoC-like system with specific traffic demands and delay requirements

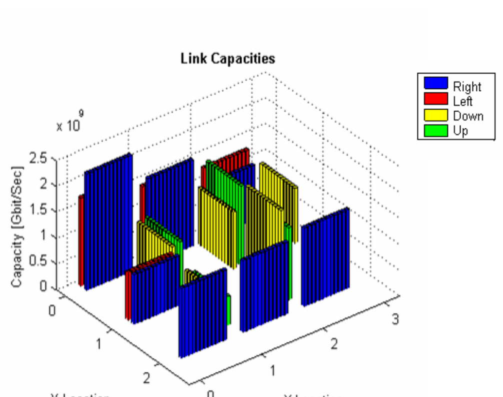
"Classic" design: 41.8Gbit/sec

Using the algorithm: 28.7Gbit/sec

Total capacity reduced by 30%

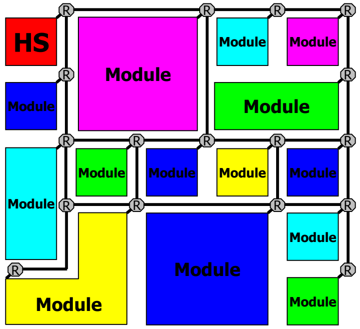
Before optimization

After optimization



Outline

- Research motivation
- QNoC Architecture principles
- System Design flow with QNoC
- **Specific topics:**
  - Wormhole delay model
  - Hot Spots
  - Fast serial asynchronous links
  - Routing in an irregular mesh



Need a static delay model

- An analytical delay model was developed for the link capacity allocation algorithm.
- Though many wormhole analysis models exists, they don't fit, because:
  - symmetrical communication demands are assumed
  - no virtual channels
  - identical link capacity is assumed in all links

Wormhole Delay Analysis

- Computed per flow
- Focus on long packets
- Packet transmission can be divided into two separate phases:
  - Path acquisition
  - Flits' transmission
- For simplicity, we assume "enough" virtual channels on every link
  - Path acquisition time is negligible

Flit Interleaving Delay

- Approximation for single link interleaving delay

$$t_j^i = \frac{1}{\frac{1}{l} \cdot C_j - \Lambda_j^i}$$

- t_j^i - *the mean time to deliver a flit of flow i over link j (waiting for transmission and transmission times)*
- C_j - *capacity of link j [bits per second]*
- Λ_j^i - the total flit injection rate of all flows sharing link *j* except flow *i* [flits/sec].

Flit Interleaving Delay

- Improved equation:

$$\tilde{t}_j^i = t_j^i + \sum_{k \mid k \in \pi_j^i} \frac{\Lambda_k^i \cdot l}{C_k} \cdot \frac{t_k^i}{dist^i(j,k)}$$

Account for all subsequent hops

Link Load

Basic delay weighted by distance

- The total delay over each flow path is:

flit size [bits]

Packet size [flits]

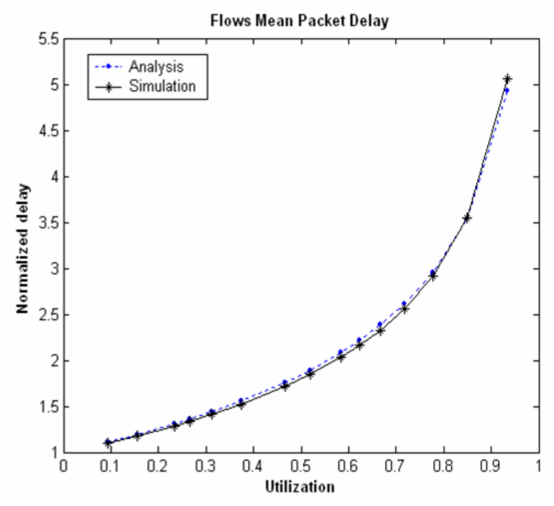
$$T^i \simeq (l \cdot m^i) \max(\tilde{t}_j^i \mid j \in \pi^i)$$

Account for weakest link

Wormhole Delay Analysis

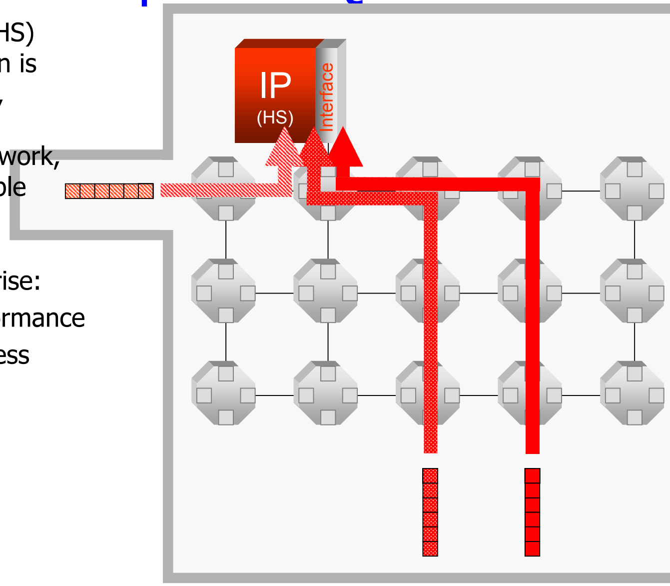
Analytical model was validated using simulations

- Different link capacities
- Different communication demands

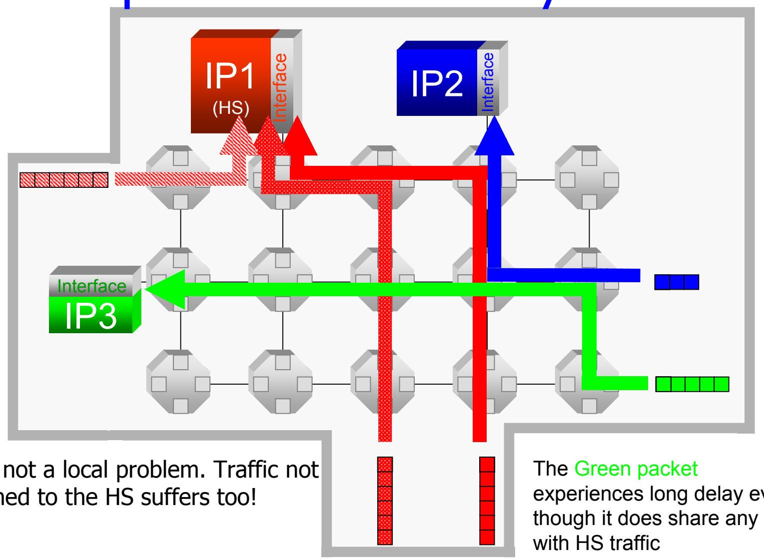


HotSpots in QNoC

- When HotSpot (HS) module utilization is temporarily high, worms "get stuck" in the network, occupying valuable resources
- Two problems arise:
  - System Performance
  - Source Fairness



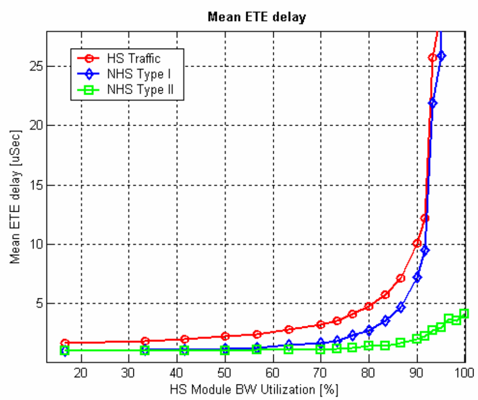
Hot Spot Affects the System



- HS is not a local problem. Traffic not destined to the HS suffers too!

The Green packet experiences long delay even though it does share any link with HS traffic

Network Performance problem

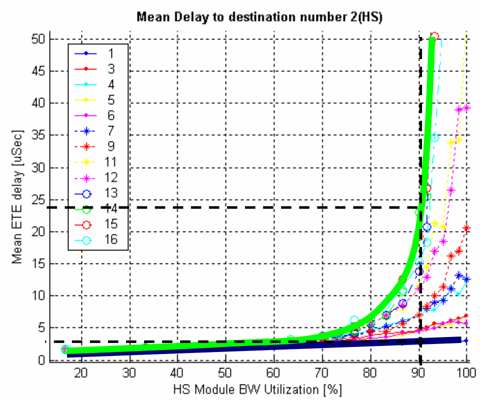


- As HS module utilization grows, a large part of the system becomes clogged

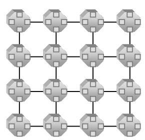
Source Fairness problem

Modules' location greatly affects the resulting QoS

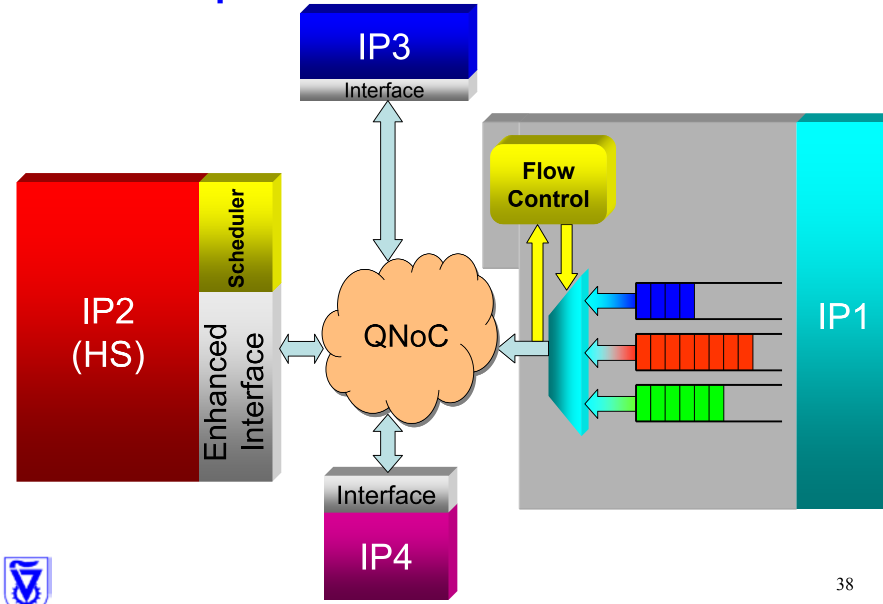
- e.g., At 90% utilization, a distant module experiences x10 the latency of a close one



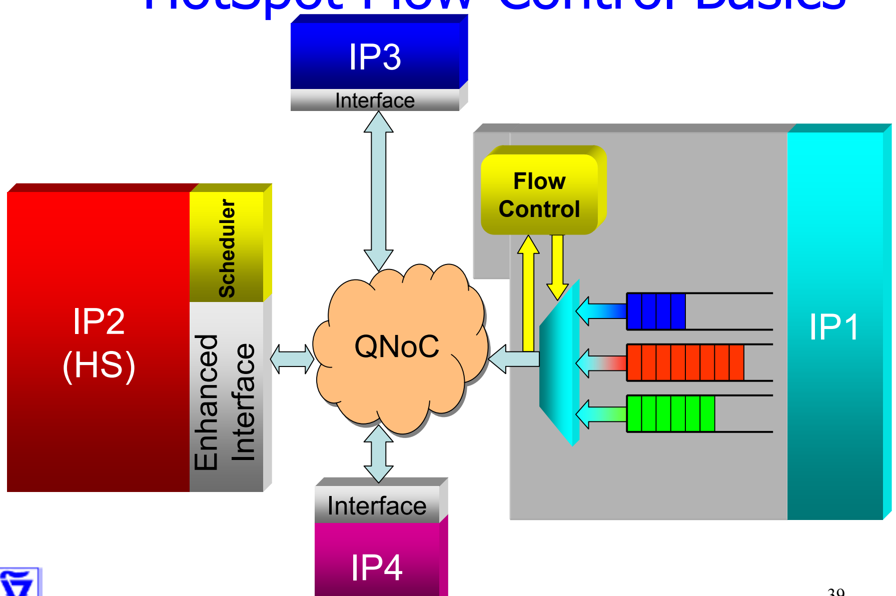
Simulation results for a 4x4 NoC with 10Gbit/Sec links, 6Gbit/Sec HS Module



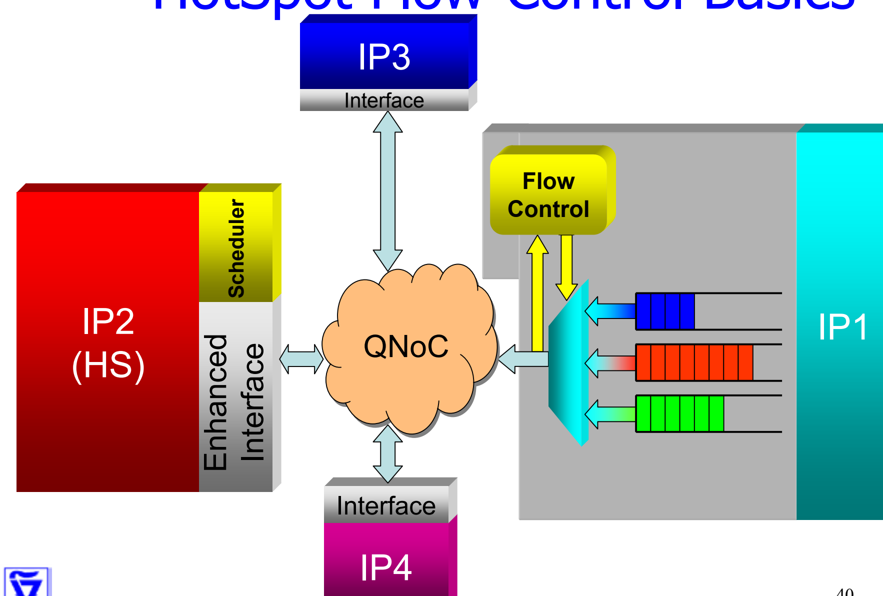
HotSpot Flow-Control Basics



HotSpot Flow-Control Basics

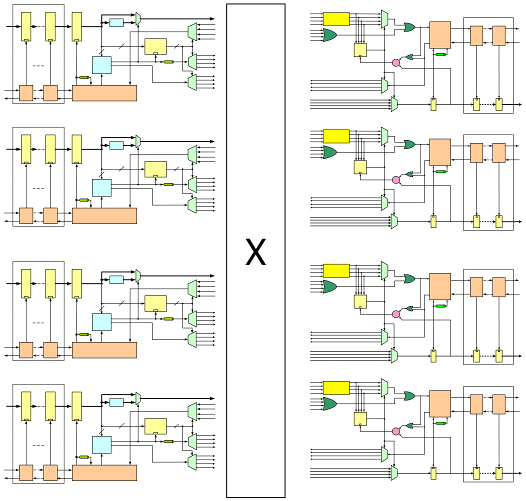


HotSpot Flow-Control Basics



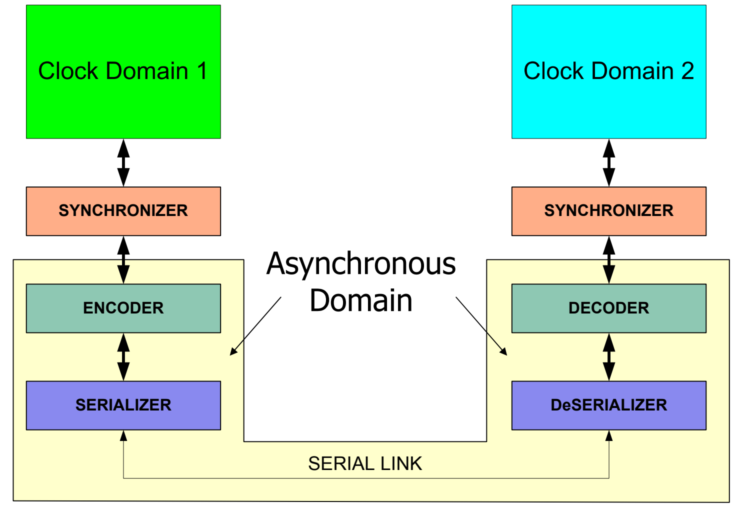
Asynchronous Router

Solves synchronization, clock domain crossings, timing, long connects



* **R. Dobkin, V. Vishnyakov, E. Friedman and R.Ginosar, "An Asynchronous Router for Multiple Service Levels Networks on Chip," ASYNC 2005.**

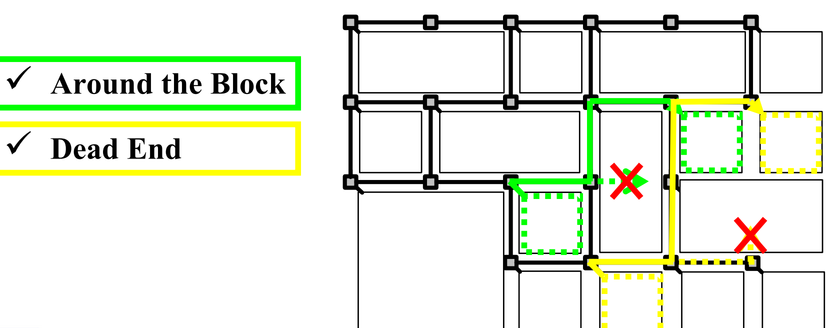
High speed asynchronous serial links



Hardware Efficient Routing in an irregular mesh

<u>The Problem</u>:

Simple Function (i.e. XY) cannot work in an irregular mesh



Traditional Routing Techniques

<u>Two main methods:</u>

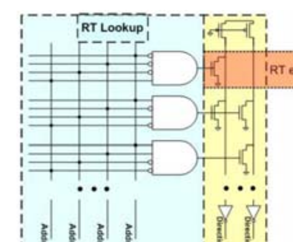
1. Distributed Routing:
- Full Tables in routers
- Each entry stores output port per each destination

2. Source Routing:
- Full Tables in sources
- Each entry stores list of routing tags (for each hop) per each destination

Use Reduced Table!:
- Stores only relevant destinations (PLA)

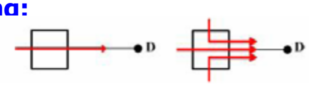
Area = (Size of Entries) + (Size of Lookup Logic)

Power = Const * Area

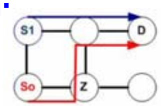


Efficient Routing: Solutions

- ***Distributed Routing (DR):*** **Function + Reduced Routing Tables**
  - **Turns Table (*TT*) routing:**



  - **XY Deviation Table (*XYDT*) routing:**

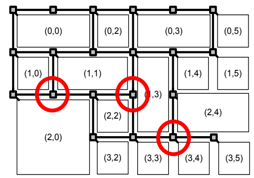


- ***Source Routing (SR):*** **Function + Reduced Source Routing Tags**
  - **Source Routing for Deviation Points (*SRDP*)**

Example:

Specific routers are *Deviation Points*

XY function for all other routers



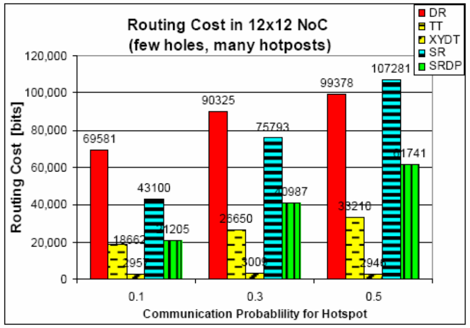
Results (random problem instances)

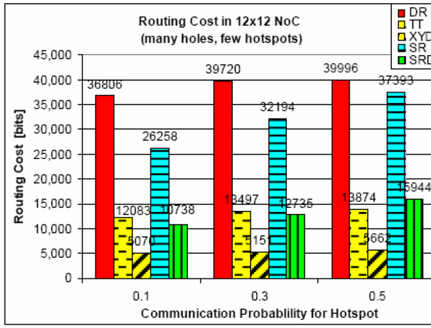
Few Holes: Low irregularity

34X savings by XYDT; **2X** by SRDP

Many Holes: High irregularity

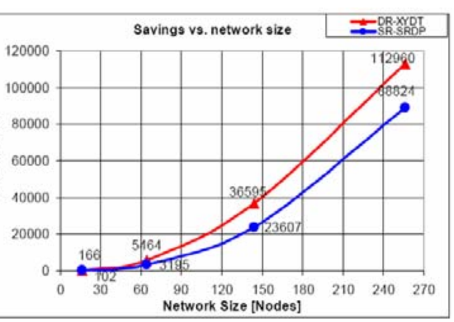
8X savings by XYDT; **2.5X** by SRDP



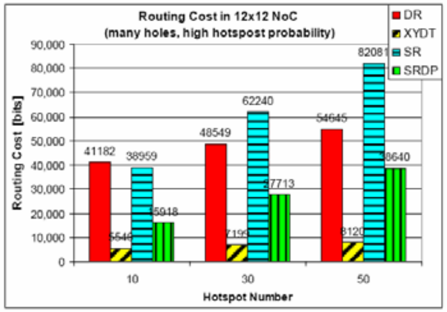


Scalability Results

Scaling of Savings:



Scaling of DR vs. SR



Summary

- Develop the QNoC design paradigm:
  - Architecture
  - Links
  - Circuits
  - Design flows & tools
- Start to investigate NoC-based multiple-core processors, as a proof-of-concept.