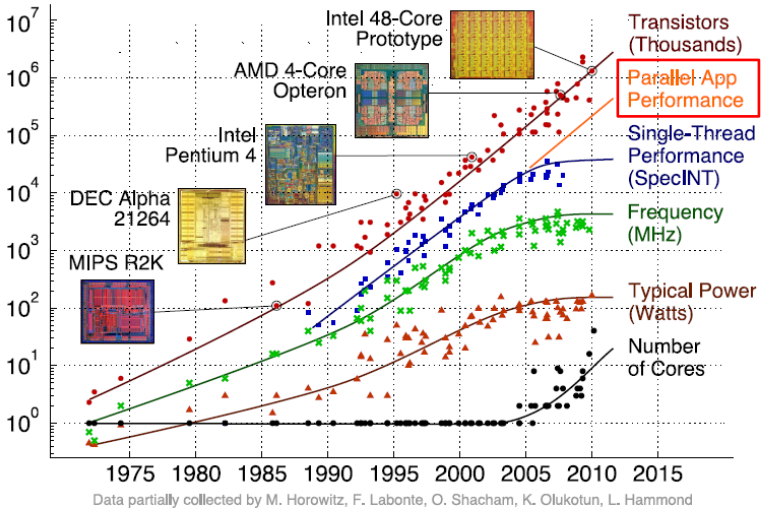
COEN-4730 Computer Architecture

Lecture 8 (part 2) Networks-on-Chip (NoC)

(Ch.5)

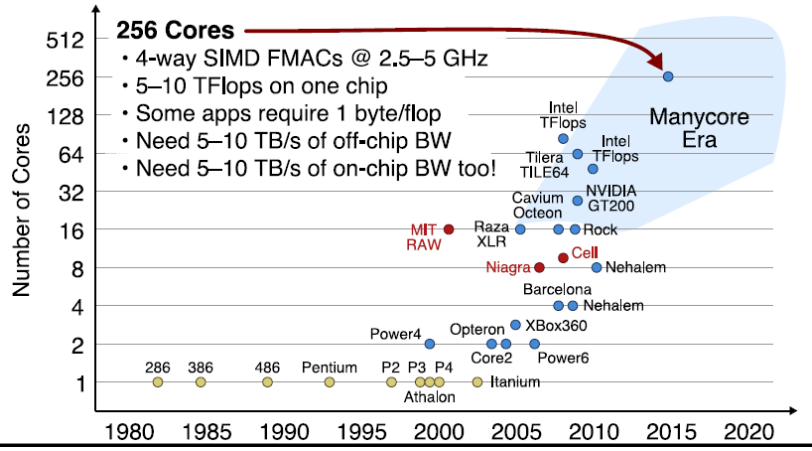
Cris Ababei
Dept. of Electrical and Computer Engr., Marquette University

Why study chip-level networks?



The future of multicore

- Parallelism replaces clock frequency scaling and core complexity
- Resulting Challenges…
  - Scalability, Programming, Power



3

3

Outline

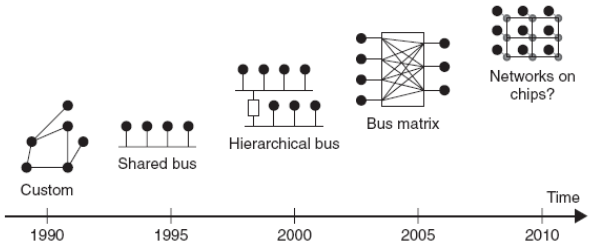
- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

4

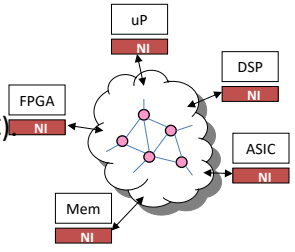
4

Introduction

- Evolution of on-chip communication architectures



- **Network-on-chip (NoC)** is a packet switched on-chip communication network designed using a layered methodology. NoC is a communication centric design paradigm for System-on-Chip (SoC).
- Rough classification:
  - Homogeneous
  - Heterogeneous

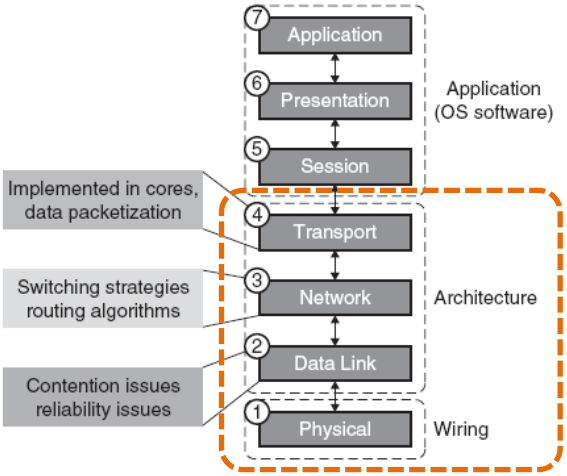


5

5

Introduction

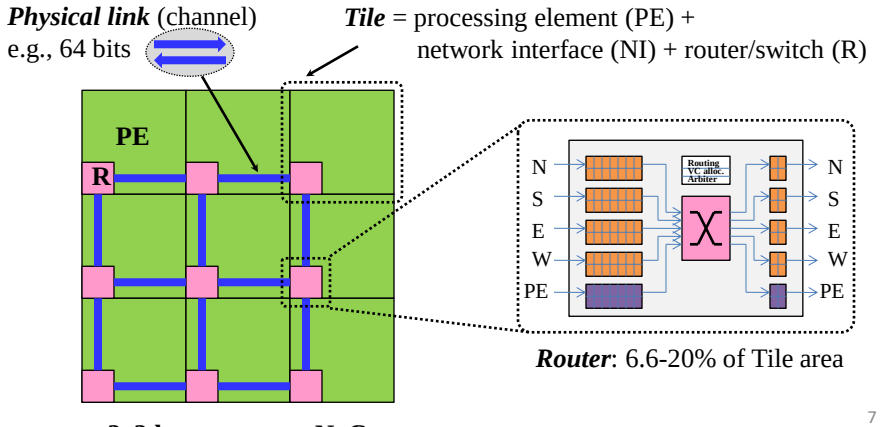
- Open Systems Interconnect (OSI) network protocol stack model



6

6

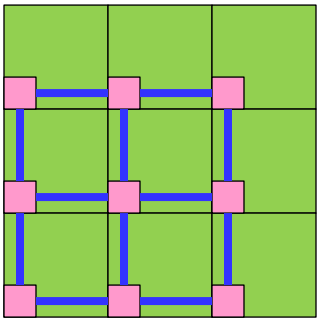
- NoCs borrow ideas and concepts from computer networks → apply them to the embedded SoC domain.
- NoCs use packets to route data from the source PE to the destination PE via a network fabric that consists of
  - Network interfaces/adapters (NI)
  - Routers (a.k.a. switches)
  - interconnection links (channels, wires bundles)

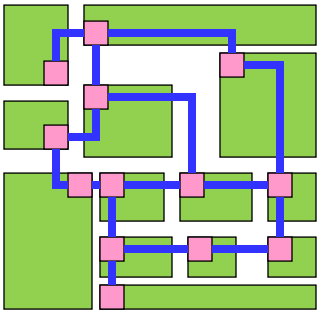


7

7

Homogeneous vs. Heterogeneous





- Homogenous:
  - Each tile is a simple processor
  - Tile replication (scalability, predictability)
  - Less performance
  - Low network resource utilization
- Heterogeneous:
  - IPs can be: General purpose/DSP processor, Memory, FPGA, IO core
  - Better fit to application domain
  - Most modern systems are heterogeneous
  - Topology synthesis: more difficult
  - Needs specialized routing

8

8

NoC properties

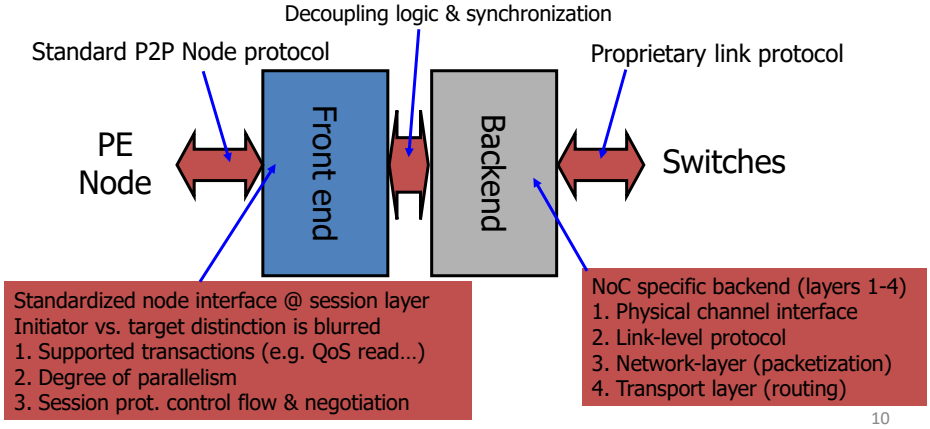
- Reliable and predictable electrical and physical properties → Predictability
- Regular geometry → Scalability
- Flexible QoS guarantees
- Higher bandwidth
- Reusable components
  - Buffers, arbiters, routers, protocol stack

9

9

Building blocks: NI

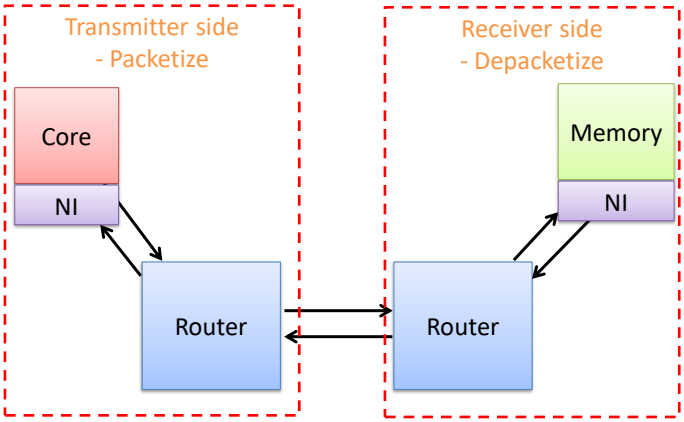
- Session-layer (P2P) interface with nodes
- Back-end manages interface with switches



10

10

Building blocks: NI

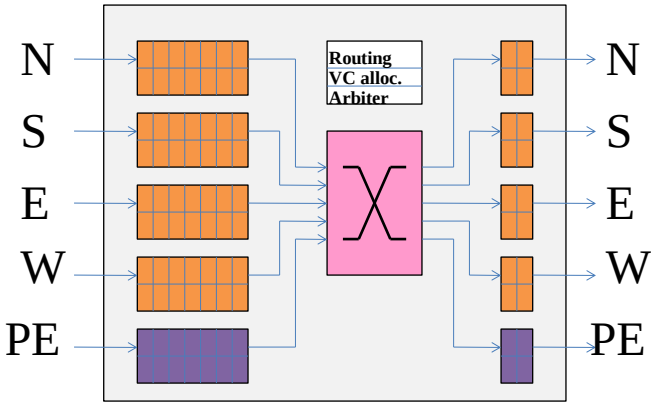


11

11

Building blocks: Router

- Router: receives and forwards packets
- Buffers:
  - Queuing
  - *Decouple* the allocation of adjacent channels in time
  - Can be organized as virtual channels.

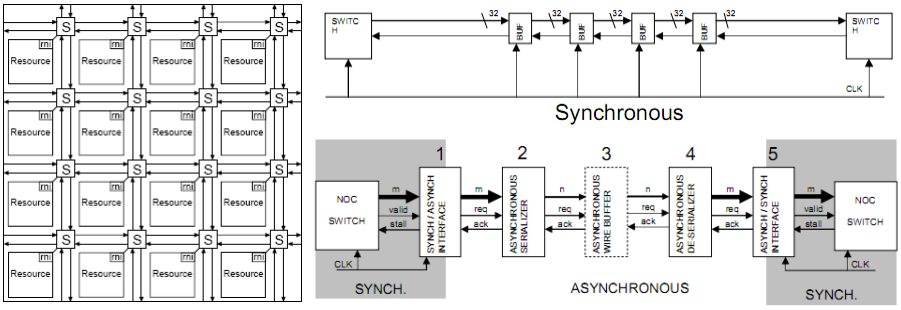


12

12

Building blocks: Links

- Connects two routers in both directions on a number of wires (e.g., 32 bits)
- In addition, wires for control are part of the link too
- Can be pipelined (include handshaking for asynchronous)

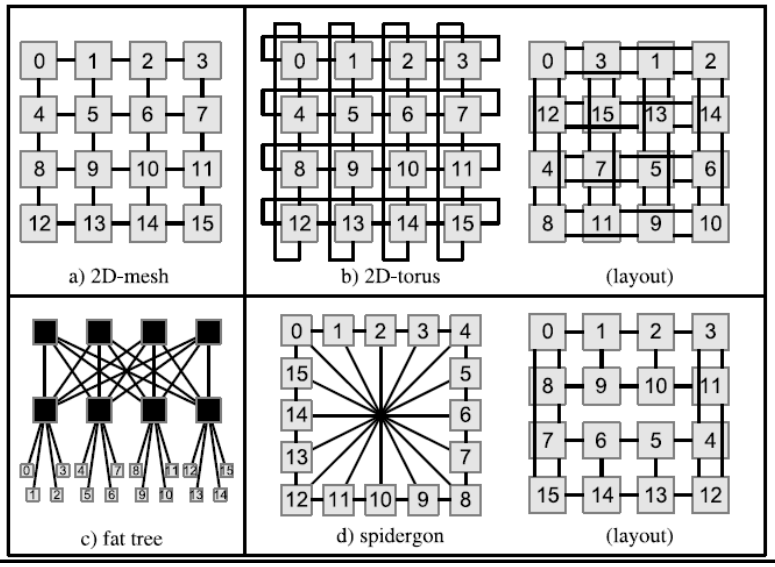


Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

NoC topologies

- "The **topology** is the network of streets, the roadmap".

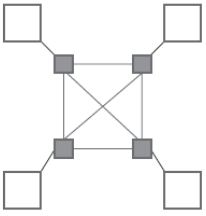


15

15

Direct topologies

- Direct Topologies
  - Each node has direct point-to-point link to a subset of other nodes in the system called neighboring nodes
  - As the number of nodes in the system increases, the total available communication bandwidth also increases
  - Fundamental trade-off is between connectivity and cost
- Most direct network topologies have an orthogonal implementation, where nodes can be arranged in an n-dimensional orthogonal space
  - e.g. n-dimensional mesh, torus, folded torus, hypercube, and octagon

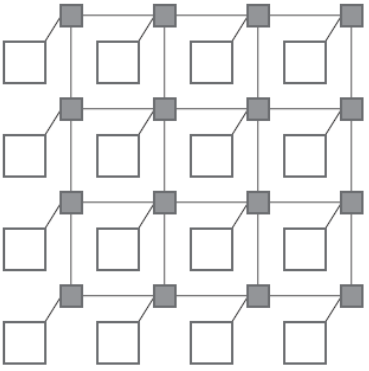


16

16

2D-mesh

- It is most popular topology
- All links have the same length
  - eases physical design
- Area grows linearly with the number of nodes
- Must be designed in such a way as to avoid traffic accumulating in the center of the mesh

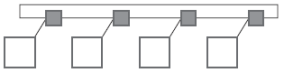


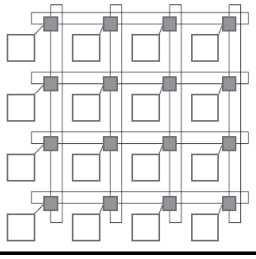
17

17

Torus

- Torus topology, also called a k-ary n-cube, is an n-dimensional grid with k nodes in each dimension
- k-ary 1-cube (1-D torus) is essentially a ring network with k nodes
  - limited scalability as performance decreases when more nodes
- k-ary 2-cube (i.e., 2-D torus) topology is similar to a regular mesh
  - except that nodes at the edges are connected to switches at the opposite edge via wrap-around channels
  - long end-around connections can, however, lead to excessive delays



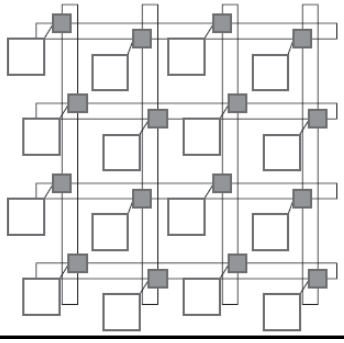


18

18

Folding torus

- Folding torus topology overcomes the long link limitation of a 2-D torus links have the same size
- Meshes and tori can be extended by adding bypass links to increase performance at the cost of higher area

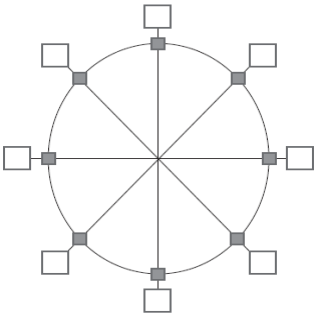


19

19

Octagon

- Octagon topology is another example of a direct network
  - messages being sent between any 2 nodes require at most two hops
  - more octagons can be tiled together to accommodate larger designs by using one of the nodes as a bridge node

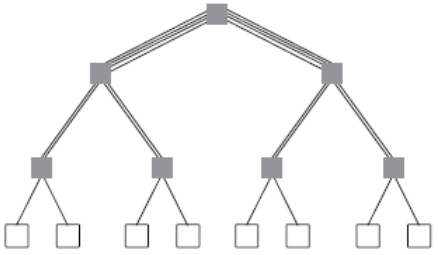


20

20

Indirect topologies

- Indirect Topologies
  - each node is connected to an external switch, and switches have point-to-point links to other switches
  - switches do not perform any information processing, and correspondingly nodes do not perform any packet switching
  - e.g. SPIN, crossbar topologies
- Fat tree topology
  - nodes are connected only to the leaves of the tree
  - more links near root, where bandwidth requirements are higher

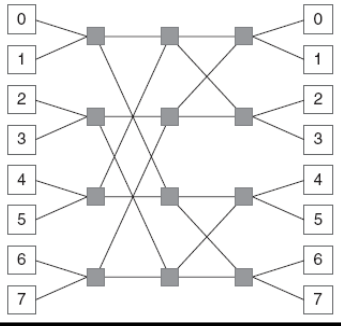


21

21

Butterfly

- k-ary n-fly butterfly network
  - blocking multi-stage network – packets may be temporarily blocked or dropped in the network if contention occurs
  - kn nodes, and n stages of kn-1 k x k crossbar
  - e.g., 2-ary 3-fly butterfly network

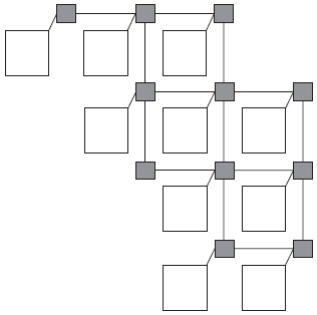


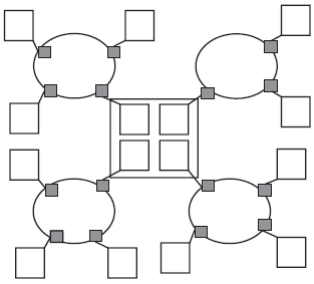
22

22

Irregular topologies

- Irregular or ad-hoc network topologies
  - customized for an application
  - usually a mix of shared bus, direct, and indirect network topologies
  - e.g., reduced mesh, cluster-based hybrid topology





Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

Routing algorithms

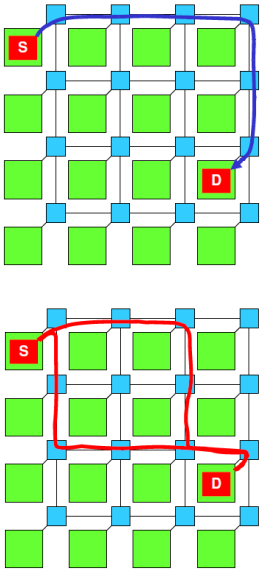
- **Routing** is the route/path (a sequence of channels) of streets from source to destination. “The routing method steers the car”.
- Routing determines the path followed by a message through the network to its final destination.
- Responsible for correctly and efficiently routing packets or circuits from the source to the destination
  - Path selection between a source and a destination node in a particular topology
- Ensure load balancing
- Latency minimization
- Flexibility w.r.t. faults in the network
- Deadlock and livelock free solutions
- Routing schemes/techniques/algos can be classified/looked-at as:
  - Static or dynamic routing
  - Distributed or source routing
  - Minimal or non-minimal routing

25

25

Static/deterministic vs. Dynamic/adaptive Routing

- **Static routing**: fixed paths are used to transfer data between a particular source and destination
  - does not take into account current state of the network
- advantages of static routing:
  - easy to implement, since very little additional router logic is required
  - in-order packet delivery if single path is used
- **Dynamic/adaptive routing**: routing decisions are made according to the current state of the network
  - considering factors such as availability and load on links
- path between source and destination may change over time
  - as traffic conditions and requirements of the application change
- more resources needed to monitor state of the network and dynamically change routing paths
- able to better distribute traffic in a network

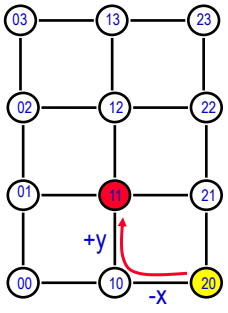


26

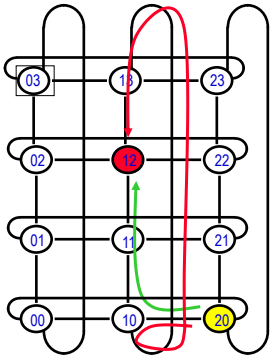
26

Example: Dimension-order Routing

- Static **XY routing** (commonly used):
  - a deadlock-free shortest path routing which routes packets in the X-dimension first and then in the Y-dimension
- Used for tori and mesh topologies
- Destination address expressed as absolute coordinates
- It may introduce imbalance → low bandwidth



For torus, a preferred direction may have to be selected. For mesh, the preferred direction is the only valid direction.

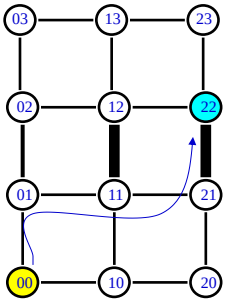


27

27

Example: Dynamic Routing

- A locally optimum decision may lead to a globally sub-optimal route



To avoid slight congestion in (01-02), packets then incur more congested links

28

28

Routing mechanics: Distributed vs. Source Routing

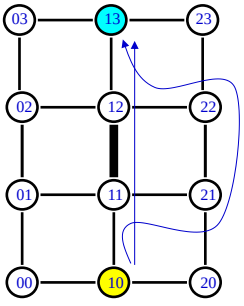
- **Routing mechanics** refers to the mechanism used to implement any routing algorithm.
- **Distributed routing**: each packet carries the destination address
  - e.g. XY co-ordinates or number identifying destination node/router
  - routing decisions are made in each router by looking up the destination addresses in a routing table or by executing a hardware function
- **Source routing**: packet carries routing information
  - pre-computed routing tables are stored at NI
  - routing information is looked up at the source NI and routing information is added to the header of the packet (increasing packet size)
  - when a packet arrives at a router, the routing information is extracted from the routing field in the packet header
  - does not require a destination address in a packet, any intermediate routing tables, or functions needed to calculate the route

29

29

Minimal vs. Non-minimal Routing

- **Minimal routing**: length of the routing path from the source to the destination is the shortest possible length between the two nodes
  - source does not start sending a packet if minimal path is not available
- **Non-minimal routing**: can use longer paths if a minimal path not available
  - by allowing non-minimal paths, the number of alternative paths is increased, which can be useful for avoiding congestion
  - disadvantage: overhead of additional power consumption

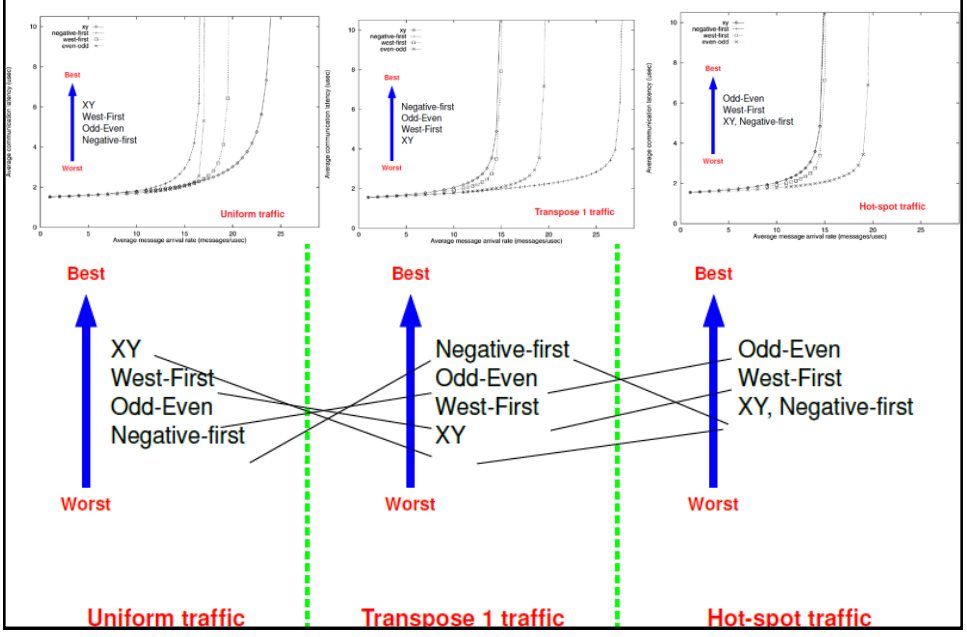


Minimal adaptive routing
is unable to avoid congested links
in the absence of minimal path diversity

30

30

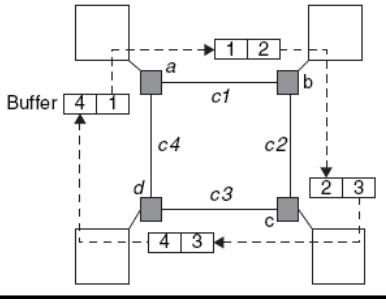
No winner routing algorithm



31

Routing Algorithm Requirements

- Routing algorithm must ensure freedom from deadlocks
  - **Deadlock**: occurs when a group of agents, usually packets, are unable to progress because they are waiting on one another to release resources (usually buffers and channels).
  - common in WH switching
  - e.g. cyclic dependency shown below
  - freedom from deadlocks can be ensured by allocating additional hardware resources or imposing restrictions on the routing
  - usually dependency graph of the shared network resources is built and analyzed either statically or dynamically



32

32

Routing Algorithm Requirements

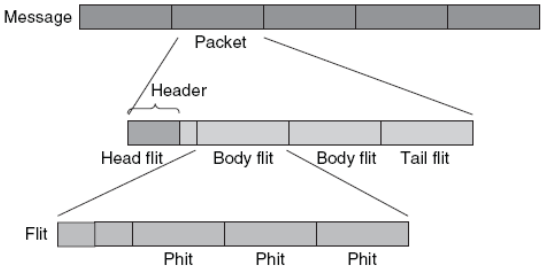
- Routing algorithm must ensure freedom from **livelocks**
  - livelocks are similar to deadlocks, except that states of the resources involved constantly change with regard to one another, without making any progress
  - occurs especially when dynamic (adaptive) routing is used
  - e.g. can occur in a deflective "hot potato" routing if a packet is bounced around over and over again between routers and never reaches its destination
  - livelocks can be avoided with simple *priority* rules
- Routing algorithm must ensure freedom from **starvation**
  - under scenarios where certain packets are prioritized during routing, some of the low priority packets never reach their intended destination
  - can be avoided by using a *fair* routing algorithm, or reserving some bandwidth for low priority data packets

Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

Switching strategies

- **Switching** establishes the type of connection between source and destination. It is tightly coupled to **routing**. **Can be seen as a flow control mechanism, as a problem of resource allocation.**
- Allocation of network resources (bandwidth, buffer capacity, etc.) to information flows
  - Phit is a unit of data that is transferred on a link in a single cycle
  - Typically, phit size = flit size
- Two main switching schemes:
  1. **Circuit (or "path") switching**
  2. **Packet switching**

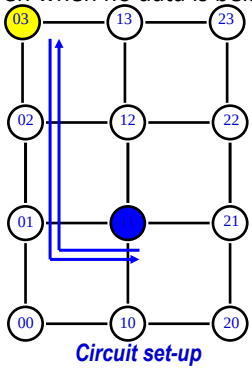


35

35

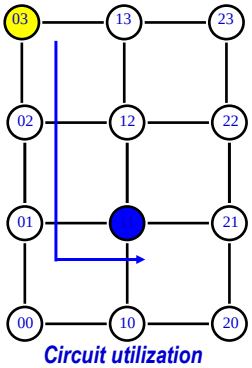
1. Pure Circuit Switching

- It is a form of bufferless flow control
- Advantage: Easier to make latency guarantees (after circuit reservation)
- Disadvantage: does not scale well with NoC size
  - several links are occupied for the duration of the transmitted data, even when no data is being transmitted



Circuit set-up

Two traversals – latency overhead
Waste of bandwidth
Request packet can be buffered



Circuit utilization

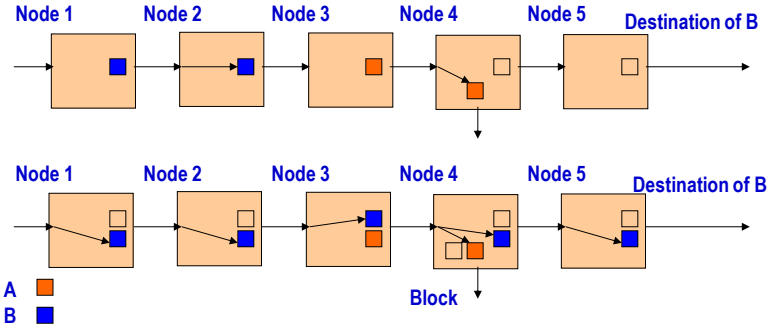
Third traversal – latency overhead
Contention-free transmission
Poor resource utilization

36

36

Virtual Circuit Switching

- Multiple virtual circuits (channels) multiplexed on a single physical link.
- Virtual-channel flow control decouples the allocation of channel state from channel bandwidth.
- Allocate one buffer per virtual link
  - can be expensive due to the large number of shared buffers
- Allocate one buffer per physical link
  - uses time division multiplexing (TDM) to statically schedule usage
  - less expensive routers



37

37

2. Packet Switching

- It is a form of buffered flow control
- Packets are transmitted from source and make their way independently to receiver
  - possibly along different routes and with different delays
- Zero start up time, followed by a variable delay due to contention in routers along packet path
  - QoS guarantees are harder to make

38

38

Three main packet switching scheme variants

1. Store and Forward (SAF) switching

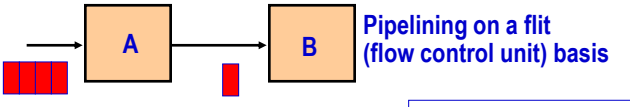
- packet is sent from one router to the next only if the receiving router has buffer space for entire packet
- buffer size in the router is at least equal to the size of a packet
- Disadvantage: excessive buffer requirements

2. Virtual Cut Through (VCT) switching

- forwards first flit of a packet as soon as space for the entire packet is available in the next router
- reduces router latency over SAF switching
- same buffering requirements as SAF switching

3. Wormhole (WH) switching

- flit is forwarded to receiving router if space exists for that flit



Pipelining on a flit (flow control unit) basis

(1) After A receives a flit of the packet, A asks B if B is ready to receive a flit
(2) B → A, ack
(3) A sends a flit to B.

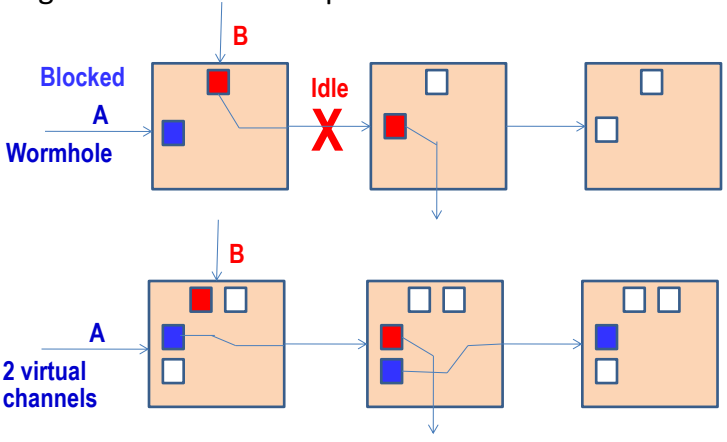
flit size < packet size
Smaller data space is needed than store-and-forward

39

39

Wormhole Switching Issues

- Wormhole switching suffers from packet blocking problems
- An idle channel cannot be used because it is owned by a blocked packet…
  - Although another packet could use it!
- Using virtual channels helps address this



40

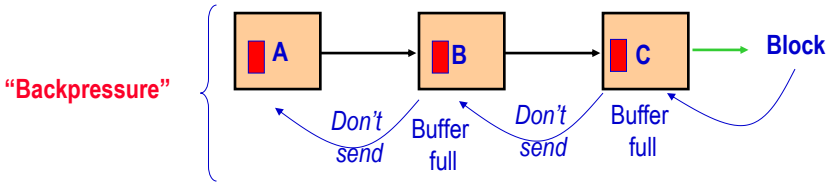
40

Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

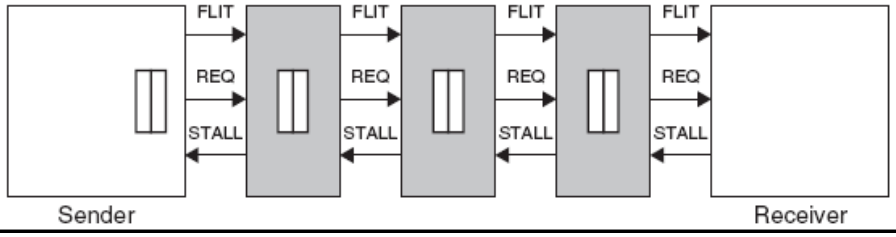
Flow control

- **Flow control** dictates which messages get access to particular network resources over time. It manages the allocation of resources to packets as they progress along their route. "It controls the traffic lights: when a car can advance or when it must pull off into a parking lot to allow other cars to pass".
- **Can be viewed as either a problem of resource allocation (switching strategy) or/and one of contention resolution.**
- Recover from transmission errors
- Commonly used schemes:
  - STALL-GO flow control
  - ACK-NACK flow control
  - Credit based flow control



STALL/GO

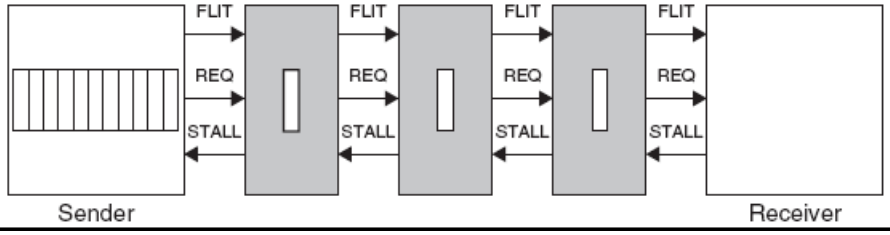
- low overhead scheme
- requires only two control wires
  - one going forward and signaling data availability
  - the other going backward and signaling either a condition of buffers filled (STALL) or of buffers free (GO)
- can be implemented with distributed buffering (pipelining) along link
- good performance – fast recovery from congestion
- does not have any provision for fault handling
  - higher level protocols responsible for handling flit interruption



43

ACK/NACK

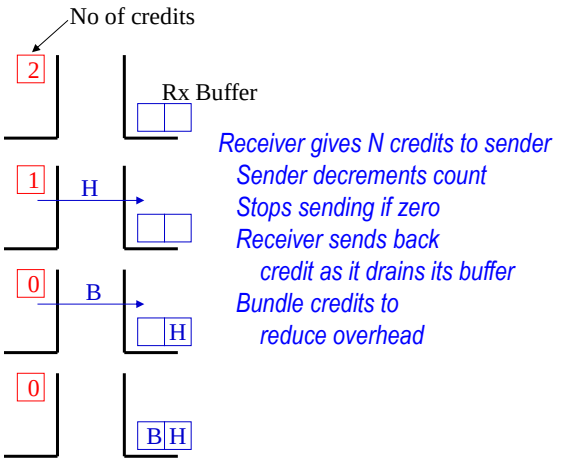
- when flits are sent on a link, a local copy is kept in a buffer by sender
- when ACK received by sender, it deletes copy of flit from its local buffer
- when NACK is received, sender rewinds its output queue and starts resending flits, starting from the corrupted one
- implemented either end-to-end or switch-to-switch
- sender needs to have a buffer of size 2N + k
  - N is number of buffers encountered between source and destination
  - k depends on latency of logic at the sender and receiver
- fault handling support comes at cost of greater power, area overhead



44

Credit Based

- Round trip time between buffer empty and flit arrival
- More efficient buffer usage; error control pushed at a higher layer



Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

Clocking schemes

- Fully synchronous
  - single global clock is distributed to synchronize entire chip
  - hard to achieve in practice, due to process variations and clock skew
- Mesochronous
  - local clocks are derived from a global clock
  - not sensitive to clock skew
  - phase between clock signals in different modules may differ
  - deterministic for regular topologies (e.g. mesh)
  - non-deterministic for irregular topologies
  - synchronizers needed between clock domains
- Pleisochronous
  - clock signals are produced locally
- Asynchronous
  - clocks do not have to be present at all

47

47

Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

48

48

Quality of Service (QoS)

- QoS refers to the level of commitment for packet delivery
  - refers to bounds on performance (bandwidth, delay, and jitter=packet delay variation)
- Two basic categories
  - Best effort (BE)
    - only correctness and completion of communication is guaranteed
    - usually packet switched
    - worst case times cannot be guaranteed
  - Guaranteed service (GS)
    - makes a tangible guarantee on performance, in addition to basic guarantees of correctness and completion for communication
    - usually (virtual) circuit switched

Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

Early Examples

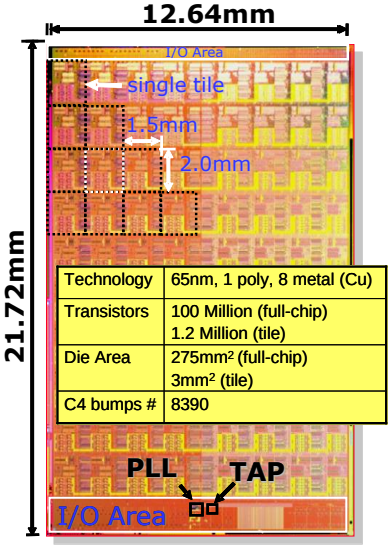
- Æthereal Developed by Philips
- HERMES Developed at the Faculdade de Informática PUCRS, Brazil
- MANGO Developed at the Technical University of Denmark
- Nostrum Developed at KTH in Stockholm
- Octagon Developed by STMicroelectronics
- QNoC Developed at Technion in Israel
- SOCBus Developed at Linköping University
- SPIN Micronetwork Université Pierre et Marie Curie, Paris, France
- Xpipes Developed by the Univ. of Bologna and Stanford University
- CHAIN (Silistix) Developed at the University of Manchester
- …

51

51

Intel's Teraflops Research Processor (2008)

- Goals:
- Deliver Tera-scale performance
  - Single precision TFLOP at desktop power
  - Frequency target 5GHz
  - Bi-section B/W order of Terabits/s
  - Link bandwidth in hundreds of GB/s
- Prototype two key technologies
  - On-die interconnect fabric
  - 3D stacked memory
- Develop a scalable design methodology
  - Tiled design approach
  - Mesochronous clocking
  - Power-aware capability



[Vangal08]

52

52

Intel (2018)

Loihi: A Neuromorphic Manycore Processor with On-Chip Learning

Mike Davies
Intel Labs, Intel Corporation

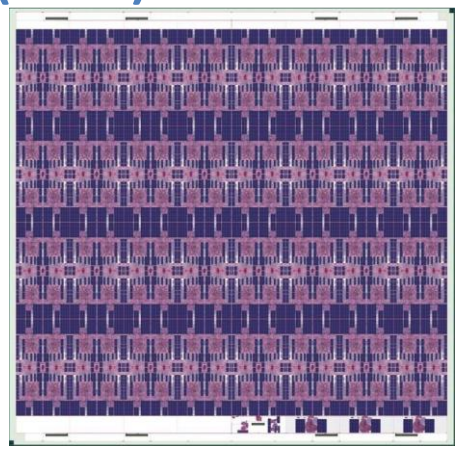
Narayan Srinivasa
Eta Compute

Tsung-Han Lin
Gautham Chinya
Yongqiang Cao
Sri Harsha Choday
Intel Labs, Intel Corporation

Georgios Dimou
Reduced Energy
Microsystems

Prasad Joshi
Nabil Imam
Shweta Jain
Yuyun Liao
Chit-Kwan Lin
Andrew Lines
Ruokun Liu

Loihi is a 60-mm² chip fabricated in Intel's 14-nm process that advances the state-of-the-art modeling of spiking neural networks in silicon. It integrates a wide range of novel features for the field, such as hierarchical connectivity, dendritic compartments, synaptic delays, and, most importantly, programmable synaptic learning rules. Running a spiking convolutional form of the Locally Competitive Algorithm, Loihi can solve LASSO optimization problems with over three orders of magnitude superior energy-delay product compared to conventional solvers running on a CPU iso-process/voltage/area. This provides an unambiguous example of spike-based computation, outperforming all known conventional solutions.



- Loihi was fabbed in Intel's 14-nm FinFET process. The chip instantiates a total of 2.07 billion transistors and 33 MB of SRAM over its 128 neuromorphic cores and three x86 cores, with a die area of 60 mm2. The device is functional over a supply voltage range of 0.50 V to 1.25 V.
- An asynchronous network-on-chip (NoC) transports all communication between cores in the form of packetized messages.

53

53

IBM (2019)

TrueNorth: Accelerating From Zero to 64 Million Neurons in 10 Years

Michael V. DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P. Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K. Nayak, Rathinakumar Appuswamy, Peter J. Carlson, Andrew S. Cassidy, Pallab Datta, Steven K. Esser, Guillaume J. Garreau, Kevin L. Holland, Scott Lekuch, Michael Mastro, Jeff McKinstry, Carmelo di Nolfo, Brent Paulovicks, Jun Sawada, Kai Schleupen, Benjamin G. Shaw, Jennifer L. Klamo, Myron D. Flickner, John V. Arthur, and Dharmendra S. Modha, IBM Research

IBM's brain-inspired processor is a massively parallel neural network inference engine containing 1 million spiking neurons and 256 million low-precision synapses. Now, after a decade of fundamental research spanning neuroscience, architecture, chips, systems, software, and algorithms, IBM has delivered the largest neurosynaptic computer ever built.

FIGURE 3. The 64-processor scale-out NS16e-4.

- TrueNorth, with 1 million neurons and 256 million synapses distributed across 4,096 neurosynaptic cores and fabricated in Samsung's 28-nm low-power process, occupies 430 mm2 of area and consumes on the order of 100 mW of power during a typical use case.
- NS16e-4 is composed of four component NS16e systems arranged within a 4-U rack-mounted standard drawer. Each NS16e platform contains 16 TrueNorth processors tiled in a 4 × 4 two-dimensional grid and connected to one another via TrueNorth's native interchip input–output (I/O) interfaces, forming a unified array of 262, 144 neurosynaptic cores totaling 64 million neurons and 16 billion synapses.
- Cores in TrueNorth are interconnected via a network on chip.

54

54

Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

NoC prototyping: EPFL Emulation Framework

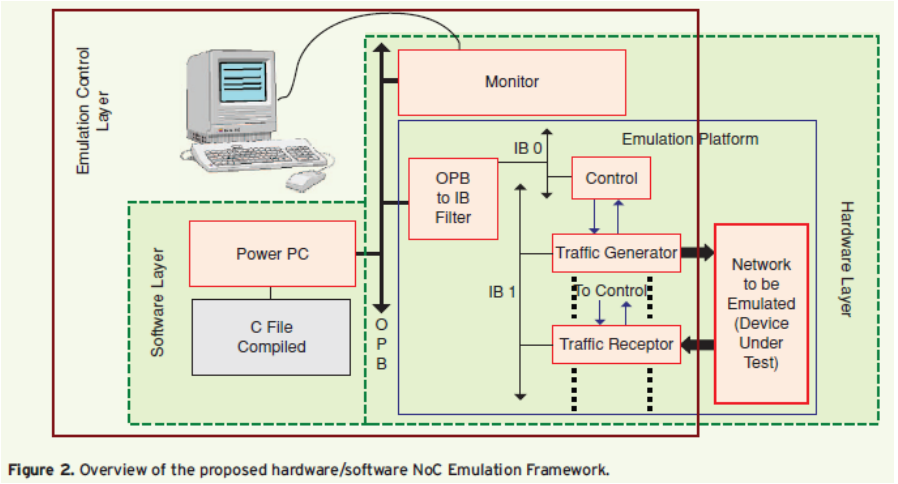
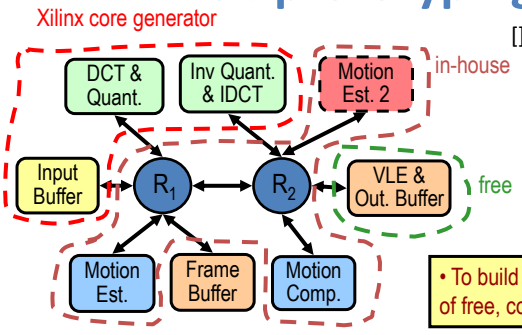


Figure 2. Overview of the proposed hardware/software NoC Emulation Framework.

[] N, Genko, D. Atienza, G. De Micheli, L. Benini, "Feature-NoC emulation: a tool and design flow for MPSoC," IEEE Circuits and Systems Magazine, vol. 7, pp. 42-51, 2007.

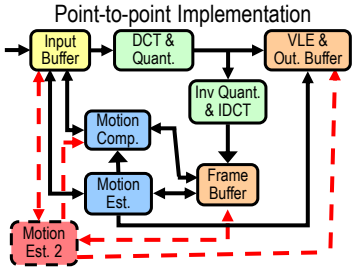
NoC prototyping: CMU

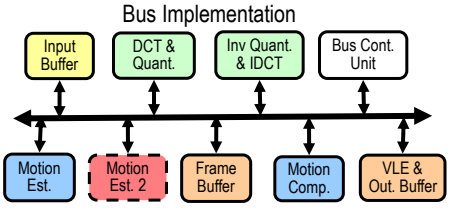


[] Umit Y. Ogras, Radu Marculescu, Hyung Gyu Lee, Puru Choudhary, Diana Marculescu, Michael Kaufman, Peter Nelson, "Challenges and Promising Results in NoC Prototyping Using FPGAs," IEEE Micro, vol. 27, no. 5, pp. 86-95, 2007.

• To build prototypes, we will likely use a mix of free, commercial, and in-house IPs.

Synthesis for Xilinx Virtex II FPGA with CIF (352x288) frames

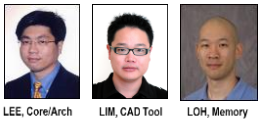




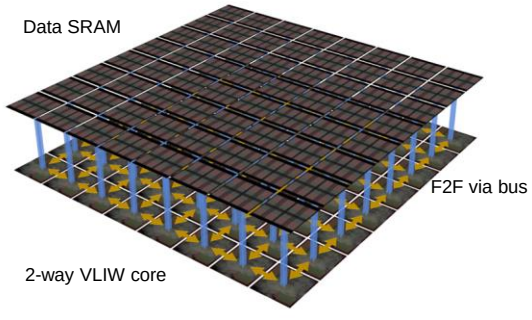
57

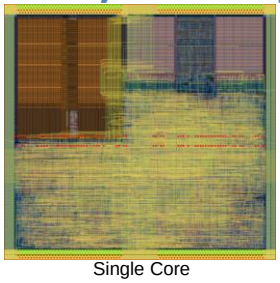
57

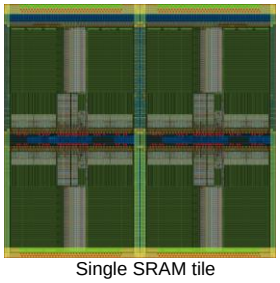
Georgia Tech 64-Core 3D-MAPS Many-Core Chip



- 3D-stacked many-core processor
- Fast, high-density face-to-face vias for high bandwidth
- Wafer-to-wafer bonding
- @277MHz, peak data B/W ~ 70.9GB/sec




Single Core


Single SRAM tile

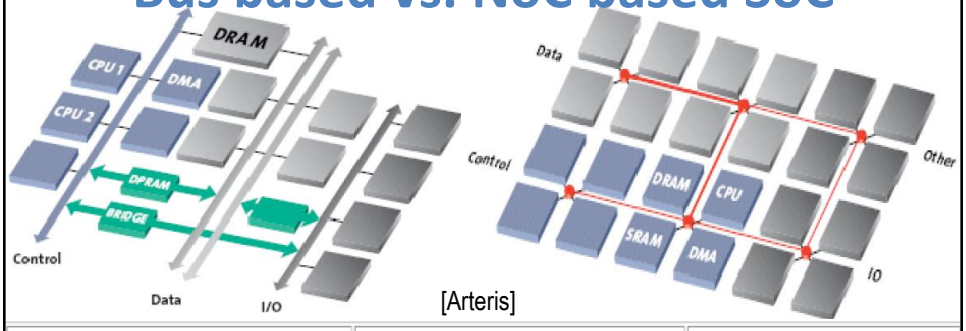
58

58

Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

Bus based vs. NoC based SoC



[Arteris]

Criteria	Bus	NoC
Max Frequency	250 MHz	> 750 MHz
Peak Throughput	9 GB/s (more if wider bus)	100 GB/s
Cluster min latency	6 Cycles @250MHz	6 Cycles @250MHz
Inter-cluster min latency	14-18 Cycles @250MHz	12 Cycles @250MHz
System Throughput	5 GB/s (more if wider bus)	100 GB/s
Average arbitration latency	42 Cycles @250MHz	2 Cycles @250MHz
Gate count	400K	210K
Dynamic Power	Smaller for NoC, see discussion in 3.5.2	
Static Power	Smaller for NoC (proportional to gate count)	

Outline

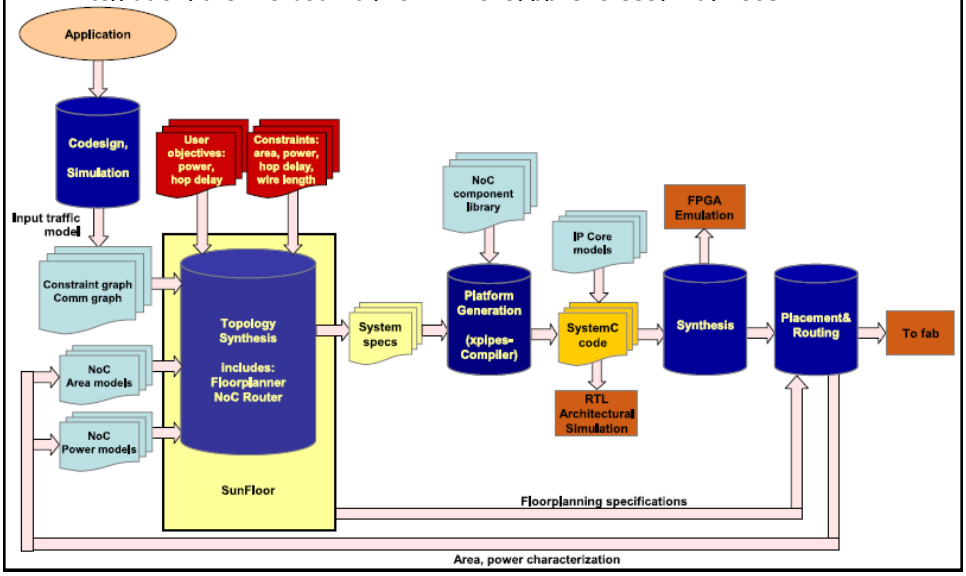
- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

61

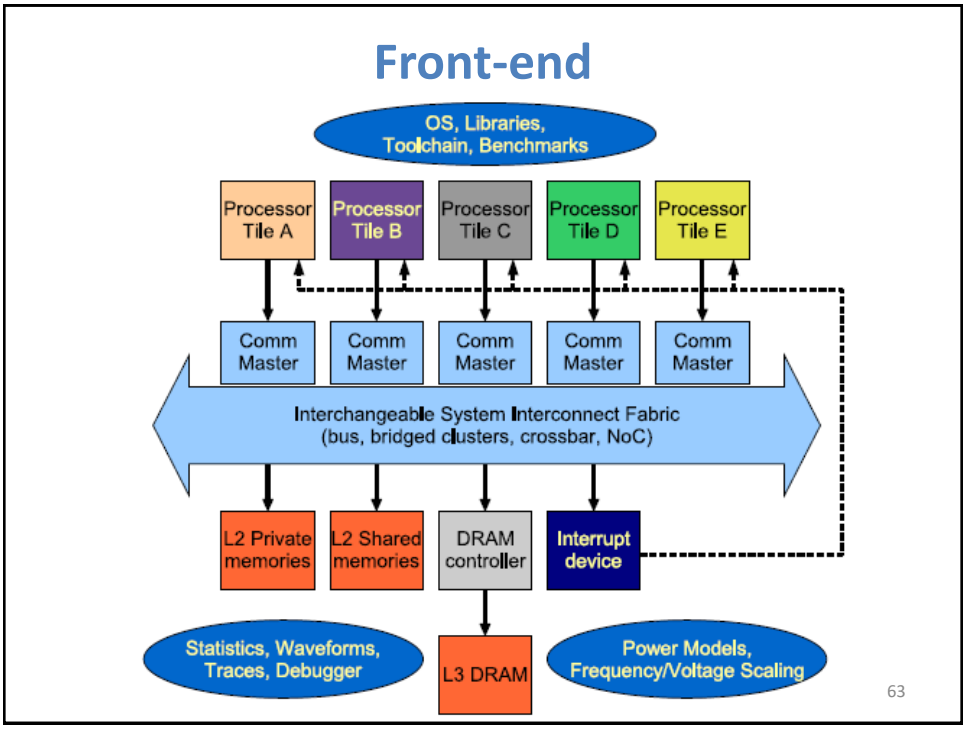
61

Example: Sunflower Design flow

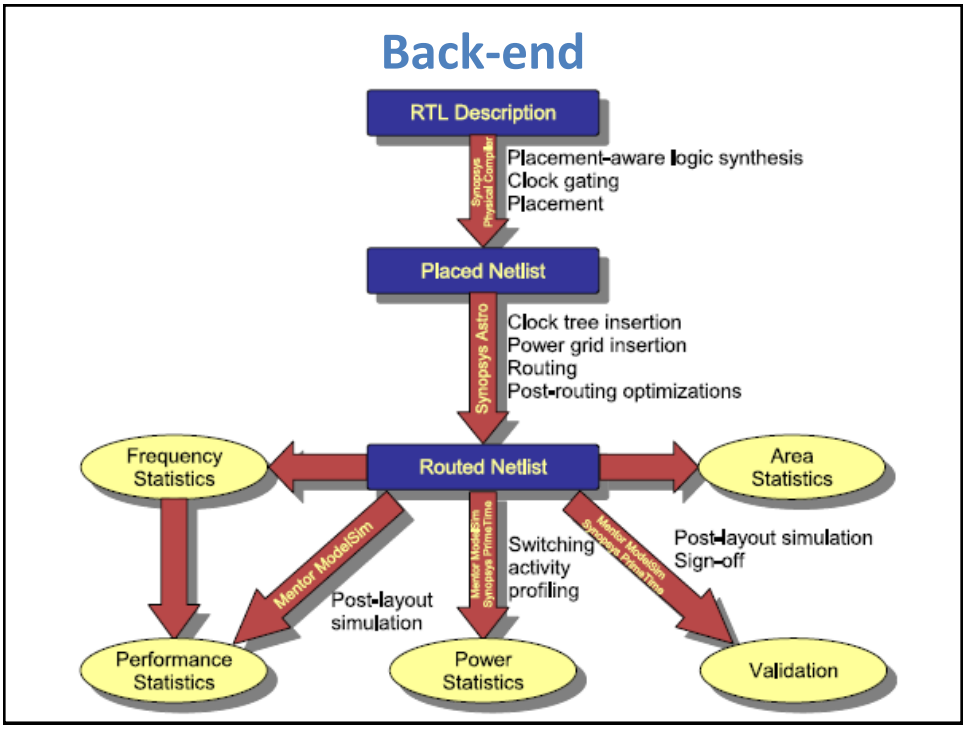
- David Atienza, Federico Angiolini, Srinivasan Murali, Antonio Pullini, Luca Benini, Giovanni De Micheli, "Network-on-Chip design and synthesis outlook," Integration, the VLSI Journal, vol. 41 no. 3, pp. 340-359, May 2008.



62


Front-end
OS, Libraries, Toolchain, Benchmarks
Processor Tile A
Processor Tile B
Processor Tile C
Processor Tile D
Processor Tile E
Comm Master
Comm Master
Comm Master
Comm Master
Comm Master
Interchangeable System Interconnect Fabric (bus, bridged clusters, crossbar, NoC)
L2 Private memories
L2 Shared memories
DRAM controller
Interrupt device
Statistics, Waveforms, Traces, Debugger
L3 DRAM
Power Models, Frequency/Voltage Scaling
63

63

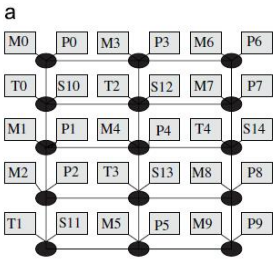

Back-end
RTL Description
Synopsys Physical Compiler
Placement-aware logic synthesis
Clock gating
Placement
Placed Netlist
Synopsys Astro
Clock tree insertion
Power grid insertion
Routing
Post-routing optimizations
Frequency Statistics
Routed Netlist
Area Statistics
Mentor ModelSim
Post-layout simulation
Mentor ModelSim Synopsys PrimeTime
Switching activity profiling
Mentor ModelSim Synopsys PrimeTime
Post-layout simulation Sign-off
Performance Statistics
Power Statistics
Validation

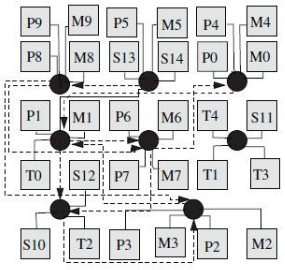
64

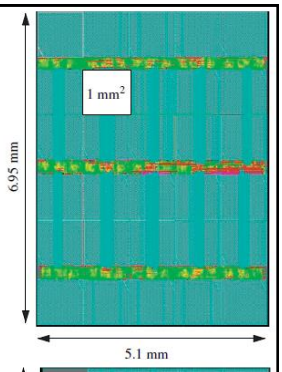
Manual vs. Design tool

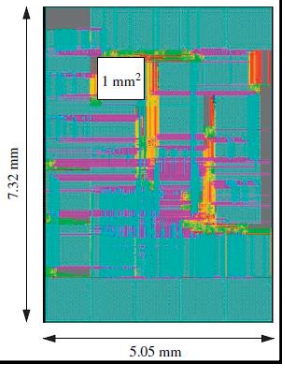
Manual

Sunflower
- 1.33x less power
- 4.3% area increase









65

Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

66

66

Status and Open Problems

- Design tools (GALS, DVFS, VFI) and benchmarks. HW/SW co-design
- Power
  - complex NI and switching/routing logic blocks are power hungry
  - several times greater than for current bus-based approaches
- Latency
  - additional delay to packetize/de-packetize data at NIs
  - flow/congestion control and fault tolerance protocol overheads
  - delays at the numerous switching stages encountered by packets
  - even circuit switching has overhead (e.g. SOCBUS)
  - lags behind what can be achieved with bus-based/dedicated wiring
- Reliability
  - Wearout mechanisms (electromigration, NBTI, etc.)
  - Reliable/robust NoC designs
  - Error correcting codes
- Security
  - Attacks (side channel attacks, etc.)
- Simulation speed
  - GHz clock frequencies, large network complexity, greater number of PEs slow down simulation
  - FPGA accelerators
- Standardization → we gain:
  - Reuse of IPs
  - Reuse of verification
  - Separation of Physical design issues, Communication design, Component design, Verification, System design
- Prototyping

Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

Trends

- Hybrid and hierarchical interconnection structures
  - NoC and Bus based
  - Custom (application specific), heterogeneous topologies
- New interconnect paradigms
  - Optical/photonic, Wireless (for latency, power)
- 3D NoC (for latency)
- Reconfigurability features (for robustness/reliability)
- GALS, DVFS, VFI (for power consumption)

69

69

Wireless NoC

- A. Sarihi, A. Patooghy, A. Khalid, M. Hasanzadeh, M. Said and A. -H. A. Badawy, "A Survey on the Security of Wired, Wireless, and 3D Network-on-Chips," in *IEEE Access*, vol. 9, pp. 107625-107656, 2021, doi: 10.1109/ACCESS.2021.3100540.

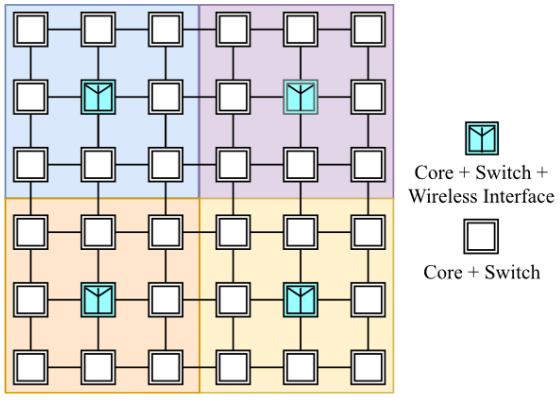


FIGURE 3. A simple example of a WiNoC with four clusters.

70

70

Photonic NoC

- T. Alexoudi *et al*., "Optics in Computing: From Photonic Network-on-Chip to Chip-to-Chip Interconnects and Disintegrated Architectures," in *Journal of Lightwave Technology*, vol. 37, no. 2, pp. 363-379, 15 Jan.15, 2019, doi: 10.1109/JLT.2018.2875995.

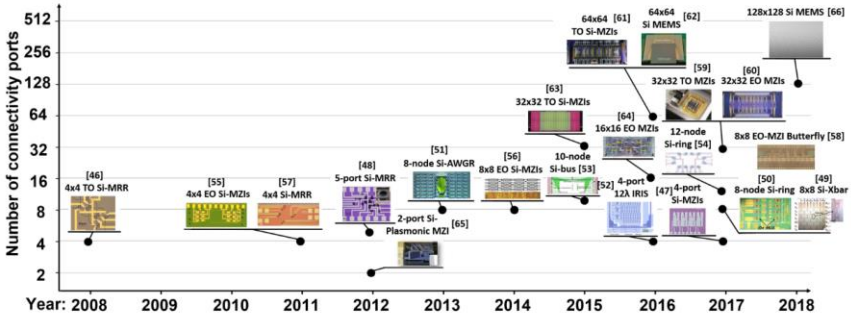


Fig. 3. Evolution of photonic Network-on-Chip and on-chip photonic switches.

Reliable/Robust NoC

- K. Wang and A. Louri, "CURE: A High-Performance, Low-Power, and Reliable Network-on-Chip Design Using Reinforcement Learning," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2125-2138, 1 Sept. 2020, doi: 10.1109/TPDS.2020.2986297.

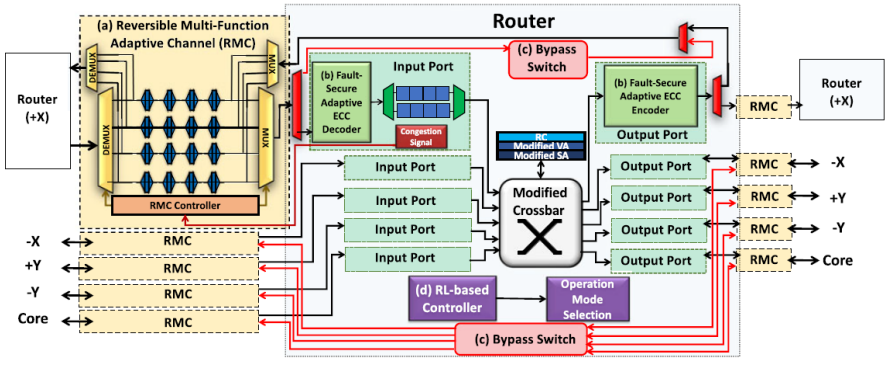


Fig. 1. CURE architecture design. CURE consists of (a) a reversible multi-function adaptive channel (RMC) between adjacent routers, (b) a new fault-tolerant router design with modified VA & SA & ST (crossbar) and adaptive error detection/correction hardware, (c) a router bypass route, and (d) a reinforcement learning (DRL)-based control policy.

Secure NoC

- A. Sarihi, A. Patooghy, A. Khalid, M. Hasanzadeh, M. Said and A. -H. A. Badawy, "A Survey on the Security of Wired, Wireless, and 3D Network-on-Chips," in IEEE Access, vol. 9, pp. 107625-107656, 2021, doi: 10.1109/ACCESS.2021.3100540.

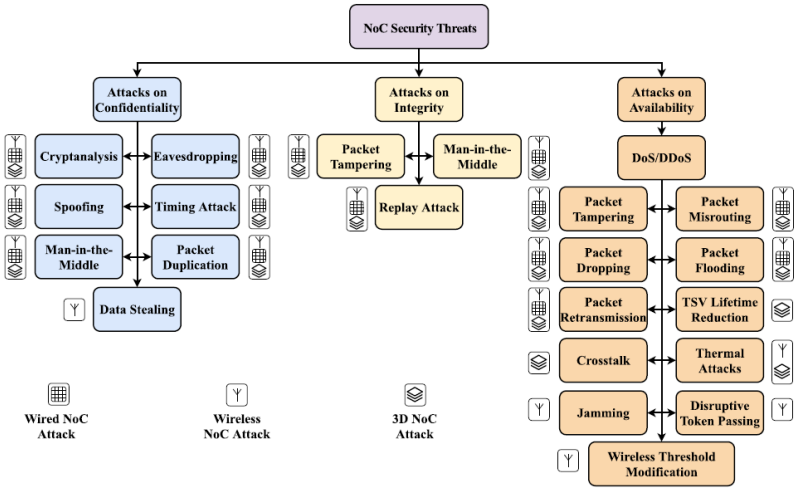


FIGURE 5. Classification of the addressed attacks based on the attacker's target security goal.

73

73

Outline

- Introduction
- NoC Topology
- Routing algorithms
- Switching strategies
- Flow control schemes
- Clocking schemes
- QoS
- NoC Architecture Examples
- NoC prototyping
- Bus based vs. NoC based SoC
- Design flow/methodology
- Status and Open Problems
- Trends
- Companies, simulators

74

74

Companies, Simulators

- For info on NoC related companies, simulators, other tools, conference pointers, etc. please see:
  - http://networkonchip.wordpress.com/

75

75

Summary

- NoC paradigm
  - New communication infrastructure for chip multiprocessors and Systems-on-Chip
  - Replaces the Bus for increasingly large numbers of cores/PEs
- Automated design flow/methodology
  - One main challenges

76

76

References/Credits

- IEEE Xplore
- Google Scholar
- http://www.diit.unict.it/users/mpalesi/DOWNLOAD/noc_research_summary-unlv.pdf

77