Networks on Chip

Kees Goossens

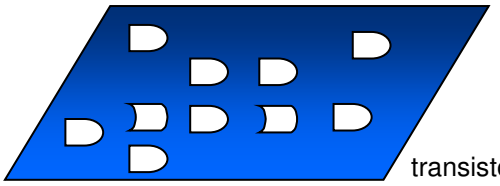
Electronic Systems Group
Faculty of Electrical Engineering
Technical University Eindhoven

on-chip interconnect: physical

3

on-chip interconnect

- essentially
  - a plane of transistors
  - with planes of wires on top, known as metal layers



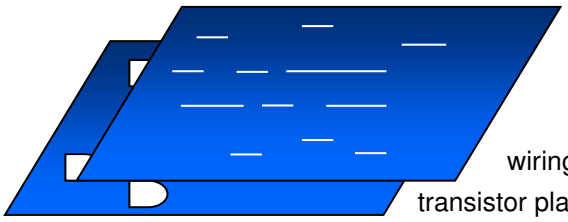
wiring plane 1

transistor plane

4

on-chip interconnect

- essentially
  - a plane of transistors
  - with planes of wires on top, known as metal layers



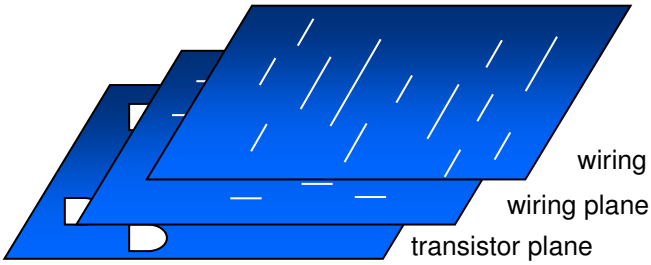
wiring plane 1

transistor plane

5

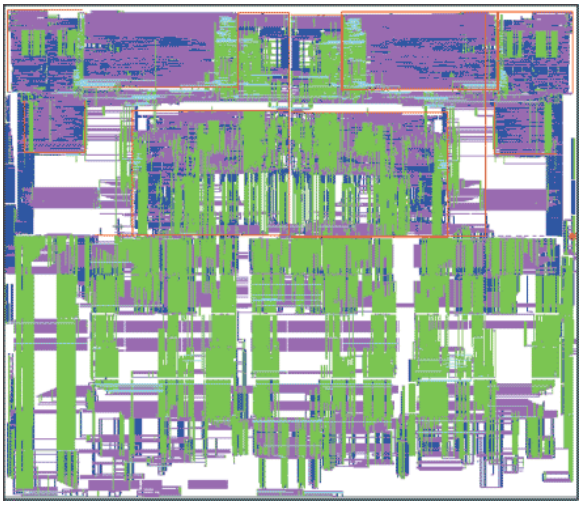
on-chip interconnect

- essentially
  - a plane of transistors
  - with planes of wires on top, known as metal layers



6

on-chip interconnect: PowerPC



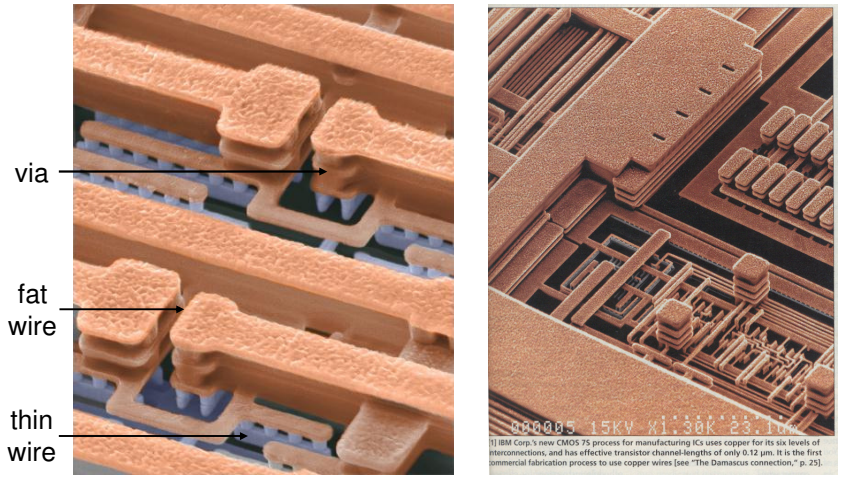
7

physical aspects

- essentially
  - a plane of transistors
  - with multiple planes of wires on top, known as metal layers
- usually, in a single plane, wires travel either in the X or in the Y direction
- transistors and the wires in different planes are connected with vertical wires, known as vias
- there are several kilometers of wire on a 1x1 cm chip!

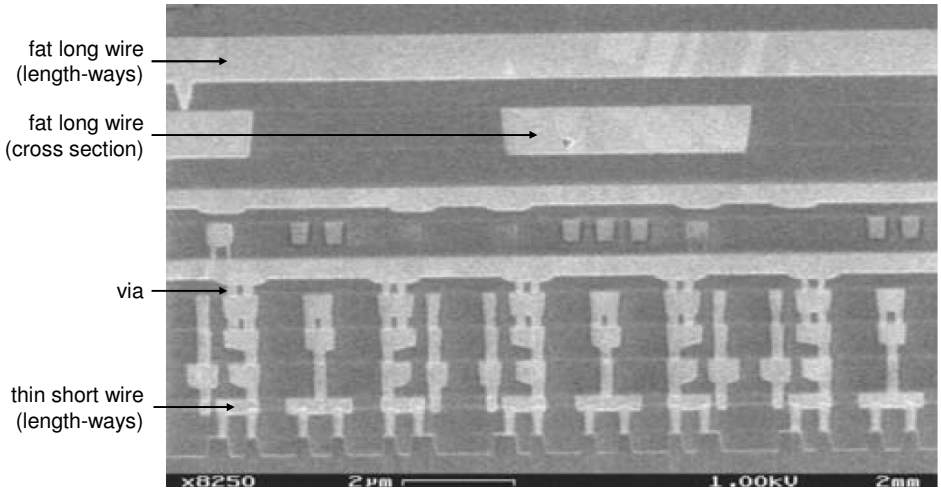
8

physical view



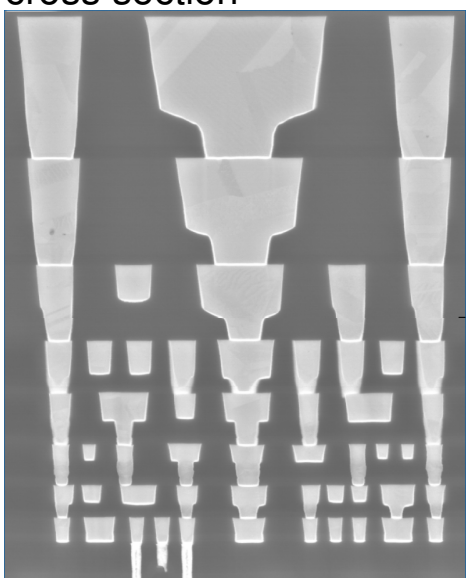
9

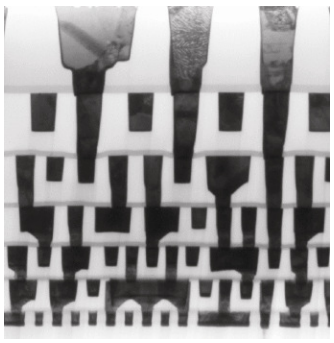
cross section



10

cross section



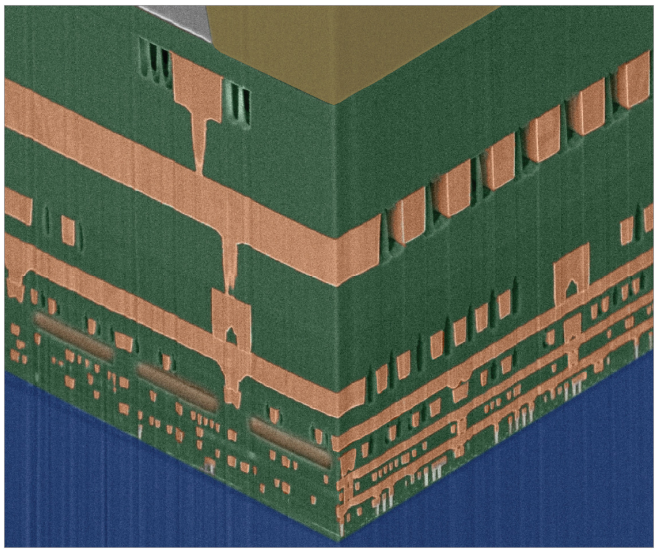


[source: Intel]

11

IBM ‘air gap’ technology

[source: IBM]



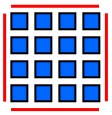
12

physical aspects

- ratio length:diameter determines the speed of the wire
- (assuming a signal must arrive at the destination in a single clock cycle)
- for local/close communication use thin short wires
- for remote/global communication use fat long wires
- in long wires the signal needs to be amplified to compensate for resistance losses, using buffers (e.g. inverters)
- (we ignore effects between multiple wires such as cross talk)

physical aspects (within one process generation)

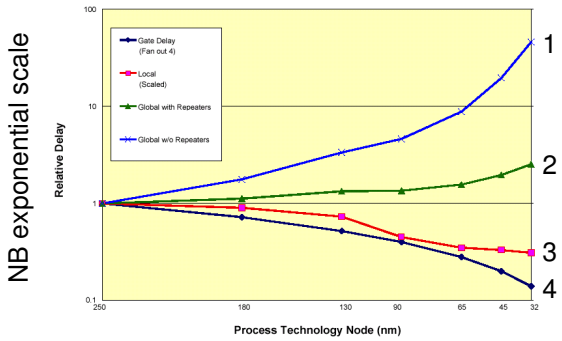
- wires are more limited than gates
  - intuitively, number of transistors increases quadratically (area)
  - but the number of wires into the area increases only linearly (border)
- as a result, connecting transistors (routing) increases chip area because gates must be spaced further apart



- moreover, long wires additionally require
  - buffers, which use space in the transistor plane, and
  - vias, which use space in the wiring planes
- some blocks have many (global) inputs and outputs
  - e.g. memory controller
  - gives rise to wire congestion (many wires converging on one place)
  - increasing chip area

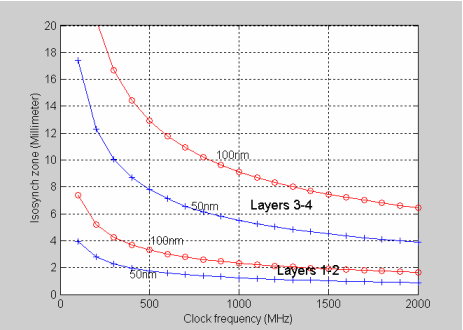
physical aspects (over process generations)

- from one process generation to the next (Moore's law)
  - transistors get smaller and faster [4]
  - wires get thinner, and slower
    - this is not an issue for local wires [3] that are short
  - but long global wires [1,2] become relatively slower than transistors



physical aspects (over process generations)

- distance that can be covered in a single clock cycle reduces with newer process generations
  - computation is cheap
  - communication is not!



below 100nm isochronous zones are smaller than 2x2 mm^2 [DeMan]

physical aspects (over process generations)

- problems
  - long global wires become relatively slower than transistors
  - distance that can be covered in a single clock cycle reduces with newer process generations
- solution
  - do not reduce the size of global wires
    (they remain as fast as in previous process generations)
- drawback
  - get more transistors but keep same number of global wires
  - i.e. (global) communication becomes relatively more expensive

17

on-chip interconnect: logical

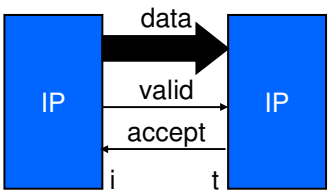
18

logical concepts

- connect up IP blocks directly
  - initiator, target
  - fixed timing vs. flexible timing: hand shake
- connect up IP blocks, with an interconnect
  - master, slave
- communication types
  - streaming data
  - memory-mapped communication

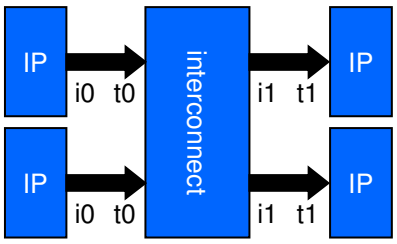
connecting two blocks directly

- hand shake
  1. initiator puts valid data on the data wires
  2. and indicates to the target when data is valid
  3. the target uses the data
  4. and indicates to the initiator that he has used (accepted) the data
- required if the blocks are not in the same clock domain, or if the transmission schedule is not static



connecting blocks using an interconnect

- master = first initiator
- slave = final target
- likely to have multiple masters and multiple slaves that can address each other



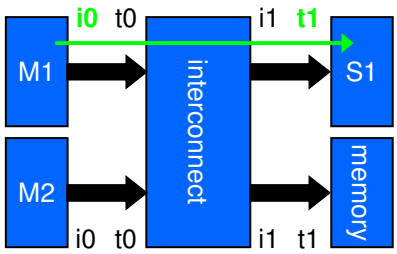
masters:
e.g. CPUs
streaming blocks

slaves:
e.g. memories,
memory controllers
streaming blocks

communication types

direction of data

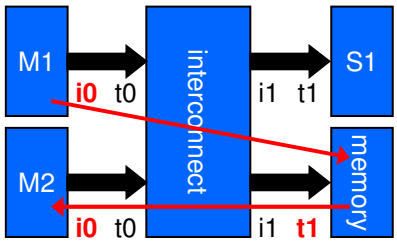
1. direct (IP-IP) streaming
   - discussed before: master to slave;
   - only requires writing



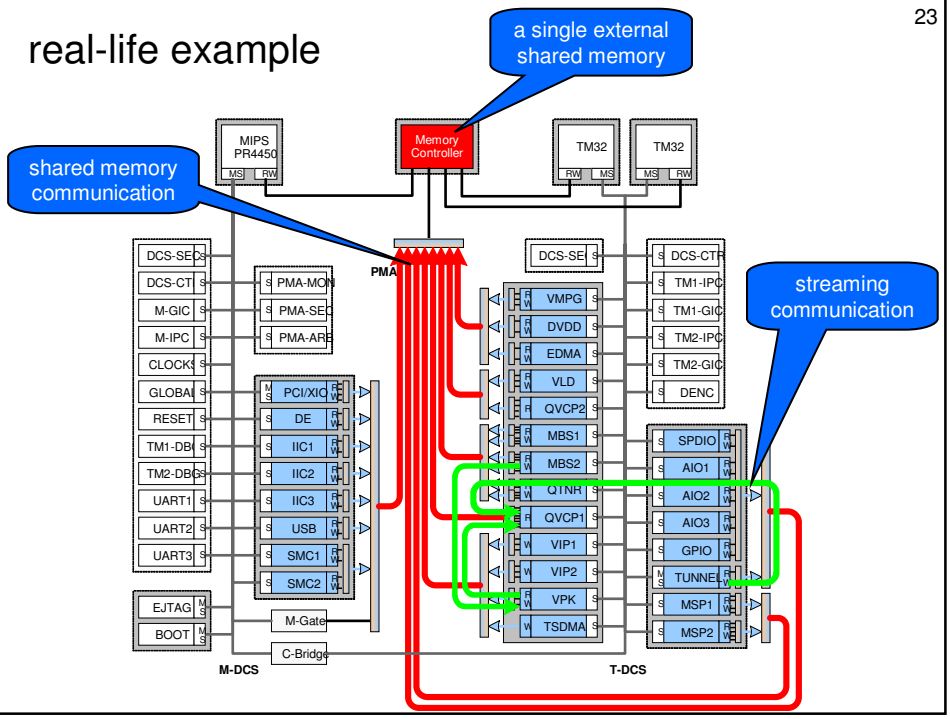
communication types

1. direct (IP-IP) streaming
   - discussed before: master to slave;
   - only requires writing
2. communication via shared memory (vast majority)
   - more complex: master 1 to slave & master 2 to slave
   - requires reading & writing

direction of data



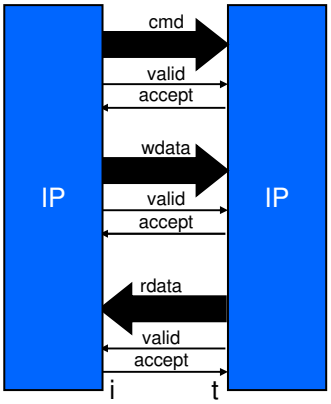
real-life example



shared-memory communication

- master 1 writes data in a memory
- master 2 reads data from a memory
  - ignore synchronisation (buffer under/overflow)
- require more complex transactions

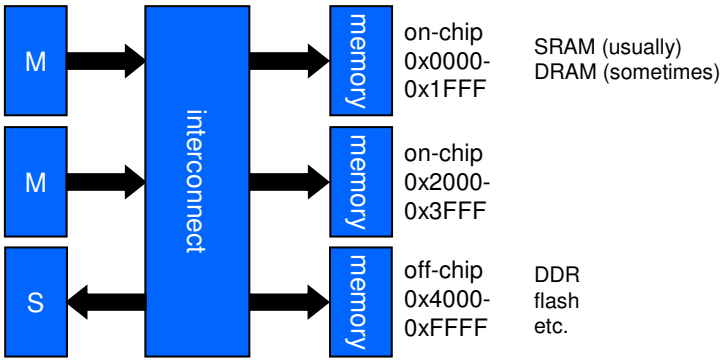
- signal groups
  - command group
    - read, write , ...
    - address, flags, …
  - write data group
    - data, mask / strobe, ...
  - read data group
    - data, errors, acknowledgements, ...
- independent valid/accept handshake per signal group



25

distributed-shared-memory communication

- logical address space is distributed over one or more memories
- often different memory types



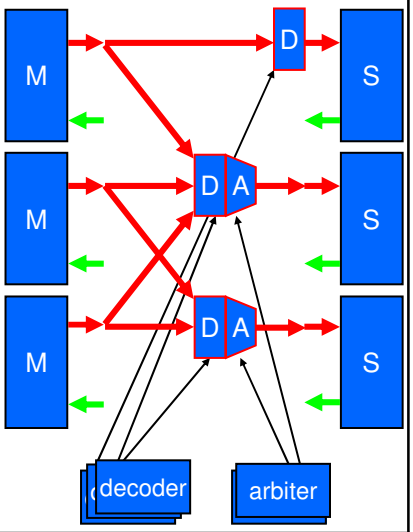
26

current on-chip interconnects

27

(abstract) interconnect implementations

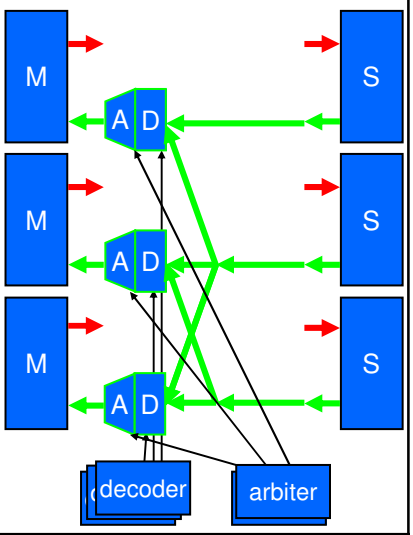
- ARM AXI, NXP DTL
- multiple masters
- split, pipelined
- concurrent, request access to the slave



28

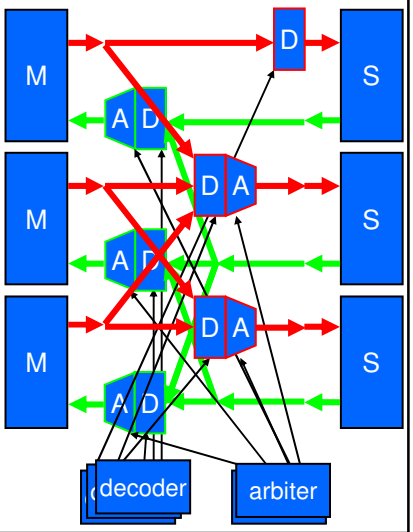
(abstract) interconnect implementations

- ARM AXI, NXP DTL
- multiple masters
- split, pipelined
- concurrent, request access to the slave
- broadcast response to all masters attached to one slave
- response arbiter for responses
- decoder for responses at master



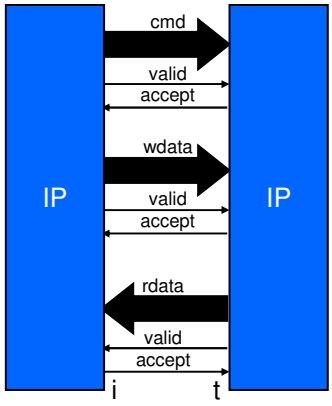
(abstract) interconnect implementations

- ARM AXI, NXP DTL
- multiple masters
- split, pipelined
- concurrent, request access to the slave
- masters can concurrently access different slaves
- maximum performance
  - split, pipelined, concurrent



logical interconnect problems

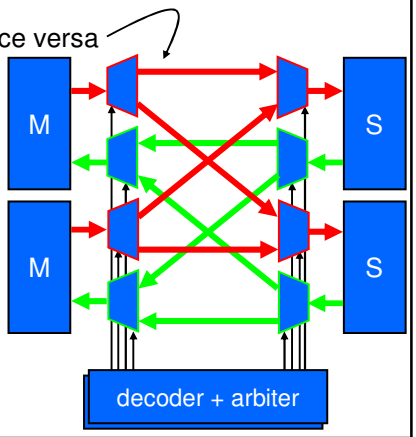
- many wires
  - ~200 per port



31

logical interconnect problems

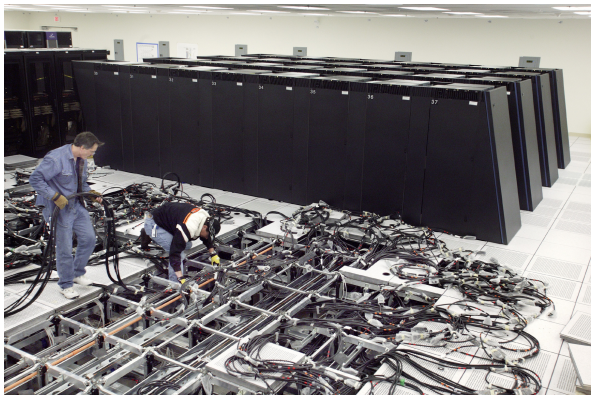
- many wires
  - ~200 per port
- long wires
  - running from master to slave and vice versa
  - Amba uses word (sub-transaction) pipelining to mitigate this
- under-utilised wires
  - e.g. address and write data wires
- restrictions in timing
  - older protocols used fixed timing, instead of valid/accept handshake
- not scalable
  - essentially, with N masters communicating to M slaves, requires NxM switch with wide links



32

super computers

- have the same problems...



IBM Bluegene

Cray

33

on-chip networks: concepts and introduction

34

interconnect problems

- physical
  - to avoid long global wires becoming relatively slower than transistors we don't reduce their size
  - they become more expensive instead
- logical
  - wires are not used efficiently
    - broadcast
    - unused signal groups
    - restrictive timing / protocol features
- transistors (computation) are cheap
- wires (communication) are not!

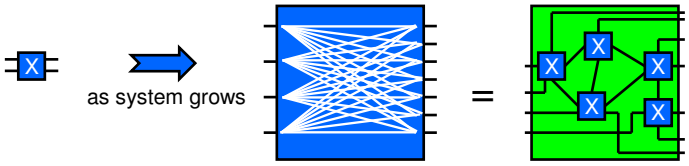
network on chip

- a NOC should offer
  - minimisation & efficient use of global wires
  - flexibility (programmable)
  - modularity & re-use
  - scalability
- fundamental network concepts
  - share wires
  - protocol layering

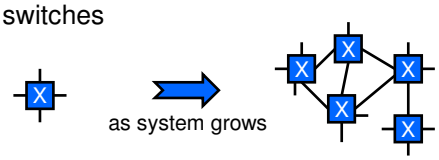
wire sharing

- NOC interconnects IP and can be seen

1. either as a large partial switch



2. or as a collection of small switches



- full switch is (almost always) overkill
- share wires between different communications

37

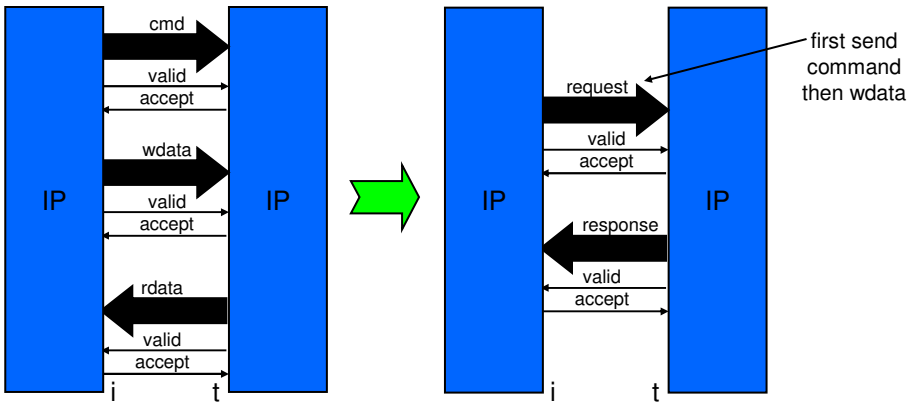
wire sharing

- wires can be shared because wires are active only ~10% of the time
- share wires to increase utilisation
- i.e. (statistical) multiplexing
  - on a single communication
  - on multiple communications
  - in different program phases
  - between different modes / use cases
  - etc.
- increase wire efficiency
- fewer wires result in
  - lower global wiring congestion
  - easier timing closure

38

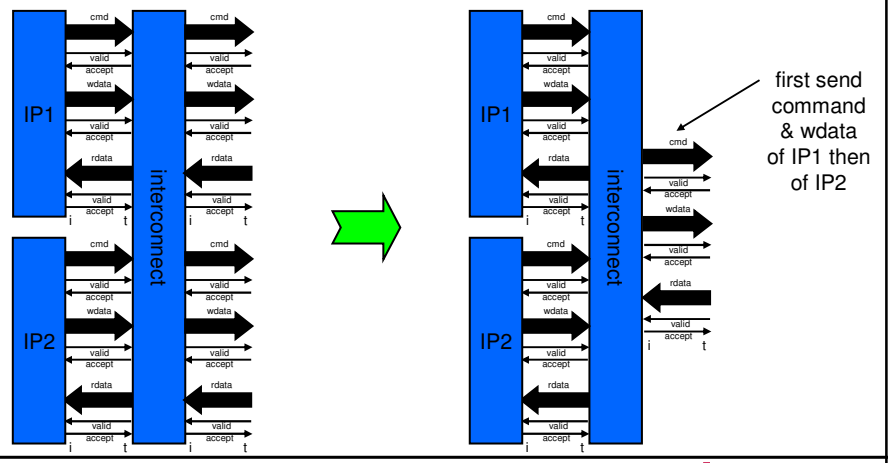
wire sharing: on one communication

- a single port between two IP blocks contains many wires, most of which are not active simultaneously
- e.g. serialise the command & write data groups on a request group



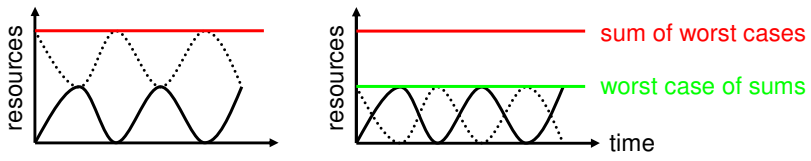
wire sharing: on multiple communications

- similarly, not all IP blocks communicate at the same time



wire sharing: on multiple communications

- multiplexing with other streams
  - average out variation of different streams

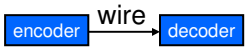


wire sharing

- is not free
  - multiplex & demultiplex
  - overhead of addressing, routing, etc.
- area & delay of
  - logic
  - buffering
  - wires
- energy

wire efficiency

- make wires more efficient
  - increase performance (& cost) of wire
  - only worthwhile for heavily used wires (otherwise only pay cost)
- however, to use the offered performance,
  the wires need to be heavily used, i.e. share the wires
- useful techniques:
  - fat wires
  - single driver
  - shielding (against cross-talk)
  - data encoding (for compression, against cross-talk delays)
  - low swing, possibly with error correction

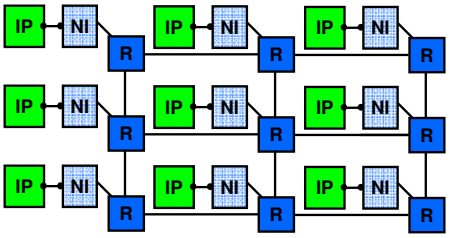


43

networks on chip concepts

- two types of components:
- routers
  - transport data in packets
- network interfaces
  - convert IP view (transactions) to network view (packets)

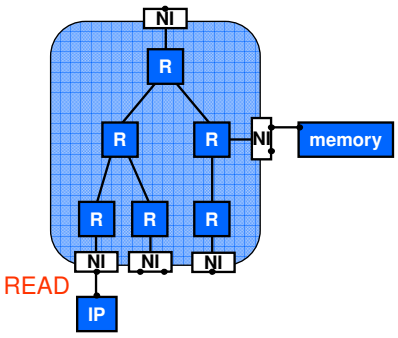
- example:
  mesh NOC



44

networks on chip: basics

1. IP does a read transaction

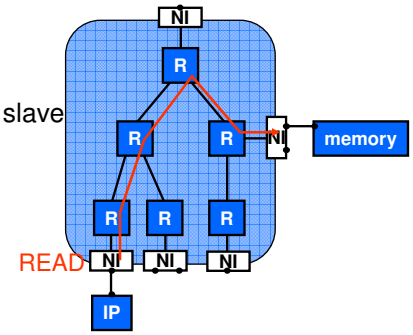


networks on chip: basics

1. IP does a read transaction
2. network interface (NI) packetises the transaction
   - chops into smaller packets
   - consisting of a header and payload

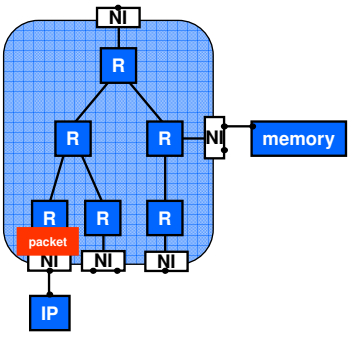
e.g. 32 bits wide
max 12 words

   - header contains the address of the slave or the path to the slave (and the ID of the sender, etc.)



networks on chip: basics

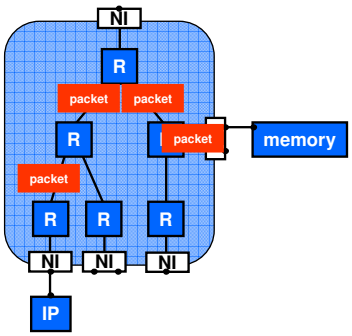
1. IP does a read transaction
2. network interface (NI) packetises the transaction
3. the NI sends the packet to the router



47

networks on chip: basics

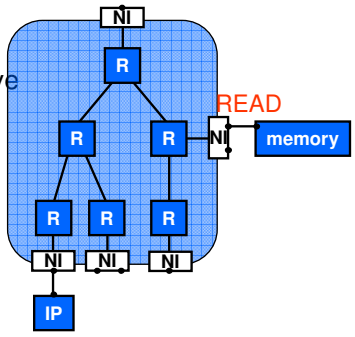
1. IP does a read transaction
2. network interface (NI) packetises the transaction
3. the NI sends the packet to the router
4. who forwards it, and so on



48

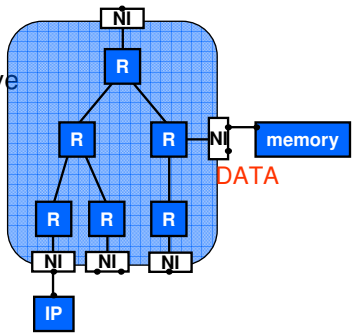
networks on chip: basics

1. IP does a read transaction
2. network interface (NI) packetises the transaction
3. the NI sends the packet to the router
4. who forwards it, and so on
5. the receiving NI unpacks the packet,
6. and presents the read transaction to the slave



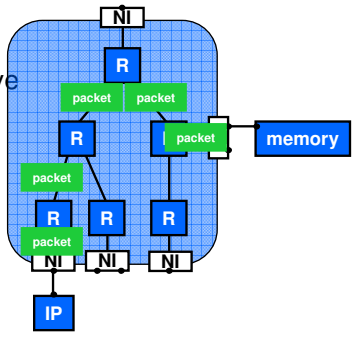
networks on chip: basics

1. IP does a read transaction
2. network interface (NI) packetises the transaction
3. the NI sends the packet to the router
4. who forwards it, and so on
5. the receiving NI unpacks the packet,
6. and presents the read transaction to the slave
7. the slave produces the data



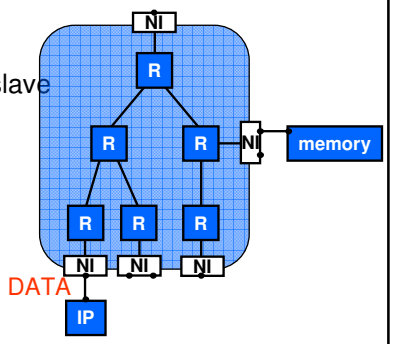
networks on chip: basics

1. IP does a read transaction
2. network interface (NI) packetises the transaction
3. the NI sends the packet to the router
4. who forwards it, and so on
5. the receiving NI unpacks the packet,
6. and presents the read transaction to the slave
7. the slave produces the data
8. the NI packetises & routers send



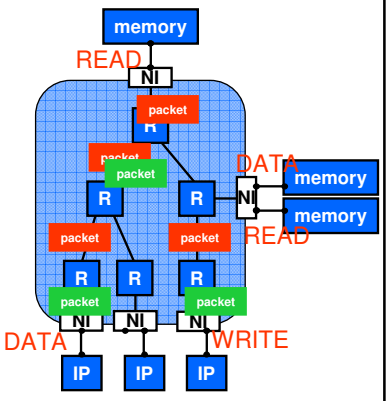
networks on chip: basics

1. IP does a read transaction
2. network interface (NI) packetises the transaction
3. the NI sends the packet to the router
4. who forwards it, and so on
5. the receiving NI unpacks the packet,
6. and presents the read transaction to the slave
7. the slave produces the data
8. the NI packetises & routers send
9. IP gets the data



networks on chip: basics

- high degree of parallelism
  - split request & response
  - pipelined transactions
  - concurrent transactions
- distributed arbitration



53

networks (on chip): concepts

- a fundamental network concept is protocol layering

1. decomposition: break problem in smaller pieces
2. abstraction: hide details
3. sharing: implement common services only once

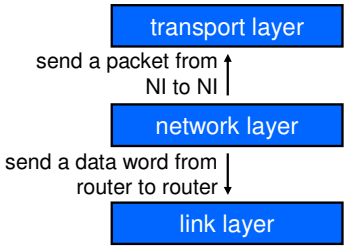
- main examples:
  - International Organisation for Standards (OSI) Open Systems Interconnect (OSI) reference model
  - Internet's TCP/IP reference model

- networks on chip architectures & implementations mostly follow OSI

54

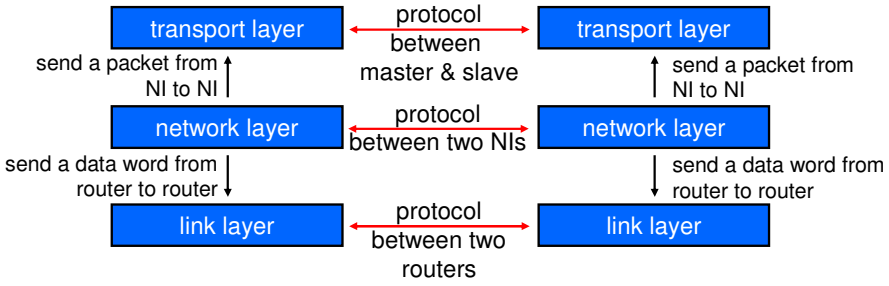
protocol layering: decomposition

- each layer offers services to higher layers, using the services of lower layers
  - e.g. send a bit over a link
  - send a packet from NI to NI
  - set up a connection between a master and a slave



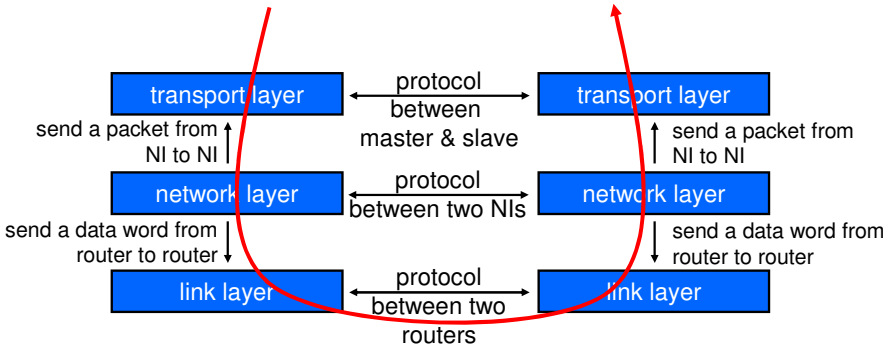
protocol layering: abstraction

- each layer uses a protocol
- hide details of implementation



protocol layering: abstraction

- each layer uses a protocol
- hide details of implementation



57

protocol layering: OSI & TCP/IP

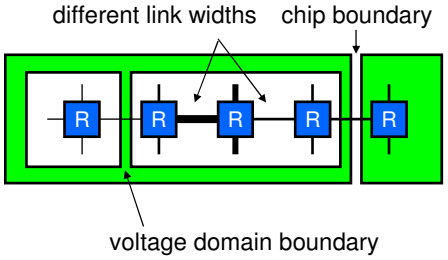
- ISO OSI reference model has 7 layers
  - application
  - presentation
  - session
  - transport
  - network
  - data link
  - physical
- the TCP/IP model has 4 layers
  - application
  - transport
  - internet
  - host-to-network

58

protocol layering: OSI on chip

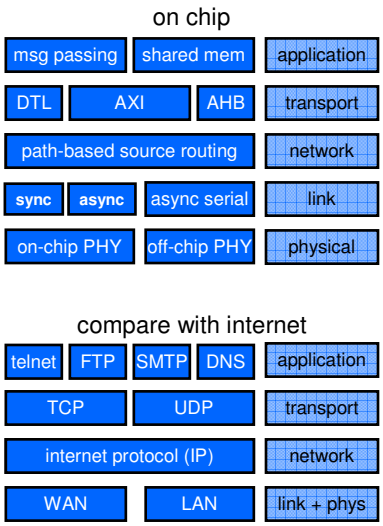
- as a result

1. each layer can be implemented, optimised, upgraded, etc. independently
   - e.g. for different process generations
2. multiple different implementations of a layer can exist
   - e.g. different link widths & types
3. in the same system



protocol layering: OSI on chip

- application
  - message passing, distributed shared memory
- transport
  - DTL, AXI, AHB, …
- network
  - different protocol optimisations
  - buffering, switching, routing
- link
  - width, synchronous, source-synchronous, asynchronous
- physical
  - serial / parallel, on / off chip, voltage / power domain



network (on chip) architecture concepts

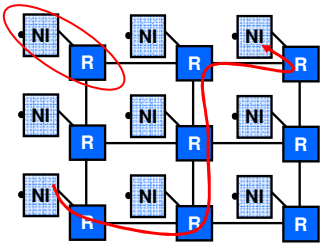
network (on chip) concepts

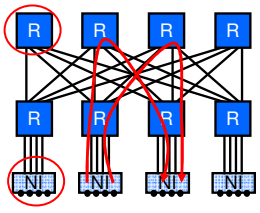
- topology
  - how to interconnect routers, network interfaces, and IP blocks
- routing
  - the path packets take through the network
- flow control
  - how packets are moved through the network
- buffer management / back pressure
  - how to deal with full buffers
- quality of service
  - how to offer minimum throughput, maximum latency, ...
- design flows
  - how to design, instantiate, configure, program networks

topology

- network interface
- router
- link
- path = series of links = route
- path length = hop count
- minimal path
- diameter of network = largest minimal hop count over all pairs of terminal nodes
- path diversity = the number of minimal paths

- regular networks
  - butterfly (k-ary n-fly), fat tree
  - torus (k-ary n-cubes), mesh, torus,
- irregular networks

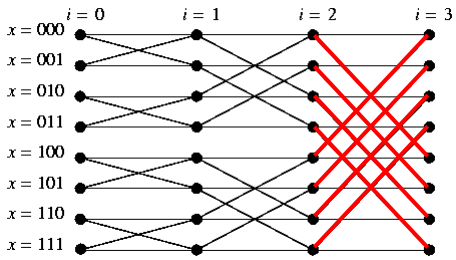




topology: butterfly

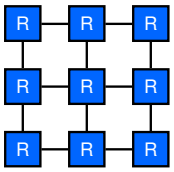
+ minimum diameter (logarithmic)
- no path diversity
- requires long wires
  - traverse at least half the diameter
  - this does not fulfil our interconnect requirements

- k-ary n-fly
  - k^n source terminals
  - k^n destination terminals
  - n stages of k^(n-1) (k)x(k) switch nodes
  - links are unidirectional

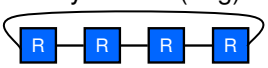


topology: torus

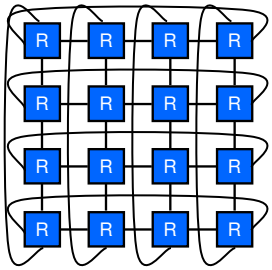
3-ary 2-mesh



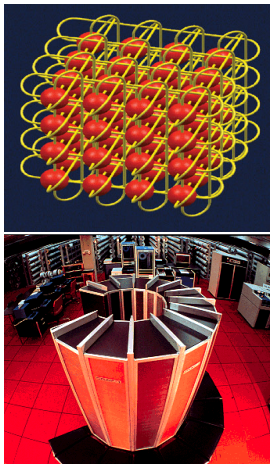
4-ary 1-cube (ring)



4-ary 2-cube (torus)



4-ary 3-cube (hypercube)



65

topology: torus

+ at low dimensions have short wires
+ good path diversity
+ can make use of locality
  – i.e. talking to neighbours is cheap, unlike in butterfly networks

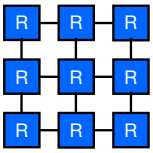
- larger hop count than butterfly networks

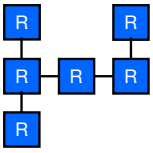
- to minimise both latency & wire length
  take minimal n that makes network bisection limited

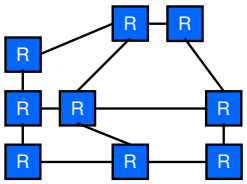
66

topology: real life

- recall that chips consist of a transistor plane with X-Y wiring planes on top
- mesh, folded torus, and express cubes are natural candidates for on-chip topologies
- however, the lay-out of a chip is hardly ever regular
- compromises
  – partial mesh, omit some nodes, mesh lay-out
  – logically a mesh, morphed lay-out



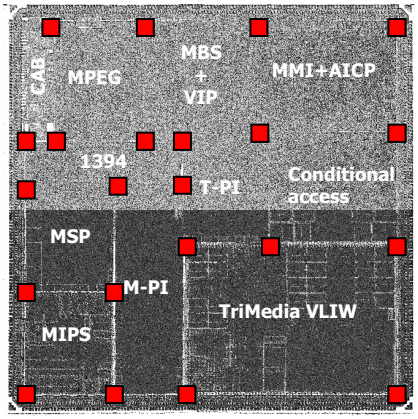




67

topology: real life

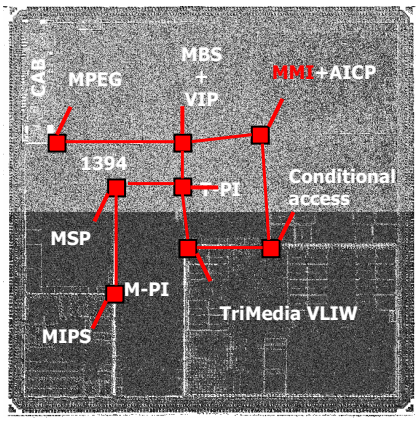
- routers at all corners of IP blocks
- can use channel routing
  - use space between IP block for long-distance (NOC) wires
- the alternative is routing the NOC wires over the IP blocks
  - not always possible, e.g. for MIPS & TriMedia that require all metal layers for local wiring



68

topology: real life

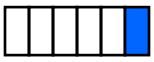
- minimal topology with one router per IP block / island
- almost all traffic is bound for the memory controller (MMI)
- the MIPS and Trimedia need low latency access and should have minimum number of hops
- in Viper 30% of long global wires are due to MMIO bus



flow control

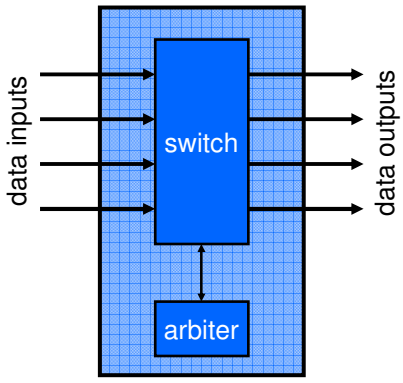
general router architecture

- move packets that arrive at inputs to outputs



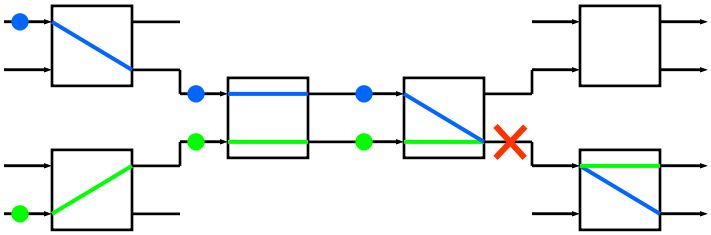
- for all-at-once (source-based) routing the packet header contains the path
- for incremental routing it contains the destination address

- require arbiter in case two packets go to same output at the same time
  - (weighed) round robin
  - priority
  - last recently used, etc.



contention

- NOCs ➔ multiplexing / sharing of wires between multiple data flows
- sharing implies contention:
- multiple packets / data
  - at the same place
  - at the same time



flow control

- techniques to deal with contention:
  when two packets arrive at the same link at the same time
  - or any other shared resource
- options
  1. avoid contention altogether
  2. deal with contention when it happens
- with or without buffering packets

73

flow control

- when two (or more) packets arrive at the same link at the same time
- buffer-less flow control
  1. avoid contention (circuit switching), or
  – deal with contention
    - drop both packets
    2. forward one packet & drop one packet
    3. forward both packets, but one to wrong output (misrouting)
- buffered flow control
  – deal with contention
    - delay both packets
    4. forward one packet & delay one packet
      – store & forward, virtual cut through, worm-hole, virtual channel

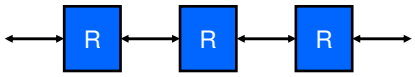
74

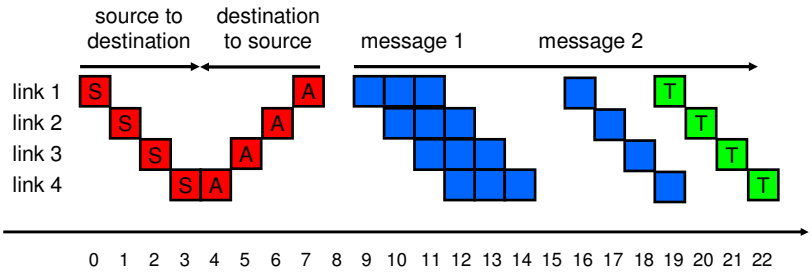
buffer-less flow control: circuit switching

- circuit switching
- first allocate a circuit from source to destination, reserving links
- then send data on circuit, guaranteed without contention
- deallocate circuit, freeing links
- like old telephone system
- used in the asynchronous transfer mode (ATM) networks

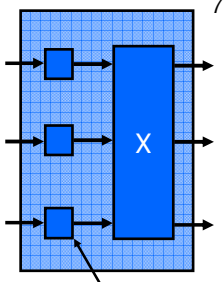
buffer-less flow control: circuit switching

- four phases
  1. set up (S)
  2. acknowledge (A)
  3. send data
  4. tear down (T)





buffer-less flow control: dropping



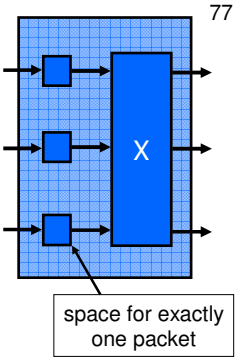
- drop both packets
- forward one packet & drop one packet

- need
  - acknowledgements,
  - re-sequencing,
  - (negative) acknowledgement (ack/nack) or timers,
  - retransmission, (and buffering until acknowledgement)
  - duplicate removal
- cannot guarantee delivery
  and hence cannot guarantee throughput and latency

- minimal latency & buffering per router

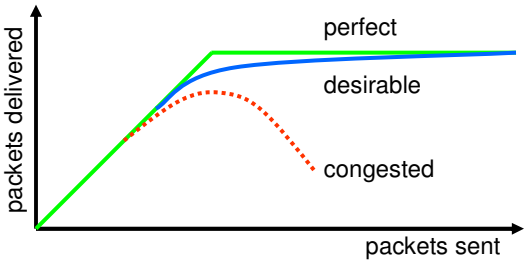
77

buffer-less flow control: dropping



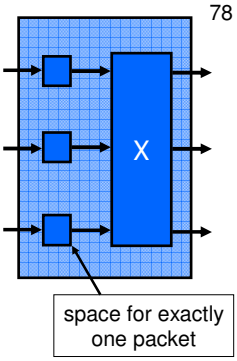
- drop both packets
- forward one packet & drop one packet

- inefficient in resource usage (links, buffers)
  - non-minimal paths
  - data is resent



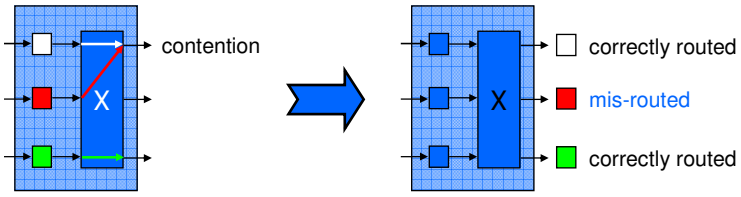
78

buffer-less flow control: misrouting



- forward both packets, but one to wrong output

- also known as deflection routing or hot-potato routing
- no packets are dropped
- still need re-sequencing
- must ensure that packet will arrive at its destination
  - live lock, time to live, ...
- minimal latency & buffering per router



buffered flow control: store and forward

- no distinction between packets & flits
- flits are shown for easier comparison later with VCT & WH

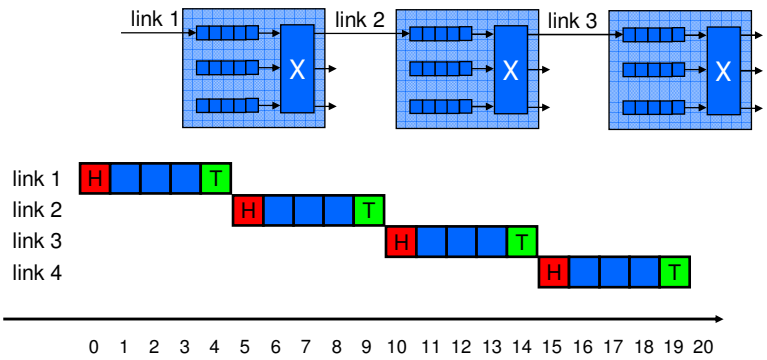
- packet is forwarded to next router
  - when the current router has received the whole packet
  - and when there the next router has space for the whole packet

- latency: transmission of entire packet
- buffering: at least one packet

- when a packet cannot proceed, it waits in one router
  - uses one buffer, does not block links

buffered flow control: store and forward

- packet is forwarded to next router
  - when the current router has received the whole packet
  - and when there the next router has space for the whole packet



buffering schemes (for QoS)

what is quality of service

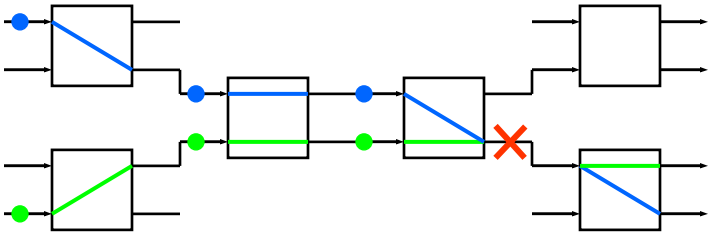
- in our context:

- lossless data transport
- uncorrupted data transport
- minimum throughput
- maximum latency
- maximum jitter

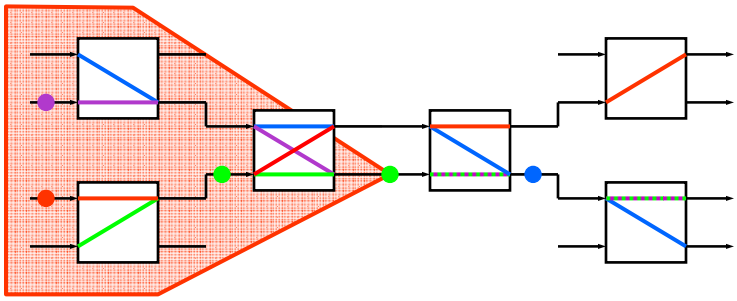
- low latency
  - cache misses, interrupts
- guaranteed bandwidth
  - audio, video, ...
- best effort
  - cache prefetches, write backs, debug / monitoring information
  - GUI, browsing, file transfers, ...

what is the problem: contention

- NOCs = multiplexing / sharing of wires between multiple data flows
- sharing implies contention:
- multiple packets
  - at the same place
  - at the same time



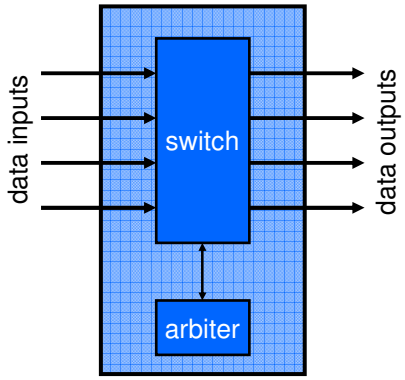
what is the problem: congestion



- congestion
  - packets wait for waiting packets wait for …
- even though they may not want to use the original contended link
- buffering strategies are not enough

general router architecture

- move packets that arrive at inputs to outputs
- in case two packets go to the same output at the same time
  - arbiter decides which packets proceed first
  - move packets through switch
  - buffer remaining packets



overview

- important components for QoS

1. flow control
2. buffering strategy
3. switch architecture
4. switch arbitration

- only with the right combination can a NOC offer QoS
- more generally
  - what resources are shared
  - how are they allocated / scheduled
  - what can be pre-empted and what not
- we show a few solutions to offer guaranteed bandwidth & latency

buffering schemes

1. input buffering
2. output buffering
3. virtual-output buffering
4. virtual-circuit buffering
5. per-slave buffering

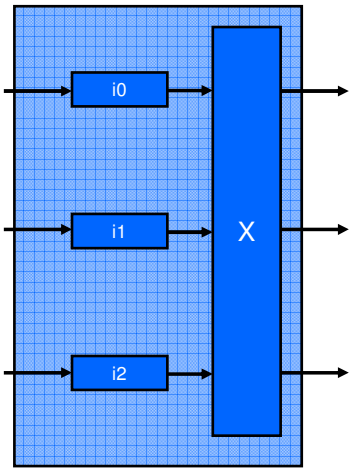
- combinations are possible too
  - e.g. input & output buffering

- we compare them on
  - # logical buffers
  - # physical memories & type
  - size of switch(es)
  - performance & cost

buffering schemes: input buffering

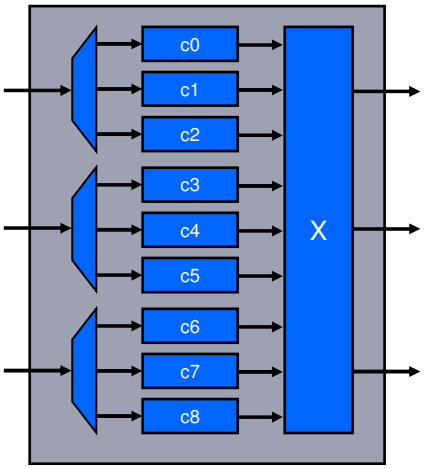
- N = degree = # inputs = # outputs
- N logical buffers: one per input
- N physical FIFOs
- N*N switch

- maximum ~58% utilisation
- simplest & cheapest
- worst performance



buffering schemes: virtual circuit buffering

- with non-blocking switch
- C logical buffers: one per circuit
- C physical FIFOs
- C*N switch
- gives potentially higher throughput than with blocking switch

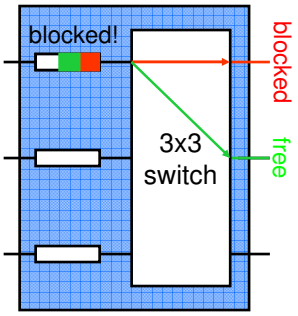


buffering schemes: problem

- buffered flow control

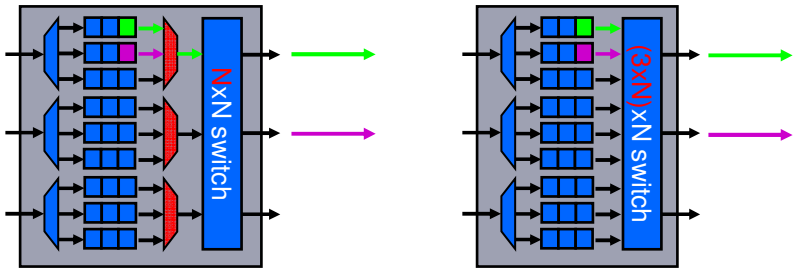
1. input buffering exhibits head-of-line (HOL) blocking
2. virtual output buffering has similar problem
3. output buffering has similar problem
4. buffering per slave
   - HOL blocking per slave
5. no buffer sharing at all
   - virtual circuit buffering

- in all but 5 flows interfere with each other due to shared buffers



buffering schemes: relation with switch

- blocking versus non-blocking
  - for virtual output, virtual circuit, per slave, virtual channel buffering



- blocking switch introduces extra dependencies between flows
  - reduces performance
  - harder to analyse performance
  - but has lower cost
    - e.g. Æthereal 6x6 130nm: 0.13mm2 blocking vs. 0.17mm2 non-blocking

to remember

- problem: wires / communication
  - physical: global wires become relatively expensive
  - logical: wires are inefficiently used
- solution: networks on chip
  - share wires
  - protocol stack
- architecture choices on
  - topology
  - routing
  - flow control
  - buffering
  - quality of service
  - design flows

the end