Sources

Main Reference Book (for the examination)
Designing Network-on-Chip Architecture in the Nanoscale Era, José Flich, Davide Bertozzi
Chapters 2 and 3

Additional References
- Timothy M. Pinkston, University of Southern California,
  http://ceng.usc.edu/smart/slides/appendixE.html
- On-Chip Networks, Natalie E. Jerger and Li-Shiuan Peh
- Principles and Practices of Interconnection Networks, William J. Dally and Brian Towles
- Chita Das webpage
On-chip Networks for shared memory multicores

- Nomenclature and Topology
- Cache implications
- Router microarchitecture
  - Baseline model
  - Optimizations
- Metrics
  - Power
  - Performance
What about an interconnection network?

An Interconnection Network is a programmable system that transports data between terminals.

- Technology: Interconnection network helps efficiently utilize scarce resources
- Application: Managing communication can be critical to performance
Why NoCs if so difficult to design?

- Increasing number of cores inside a single chip
- Reliability, flexibility, scalability, etc....
Memory Model in CMPs

• Message Passing
  • Explicit movement of data between nodes and address spaces
  • Programmers manage communication

• Shared Memory
  • Communication occurs implicitly through loads/stores and accessing instructions
  • Will focus on shared memory
  • Look at optimization for cache coherence protocols
Memory Model in CMPs

- Logically
  - All processors access some shared memory

- Practically...
  - cache hierarchies reduce access latency to improve performance

- Requires cache coherence protocol
  - to maintain coherent view in presence of multiple shared copies
  - **Consistency model**: the behaviour of the memory model in multi-core environment, i.e. what is allowed and what is not allowed
  - **Coherence**: shadow the cache hierarchy to the programmer (without lose performance improvement)
Tiled multi-core architecture with shared memory

Source: Natalie Jerger, ACACES Summer School, 2012
Coherence Protocol on Network Performance

- Coherence protocol shapes communication needed by system

- Single writer, multiple reader invariant
  - Requires:
    - Data requests
    - Data responses
    - Coherence permissions/forwards/Acks

- Suggested reading for a quick review of coherence:
Hardware cache coherence

- Rough goal:
  - All caches have same data at all times
  - Minimal flushing, maximum caches → best performance

- Two solutions:
  - **Broadcast-based protocol:**
    - All processors see all requests at the same time, same order.
    - Often relies on bus
    - But can broadcast on unordered interconnect
  - **Directory-based protocol:**
    - Order of the requests relies on a different mechanism than bus
    - Maybe better flexibility and scalability
    - Maybe higher latency
Scalable Cache Coherence

Source: Natalie Jerger, ACACES Summer School, 2012
Coherence Protocol Requirements

- Different message types
  - Unicast, multicast, broadcast

- Directory protocol
  - Majority of requests: Unicast
  - Lower bandwidth demands on network
  - More scalable due to point-to-point communication

- Broadcast protocol
  - Majority of requests: Broadcast
  - Higher bandwidth demands
  - Often rely on network ordering
Impact of Cache Hierarchy

• Sharing of injection/ejection port among cores and caches

• Caches reduce average memory latency
  • Private caches
    • Multiple L2 copies
    • Data can be replicated to be close to processor
  • Shared caches
    • Data can only exist in one L2 to bank
    • Addresses striped across banks (Lots of different ways to do this)
  • Aside: lots of research on cache block placement, replication and migration

• Serve as filter for interconnect traffic
On-chip Network: Private L2 Cache Hit

1. Miss A
2. L1 I/D Cache
3. Private L2 Cache

Source: Chita Das, ACACES Summer School, 2011
On-chip Network: Private L2 Cache Miss (off-chip)

1. LD A
2. Miss A
3. Private L2 Cache
   - Tags
   - Data
   - Controller
4. Format message to memory controller
5. Memory Controller
   - Request sent off-chip
6. Data received, sent to L2

Source: Chita Das, ACACES Summer School, 2011
On-chip Network: Shared L2 Local Cache Miss

1. LD A
2. Miss A
3. Format request message and sent to L2 Bank that A maps to
4. Send data to requestor
5. Receive message and sent to L2 Hit
6. Send data to requestor
7. Receive data, send to L1 and core

Source: Chita Das, ACACES Summer School, 2011
Network-on-Chip details
Two broad classes: Direct and Indirect Networks

- **Direct Networks**: Every node is both a terminal and a switch
  Examples: Mesh, Torus, k-ary-n-cubes…

- **Indirect Networks**: The network is basically composed of switches that connect the end nodes
  Examples: MIN, Crossbar, etc…

Source: Natalie Jerger, ACACES Summer School, 2012
Topology abstract metrics 1

- **Switch Degree**: Number of links/edges incident on a node
  - Proxy for estimating **cost**
  - Higher degree requires more links and port counts at each router

Source: Natalie Jerger, ACACES Summer School, 2012
- **Hop Count**: Number of hops a message takes from source to destination
  - Proxy for network *latency*
  - Every node, link incurs some propagation delay even when no contention
- **Network diameter**: large min hop count in network
  - Average minimum hop count: average across all source/destination pairs
- **Minimal hop count**: smallest hop count connecting two nodes
  - Implementation may incorporate non-minimal paths (increase avg hop count)

Source: Natalie Jerger, ACACES Summer School, 2012
Abstract metrics are just proxies: Does not always correlate with the real metric they represent

- Example:
  - Network A with 2 hops, 5 stage pipeline, 4 cycle link traversal vs.
  - Network B with 3 hops, 1 stage pipeline, 1 cycle link traversal
  - **Hop Count** says A is better than B
  - But A has 18 cycle latency vs. 6 cycle latency for B

- Topologies typically trade-off hop count and node degree
Traffic patterns: How to stress a NoC

• Synthetic traffic patterns
  – Uniform random, Matrix transpose, Hot Spot
  – Many others based on probabilistic distributions and pattern selection algorithms
  – **PROS:**
    • Fast analysis, corner case evaluation, future traffic pattern generation
  – **CONS:**
    • It could be not real

• Real traffic patterns
  • Real benchmarks executed on the simulated architecture
  • Complete evaluation of the system performance
  – **PROS:**
    • Data collected are from real scenarios for sure
  – **CONS:**
    • Time consuming simulations, no more scenarios than what provided by the exploited bench suite
Routing
- Defines the “allowed” path(s) for each packet (*Which paths?*)
- Problems
  - Livelock and Deadlock

Arbitration
- Determines use of paths supplied to packets (*When allocated?*)
- Problems
  - Starvation

Switching
- Establishes the connection of paths for packets (*How allocated?*)
- Switching techniques
  - Circuit switching, Packet switching
Until now old wine in a new bottle... but for caches

Routing algorithm
Deadlock
Router/switch
Flow control
Packets
Throughtput
Latency

Where is the difference?
Until now old wine in a new bottle...but for caches

On-chip network criticalalities

- Low power
- Limited resources
- High performance
- High reliability
- Thermal issues
Network-on-Chip: router architecture
NoC granulatity overview

- Messages: composed of one or more packets
  (NOTE: If message size is ≤ maximum packet size only one packet created)

- Packets: composed of one or more flits

- Flit: flow control digit

- Phit: physical digit
  (Subdivides flit into chunks = to link width)

**Off-chip:** channel width limited by pins

**On-chip:** abundant wiring means phit size == flit size
NoC microarchitecture based on granularity

- Message-based: allocation made at message granularity
  - circuit switching
- Packet-based: allocation made to whole packets
  - Store and forward (SaF)
    - Large latency and buffer required
  - Virtual Cut Through (VCT)
    - Improves SaF but still large buffers and latency
- Flit-based: allocation made on a flit-by-flit basis
  - **Wormhole**
    - Efficient buffer utilization, low latency
    - Suffers Head of Line (HoL)
  - **Virtual channels**
    - Primary to face deadlock
    - Then face HoL
Network-on-Chip: wormhole and wormhole+VCs
Switch/Router Wormhole Microarchitecture

- Flit-based, i.e. Packet divided in flit
- Pipelined in 4 stages
  - BW, RC, SA, ST, LT
- Buffers organized on a flit basis
- Single buffer per port
- Buffer states:
  - G – idle, routing, active waiting
  - R – output port (route)
  - C – credit count
  - P – pointers to data
Switch/Router Virtual Channel Microarchitecture
Router components

- Input buffers, route computation logic, virtual channel allocator, switch allocator, crossbar switch
- Most OCN routers are input buffered
- Use single-ported memories
- Buffer store flits for duration in router
- Contrast with processor pipeline that latches between stages

**Basic router pipeline** (Canonical 5-stage pipeline)

- BW: Buffer Write
- RC: Routing computation
- VA: Virtual Channel Allocation
- SA: Switch Allocation
- ST: Switch Traversal
- LT: Link Traversal
Router components

- Routing computation performed once per packet
- Virtual channel allocated once per packet
- Body and tail flits inherit this info from head flit
- Router performance
  - Baseline (no load) delay: 5 cycles + link delay \times \text{Hop} + t_{\text{serialization}}
  - How to reduce latency?
Pipeline optimization: Actual Baseline=BW+RC

- Usually RC happens at the BW stage
  - A single RC unit is required per input port
  - Is fast, since BW is not the critical stage
- Routing computation needed at next hop
  - Can be computed in parallel with VA
Pipeline optimization: Speculation

- Assume that Virtual Channel Allocation stage will be successful
  - Valid under low to moderate loads

- Entire VA and SA in parallel

- If VA unsuccessful (no virtual channel returned)
  - Must repeat VA/SA in next cycle

- Prioritize non-speculative requests
Pipeline optimization: Speculation + LRC

- LookAhead Route Computation (LRC)
  - the output port is computed in the previous node
  - 4 actions are performed in parallel
- The LRC does not actually compute the route but only reads the precomputed one in the head flit
Router Pipeline: module dependencies

- Dependence between output of one module and input of another
  - Determine critical path through router
  - Cannot bid for switch port until routing performed

Li-Shiuan Peh and William J. Dally. 2001. A Delay Model and Speculative Architecture for Pipelined Routers
Li-Shiuan Peh and William J. Dally. 2001. A Delay Model and Speculative Architecture for Pipelined Routers
Switch/Router Flow Control

- Flow control determines how a network resources, such as bandwidth, buffer capacity and control state are allocated to packets that are traversing the NoC
  - Resource allocation problem: from the resources point of view
  - Contention resolution: from the packet point of view
  - Bufferless, buffered
Switch/Router Buffered Flow Control

- Buffers
- More flexibility, with the possibility to decouple resource allocation in steps
- Two modes
  - Wormhole flow control
  - Virtual channel flow control

Switch/Router Buffered Wormhole Flow Control

- Allocate on a per flit basis
- More efficient in buffer consumption
- Head of Line (HOL) blocking issues
- Buffered solutions allow to decouple resource allocation

U – upper outport, L – lower outport
In port States (I,W,A) (idle, waiting, allocated)
Flits (H,B,T) (head, body, tail)

Switch/Router Virtual Channel Flow Control

- Multiple buffers on the same input port
- Need for a state on each virtual channel
- More complex to manage than wormhole
- Allows to manage different flows at the same time
- Solves the HoL issues
- Deadlock avoidance property

Why Virtual Channels (VCs): Head of Line Block

Buffer Management and Backpressure

- How to manage buffers between neighbors (i.e. how can I know the downstream destination router buffer is full?)
- Three ways:
  - **Credit based**
    - The upstream router keeps track of the available flit slots available in the downstream router
    - Upstream router decreases counter when sends a flit while downstream router increases the counter (backward) when a flit leave the router
    - Accurate fine grain control on flow control, but a lot of messages
  - **On/off**
    - Threshold mechanism with single bit low overhead to signal upstream router the permission to send
  - **Ack/nack**
    - No state in the upstream node
      - Sends and wait for ack/nack, no net gain
      - Waist of bandwitdh, sending without ack guarantee
Credit-based flow control

On-off flow control

Ack-nack flow control

NoCs: some implementation details

From: “Designing Network-on-Chip Architectures in the Nanoscale Era” José Flich Davide Bertozzi, 2011
Buffer structure and Crossbar
Input-first Switch Allocator

Input 1

Enable Granted VC

Per output arbiters

Requests from input 2

Requests from input N

Gather requests for the same output

Gather grants for the same input

N:1 Arbiter

Output 1

N:1 Arbiter

Output 2

\[ \text{At least one grant} \]

\[ \text{Grants for input 1} \]

\[ \text{Grants for input 2} \]

\[ \text{Grants for input N} \]
Input-first VC allocator

- At least one grant
- Selected Output Port
- V:1 Arbiter
- Candidate Output VCs
- Align request to output
- Select only one output VC

Per output VC arbiters:
- NV:1 Arbiter
- Grant for input 1 – VC0
- Grants to rest output VCs

Requests from rest input VCs
Gather requests for the same output VC
Gather grants for the same input VC
EXTRA
Evaluation metrics for NOCs

**Performance**
- **Network centric**
  - Latency
  - Throughput
- **Application Centric**
  - System throughput (Weighted Speedup)
  - Application throughput (IPC)

**Power/Energy**
- Watts/Joules
- Energy Delay Product (EDP)

**Fault-Tolerance**
- Process variation/Reliability

**Thermal**
- Temperature
Buffer power, crossbar power and link power are comparable.

Arbiter power is negligible.

Source: Chita Das, ACACES summer school 2011
Switch/Router Bufferless Flow Control

- No buffers
- Allocate channels and bandwidth to competing packets
- Two modes
  - Dropping flow control
  - Circuit switching flow control

Bufferless Dropping Flow Control 1

- Simplest flow control form
- Allocate channel and bandwidth to competing packets
- In case of collisions we experience packet drops
- Collision can be signaled or not using ack-nack messages

• With no ack messages the only viable way is timeout timers

• Ack messages can reduce latency

Bufferless Circuit switching Flow Control 1

- It allocates all needed resources before send the message
- When no further packets must be sent, the circuit is deallocated
- Head flit arbitrates for resources, and if stalled no resend needed