Packet Switch Architecture

Nikolaos Chrysos, Manolis Katevenis

FORTH and Univ. of Crete, Greece

`http://archvlsi.ics.forth.gr/~kateveni/534`

8. Networks on Chip (NoCs)
Table of Contents:

- **8.0 Introduction**
  - chip multiprocessors / systems on chip

- **8.1 Direct topologies**
  - *k*-ary *n*-cubes

- **8.2 Flow control**
  - bufferless, store & forward, virtual cut-through, wormhole, virtual channel

- **8.3 Routers overview**

- **8.4 Deadlocks & routing overview**

8.0 NoCs for CMPs & SoCs

- **Chip multi-processors (CMPs)**
  - modern processors (~ 2007-) scale up by increasing the number of cores per chip, not the frequency

- **Systems on chip (SoCs)**
  - integrate a full system on chip to economize on number of discrete components
  - CPUs, GPUs, accel., netw. interfaces (NICs), caches, DRAM ctrl…
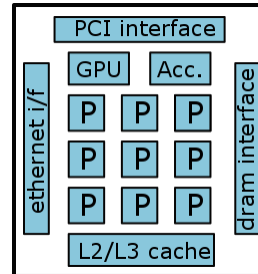
- **As chips integrate more elements, they use a NoC for their interconnect**
  - nodes communicate using ("universal") network packets (header + payload)
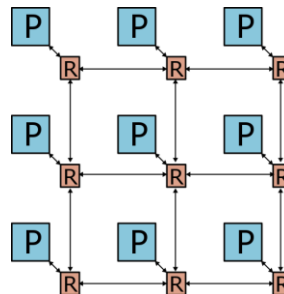
8.0 Networks on Chip

9-node mesh NoC

- Typically direct networks
  - each node both a terminal and router
  - versus indirect networks, like banyan, Benes, etc, commonly used in fabrics
- Point-to-point links between routers

- NoCs similar to off-chip networks (& fabrics) but many trade-offs differ
  - signals don't cross chips boundaries: power consumption mainly due to <u>buffers</u>, wires & logic, not due to transmissions
  - buffers are expensive (compared to other components) and therefore small
  - channels (parallel wires instead of optical/fiber links) are cheap(er) and can be made fast (=wide)
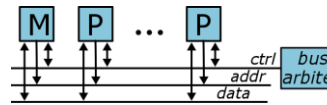
8.0 Traditional on-chip interconnects
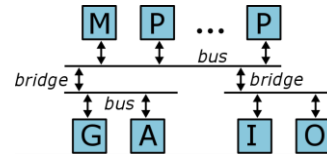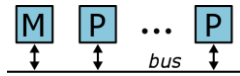
- Separate wires/buses, sometimes arbitrarily connected, like in hardware design
  - per purpose / protocol / speed

- Buses can offer high speed (if short), but turn-around ovrhd is a concern
  - separate wires for sending address/ctrl →
  - accesses serialized by bus arbiter →
  - limit on number of devices

- Multiple (hierarchical) buses
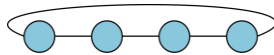 - for scalability and to bridge different bus protocols or speeds

- Bus-like on-chip interconnects are used today (2015)
  - ring networks in CMPs with up to 8-16 (32?) cores
- Scalable prototypes & next-generation CMPs (will) rely on NoCs

8.1 Torus : *k*-ary *n*-cubes

4-ary 1-cube (ring)

4-ary 2-cube


- N= k^n nodes in a n-dimensional cube
  - 2n + 1 ports per router
- k = number of nodes along each dimension
- Each node has a n-digit radix-k address
- Two nodes are (<u>adjacent</u>) connected w. a bidir channel iff they differ by (1 mod k) in one or more digits

Recursive definition

- How to create a k-ary n-cube:
  - stack k k-ary (n-1)-cubes
  - add 1 new MS addr. digit to each node
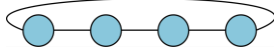   - different for every (n-1)-cube
  - connect adjacent nodes
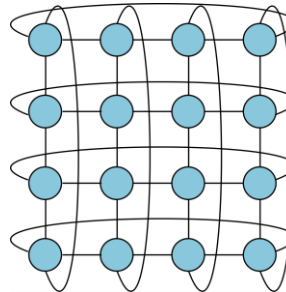
*Current NOCs limited to 2D but might change with 3D integration*

8.1 Torus : *k*-ary *n*-cubes

4-ary 1-cube (ring)

4-ary 2-cube

- Bisection wire count
    - $2k^{n-1}$ (= 2N/k) same-direction wires available to $k^n/2$ (N/2) nodes
- Torus are blocking
    - worst-case bisection channel load k/4
    - for uniform traffic: k/8

- Higher dimensional tori
    - more bisection b/w (less blocking) for fixed number of nodes
    - and smaller latency
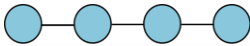   - at the cost of 2 extra router ports / dimension
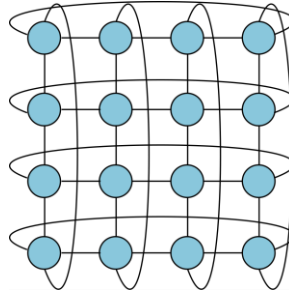
- For fixed N, going from n to n'=n+1
    - k' = (k)^(n/n+1) < k
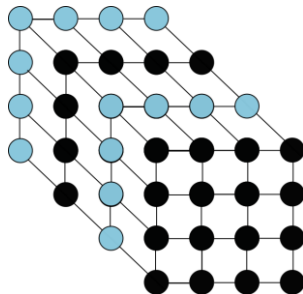   - for small n, k' << k

8.1 Mesh: *k*-ary *n*-cubes
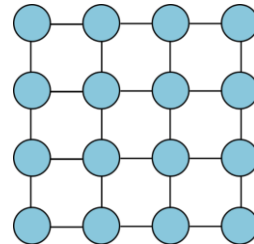
4-ary 1-mesh

4-ary 3-mesh

Routers in the middle have 2 n + 1 ports

Routers at the border of i dimensions have i less ports – excess ports can be used for I/O or express channels

4-ary 2-mesh

- Meshes may use node-concentration
    - multiple nodes / router
- or express channels or multiple nets or …)

- Tori networks w/o wrap- around channels
- Bisection wire count
    - k^{n-1} half than tori netw.

- For up to 64 nodes, simple meshes offer better throughput per power performance
    - Psathakis, e.a., "A Systematic Evaluation of Emerging Mesh-like CMP NoCs", ANCS 2015

8.1: Cubes vs. Crossbar & Clos for NoCs

- Why do most NoCs (and several multiprocessors/ supercomputers) use Tori and meshes instead of indirect (banyan, Clos, Benes) networks ?
  - shorter links → smaller latency & power consumption, especially for NoCs
    - chip end-to-end communication latency (measured in clock cycles) is substantial and becomes worse with new technologies
- Cubes favor neighbor communic.: common in multiproc. & scientific computing
- Cubes have worse bisection bandwidth, and don't offer equidistant paths
  - takes longer to travel "further away" (and the probability of congestion increases)
  - innermost links tend to experience higher load under e.g. all-to-all traffic

- Crossbars & Clos have been used to build efficient on-chip netw. or switches
  - Passas, e.a., "The Combined Input-Output Queued (CIOQ) Crossbar Architecture for High-Radix On-Chip Switches", IEEE Micro, 2015.
  - Chrysos, e.a., "High-Radix Switches Made of Bufferless Clos Networks", IEEE HPCA, 2015.
- Mesh preferred over Tori, because of no wrap-around links – simpler to provide deadlock-free operation using e.g. X-Y routing
  - meshes also have same-length links, but so do Tori if we redraw them appropriately

8.2 The unit(s) of communication in NoCs

- **Messages:**
  - generated by the user / app (e.g. cache line)
- **Packets:**
  - msg segments that make sense to the network
  - each pkt has a hdr: dst, src, seq, etc.
  - all bits (flits) of a pkt follow the same path
- **Flit (flow control units):**
  - flits = packet segments
  - buffers reserved at the granularity of flits
  - head flit (carries the dest addr) and establishes the path of the pkt
  - body and tail flits follow the path of the head flit
    - zero or many body flits -- head may be = tail
- **Phits (datapath width):**
  - all bits (e.g. 32, 128) of a phit transmitted in parallel
  - commonly 1 flit = 1 phit
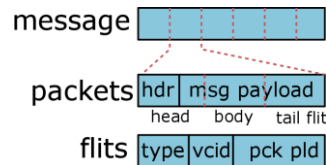
Similar terminology in off-chip multiprocessor networks

8.3 Flow control

Flow control determines when buffers and links are assigned to msgs/packets

- a good flow control reduces packet latency & increases utilization of resources
 - in a cost-effective manner...

- **Bufferless flow control: no buffers inside the network**
  - message-based circuit switching:
    - setup flit travels to dest and reserves links on the way
    - ack flit travels back to source
    - payload flits go through the network with no delay (no buffers needed)
    - termination flit de-allocates each reserved hop
  - packet based deflection routing (misrouting): no setup ovrhd, each pkt tries to reach destination; if conflict on next hop, misroute packets = send them to idle ports (# inputs / router = # outputs / router); in a mesh network, they will be able to reach their dest, but with what latency (priority to older pkts)

- **Buffered flow control** (mostly pkt-based - msgs too long for on-chip buffers)
  - store-and-forward : switch buffers reserved for entire packet
    - store full packet (all flits) before transmitting it downstream
  - virtual cut-through : switch buffers reserved for entire packet
    - can transmit head before receiving tail

8.3 Packet-based (buffered) flow control

*4-flit packet from node 2 to node 0*
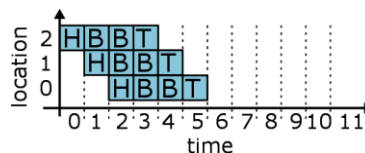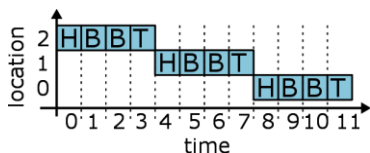
- Packet-based flow control
  - latency: # hops **x** pkt xmit time

- Virtual cut-through
  - latency : # hops **+** pkt xmit time

- Disadvantages:
  - router buffer size proportional to packet size
    - maybe too large for on-chip networks
  - if a packet is blocked (cannot move downstream), buffers cannot be reused by other pkts

8.3 Wormhole (flit-based) flow control

4-flit packet from node 2 to node 0

Other pkts cannot use channel 2→1
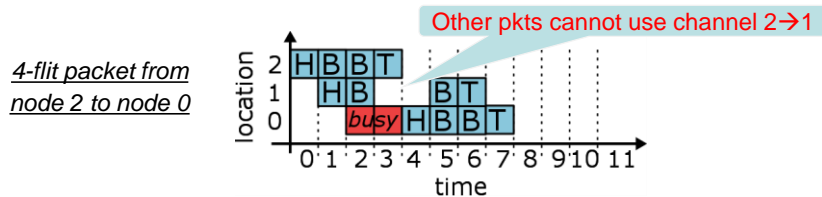
- Like in virtual cut-through, flits are forwarded before receiving full packet
  - latency = first-bits in to last-bit out of network = ~ # hops + pkt xmit time
- However, *buffers reserved at the granularity of flits*
  - buffers at routers can be smaller than packets
- Head flit governs route = the next output channel at each router
  - remaining flits follow in a pipeline fashion
- Cannot have flits from different packets in the same queue
  - in case of contention, flits may wait in upstream routers
    - congestion/blocking spreading
- Also, output channel reserved until all flits of a packet have crossed it, and have departed from downstream queue → needless (~HOL) blocking

13

8.3 Virtual channel flow control

Virtual channels
- Multiple buffers/queues (virtual channels) at each sw. input
  - used for HOL de-blocking, ~ VOQs but not necessarily one VC per output; instead one VC per pkt
  - used also for deadlock avoidance
- Virtual channels can be applied to packet- or flit-based flow control

Virtual channel flit-based flow control (wormhole-like)
- At any given router, all flits of the same pkt are stored in the <u>same VC buffer</u>
- Like wormhole: head flit governs output port -- now also the next-hop VC
  - next flits follow; each flit has to secure downstream credits for the next-hop VC
- Flits that have acquired next-hop VCs can be interleaved on outputs

Drawbacks:
- Each VC (input queue) held for the duration of a pkt – compared e.g. to ATLAS, where a VC can store multiple packets
- Also multiple VCs at one input may be allocated to pkts going to the same destination → pkts to non-congested outputs may not find available VC
  - see also chapter 6

14

8.3 Overview of NoC virtual channel router

- Commonly IQ input-queued xbar
  - few ports / router → xbar OK
- Credits counters maintained by VC allocator
  - credits sent/received from separate wires (out-of-band)
- 5-stage processing pipeline
  - FW: write flit at input buffer
  - R: find output (only head flit)
  - VA: allocate next-hop virtual channel (only head flit)
    - a downstream VC available only iff corresponding VC cred-count full?
  - SA : crossbar scheduler
  - ST: switch traversal
- Latency can be reduced using e.g. speculation
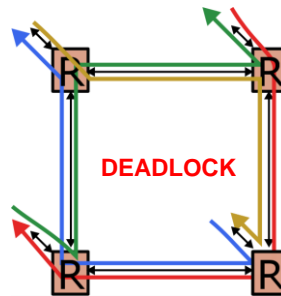 - Peh and Dally "A Delay Model and Speculative Architecture for Pipelined Routers", HPCA 2011
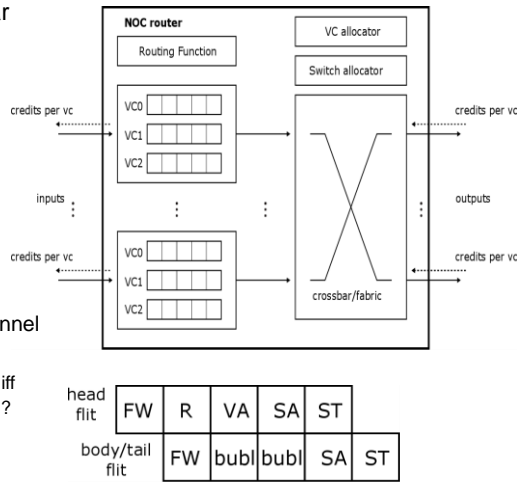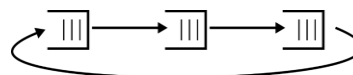
8.4 Routing Deadlocks

Deadlocks: routing or protocol induced

- no progress → disaster!

  (livelock: pkts move, but no "real progress" is made, e.g. bouncing back and forth)

DEADLOCK

Routing deadlock (w. wormhole)

- Figure (wormhole): four pkts, none can move
  - green occupies left link, needs top
  - yellow occupies top link, needs right
  - red occupies right link, needs bottom
  - blue occupies bottom link, needs left
  - → *circular dependency*

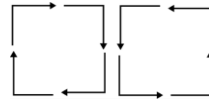**In routing deadlock, buffers usually fill up**

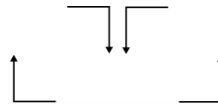*no packet can proceed, because downstream buffer occupied by other packet(s)*
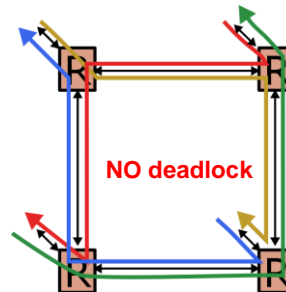
8.4 Routing Deadlocks

- All possible turns in mesh networks →
  - they permit circular link dependencies

- Dimension ordered routing (DOR) →
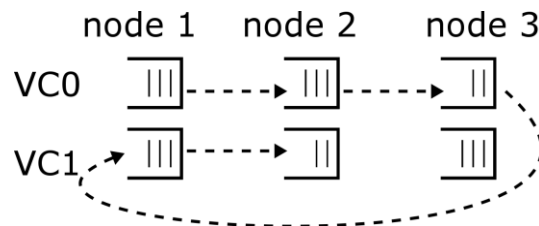 - disallow turns to prevent dependencies

- Example X-Y (DOR) routing →

NO deadlock

- Prevents circles but single-path / deterministic routing
  - even worse, some paths can never be used...

8.4 Routing Deadlocks

node 1   node 2   node 3

VC0

VC1

Use virtual channels to enable arbitrary, e.g. adaptive, routing

(NoC-style adaptive routing: the head flit dynamically selects path - other flits follow)

- Change (increment) VC at every hop – too many VCs if too many hops
- Change the VC when crossing the "timeline " (see example)
  - Dally, "Deadlock-free adaptive routing in multicomputer networks using virtual channels", IEEE TPDS, 1993
- **Theory**: Duato, "A new theory of deadlock-free adaptive routing in wormhole networks", IEEE TPDS 1993