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

Research Topics
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

Interconnect power grows to 65%–80% within 5 years
(using optimistic interconnect scaling assumptions for a uniprocessor)
NoC scalability vs. alternatives

For Same Performance, compare the cost of:

- **NoC**:
  - [Diagram]
- **Non-Segmented Bus**:
  - [Diagram]
- **Segmented Bus**:
  - [Diagram]

Asymptotic cost scalability

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

<table>
<thead>
<tr>
<th>Arch</th>
<th>Total Area</th>
<th>Power Dissipation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS-Bus</td>
<td>$O(n^{\frac{1}{2}})$</td>
<td>$O(n \sqrt{n})$</td>
</tr>
<tr>
<td>S-Bus</td>
<td>$O(n^{\frac{1}{2}})$</td>
<td>$O(n \sqrt{n})$</td>
</tr>
<tr>
<td>NoC</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>PTP</td>
<td>$O(n^{\frac{1}{2}})$</td>
<td>$O(n \sqrt{n})$</td>
</tr>
</tbody>
</table>


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


---

**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 **Service Levels** 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:
  - Input ports
  - Output ports
  - Buffer spaces
  - SL
  - Control & Routing
  - Scheduler

**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**

1. Define inter-module traffic
2. Place modules
3. Allocate link capacities
4. Verify QoS and cost

**QNoC Design Flow**

1. Define inter-module traffic
2. Place modules
3. Allocate link capacities
4. Verify QoS and cost

- 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
  - Such that:

\[
\forall e \in \text{path}(i) \quad \sum_{e \in \text{path}(i)} f^i < C_e \\
\forall \text{flow } i \quad T^i \leq T^i_{REQ}
\]

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 */
1) foreach link s:
   2) C_s = \sum_{f \in \mathcal{F}} m_{sf} \cdot l_f
3) end foreach
4) foreach s \in \mathcal{S} :
   5) T^s \leftarrow \text{Delay Model} (C_s, f)
6) while (T^s > T_{\text{req}})
   7) foreach s \in \mathcal{S} :
      8) \forall f \neq s: C_f = C_s
      9) C_s = C_s + \delta
   10) T^s \leftarrow \text{Delay Model} (C_s, f)
   11) end foreach
   12) \epsilon = \text{argmin}(T^s)
   13) C_{\epsilon} = C_{\epsilon} + \delta
   14) end while
15) 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%

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%

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 exist, 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
  \[
  t_j^i = \frac{1}{l \cdot C_j - \Lambda_j^i}
  \]
  - \( t_j^i \) - 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]
  - \( \Lambda_j^i \) - the total flit injection rate of all flows sharing link \( j \) except flow \( i \) [flits/sec].

**Improved equation:**

\[
\tilde{t}_j^i = t_j^i + \sum_{k \in \pi^j} \frac{\Lambda_k^i \cdot l}{C_k} \cdot \frac{t_k^i}{\text{dist}'(j,k)}
\]

- Account for all subsequent hops
- Link Load
- Basic delay weighted by distance

- The total delay over each flow path is:
  \[
  T^i = (l \cdot m^i) \max(\tilde{t}_j^i \mid j \in \pi^i)
  \]
  - flit size [bits]
  - Packet size [flits]
Wormhole Delay Analysis

- Analytical model was validated using simulations
  - Different link capacities
  - Different communication demands

![Graph showing normalized delay vs. utilization](image)

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

![Diagram of HotSpot in QNoC](image)

Hot Spot Affects the System

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

![Diagram showing network with HotSpot affecting System](image)

Network Performance problem

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

![Graph showing Mean ETE delay](image)
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

- IP1
- IP2 (HS)
- IP3
- IP4

QNoC

Enhanced Interface

Scheduler

Flow Control

Interface
Asynchronous Router
Solves synchronization, clock domain crossings, timing, long connects


High speed asynchronous serial links

Hardware Efficient Routing in an irregular mesh

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

- Around the Block
- Dead End

Traditional Routing Techniques

Two main methods:
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**

<table>
<thead>
<tr>
<th>Scaling of Savings:</th>
<th>Scaling of DR vs. SR</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Net Size [Nodes]</th>
<th>Savings [k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>2000</td>
</tr>
<tr>
<td>60</td>
<td>4000</td>
</tr>
<tr>
<td>90</td>
<td>6000</td>
</tr>
<tr>
<td>120</td>
<td>8000</td>
</tr>
<tr>
<td>150</td>
<td>10000</td>
</tr>
<tr>
<td>180</td>
<td>12000</td>
</tr>
<tr>
<td>210</td>
<td>14000</td>
</tr>
<tr>
<td>240</td>
<td>16000</td>
</tr>
<tr>
<td>270</td>
<td>18000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hotspot Number</th>
<th>Routing Cost [k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2000</td>
</tr>
<tr>
<td>20</td>
<td>4000</td>
</tr>
<tr>
<td>30</td>
<td>6000</td>
</tr>
<tr>
<td>40</td>
<td>8000</td>
</tr>
<tr>
<td>50</td>
<td>10000</td>
</tr>
</tbody>
</table>

**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.