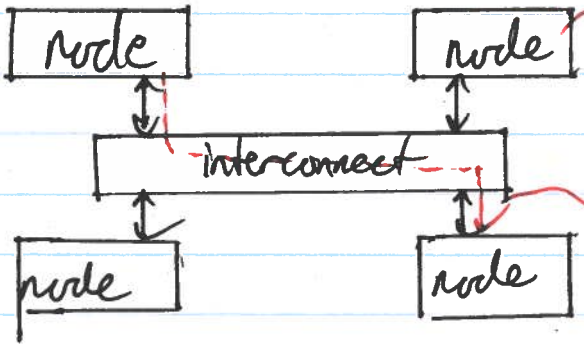
On-Chip Interconnect



PE + I/O + memory
- shared address space
packets (or "flows" for 2+ packets)

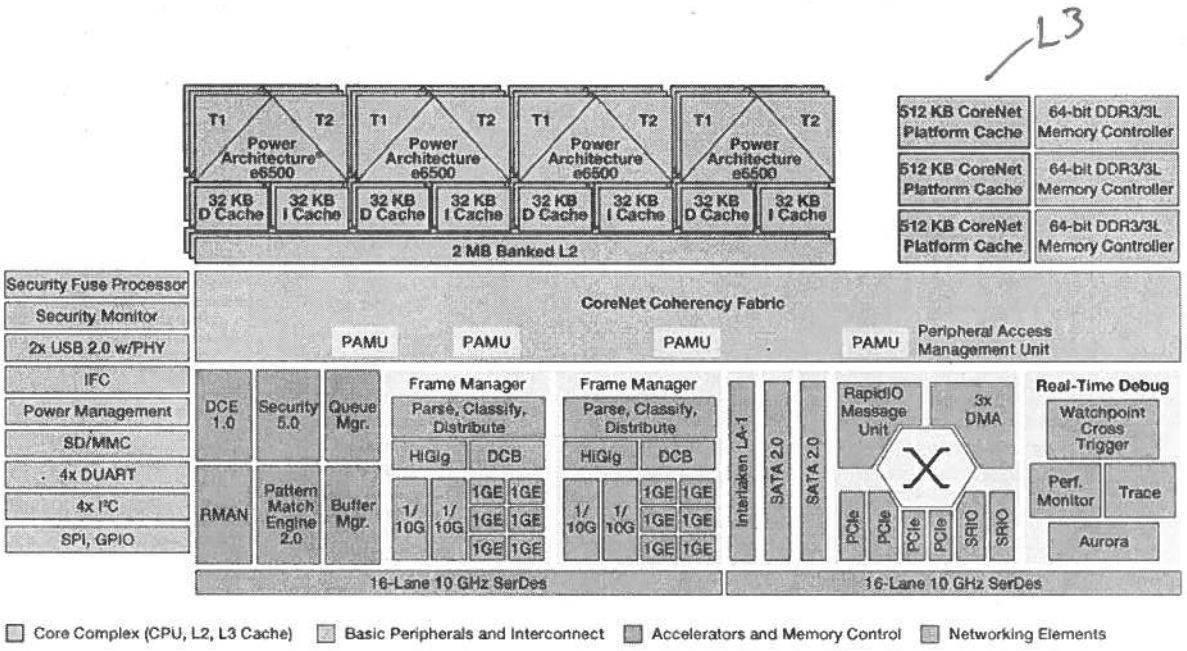
- PEs typically have L1 and L2 caches
- L3 cache may be with PE or with memory

NXP T4240 multicore processor
- Handouts > onchipConnect
- applications: aerospace, automation, mobile infrastructure
- 3 groups of multithreaded cores
- L1-I, L1-D per core
- 2 MiB L2 per group
- 3 DDR3 controllers with associated 512 MiB L3s

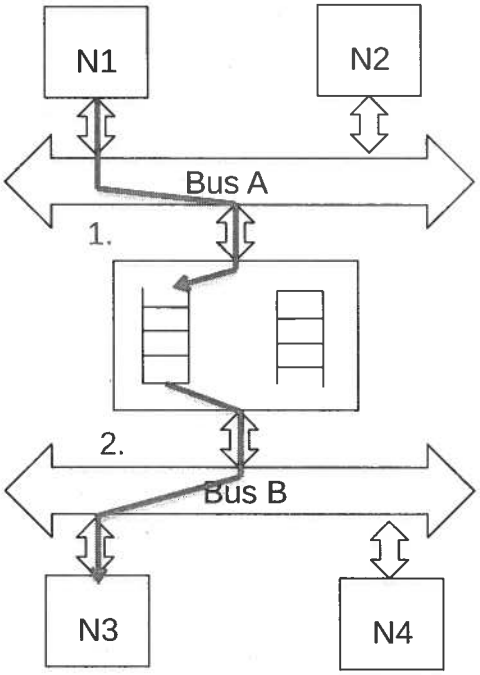
Communication Metrics

- throughput (bytes/s)
  - T^{iso} = max throughput per node in isolation (no conflicts)
  - T^{col} = max throughput per node collectively (conflicts)
- latency: Δ time for packet from source to destination
- energy: per bit to transfer a packet
- area: silicon used for connection

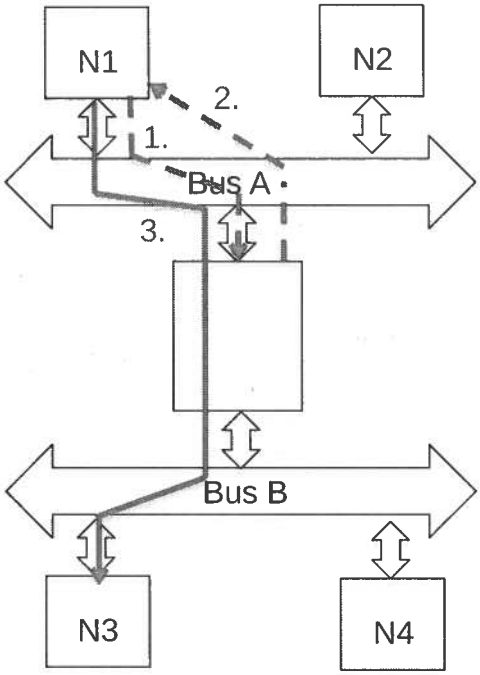
QorIQ T4220 Multi-core



Multi-Bus



store-and-forward



pass-through

ARM926EJ-S

- 32-bit Harvard architecture (2001)

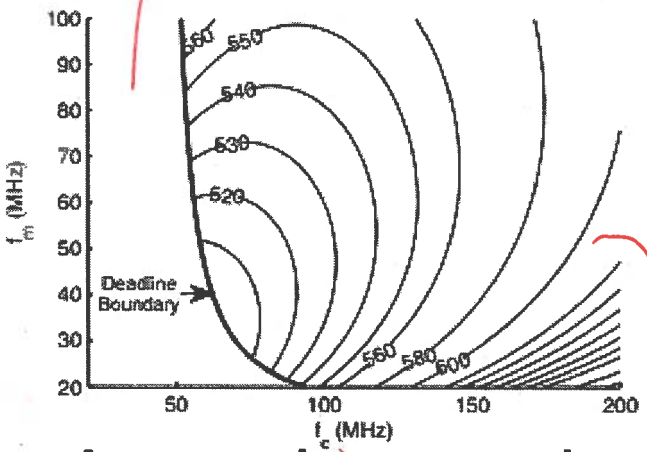
Capacitance (nF)				Power (mW)	
C_{ca}	C_{cs}	C_{ma}	C_{ms}	I	R
0.505	0.224	0.540	0.210	6.570	67.434

CPU clock	20 - 200 MHz (2 MHz step)	91 steps
Bus clock	20 - 100 MHz (f_c/n)	
Voltage	1.504 - 1.824 V (0.32 V step)	11 steps
L1 cache	8 KiB I, 8 KiB D	

and memory

0.032 V step

E vs f_c and f_m



core active cycles, mem active cycles

K = 1.4x10^8, M = 3.0x10^7, T = 3 s

* for a set of tasks, T = LCM of periods

energy

- Assuming continuous – use continuous optimization

nonlinear

- Assuming discrete – perform exhaustive search

Yun, et al System-wide Energy Optimization for Multiple DVS Components and Real-time Tasks, 2010

Notation

N = # nodes
L = packet length in bits
W = link width in bits
f = transmission frequency (bandwidth = Wf)
P_D = propagation delay

- we'll assume that arbitration can be performed in parallel with a prior transmission and hence no cost

On-Chip Inter-connect Types

1. Bus
2. Crossbar
3. MultiBus (with bridges)
4. Network-on-Chip (NoC)

Bus

- common connection between all nodes

latency $\Delta_{bus} = \frac{L}{fW} + P_0$

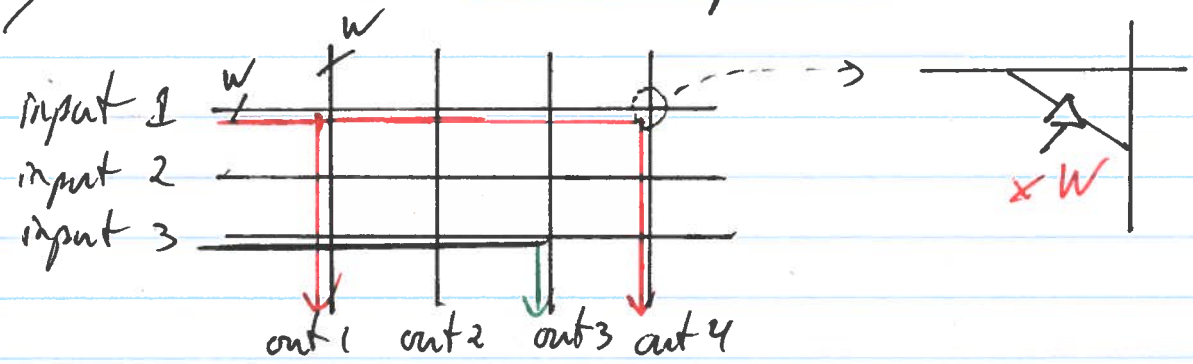
throughput in isolation $T_{bus}^{iso} = \frac{L}{\Delta_{bus}}$

throughput per node $T_{bus}^{col} = T_{bus}^{iso} / N$

- longer wires ⇒ higher capacitance ⇒ higher power and low speed
- lacks parallelism of transfers
- scales poorly: suitable for 4–8 nodes

Crossbar

- fully connected network of N inputs and M outputs



- requires $N \times M \times W$ drivers
- max 1 active driver per output line
- 0–M active drivers per input line

- assuming $N \leq M$

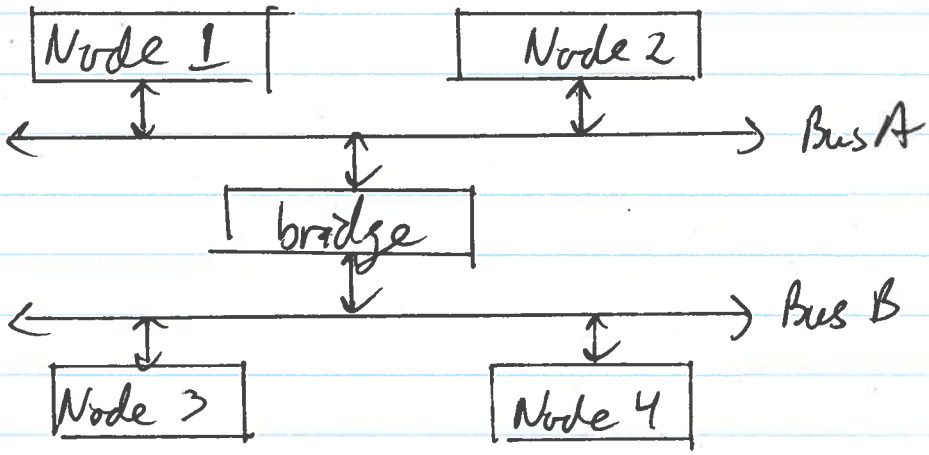
$$\Delta_{xbar} = \Delta_{bus}$$

$$T_{xbar}^{col} = T_{xbar}^{iso} = T_{node\ bus}^{iso}$$

- in theory, scales perfectly
- wire capacitance $\propto M$
- area scales with $N \times M$
- typically scales to 8×8

MultiBus

- can scale beyond buses and crossbars



- K = # buses, $T^{col}_{Kbus} = K\,T^{col}_{bus}$ (when packets stay local to their bus)

Bridge Types

a) store-and-forward
b) pass-option

Store-and-Forward (handout - page 2)

- a buffer in each direction
- example operation: send packet from N1 to N3
  1. N1 contends for Bus A, wins arbitration, sends packet to the bridge's A→B buffer
  2. the bridge contends for Bus B, wins arbitration, sends packet to N3

- latency: $\Delta_{2\,bus} = 2\,\Delta_{bus}$ (assuming low bridge latency)

Pass-Through (hardcut)

- no buffering
- example operation: N1 → N3
  1. N1 contends for Bus A, wins arbitration, notifies bridge
  2. bridge contends for Bus B, wins arbitration, acknowledges N1
  3. N1 sends directly to N3 (through bridge)
- latency: $\Delta_{2\,bus} = \Delta_{bus} + 3 \times P_D$
  ↑ $(L/fw + P_D)$
- can deadlock: example N1 → N3 and Node 4 → N2
  1. N1 wins Bus A while N4 wins Bus B
  2. both notify the bridge simultaneously
  - can be solved with priorities and NAck from bridge → N4 (abort and retry)

MultiBus Example: Phillips Nexperia (handout - page 3)

- multimedia platform e.g. digital TV set-top box

- 2 processors (general, media), accelerators
- 4 ~~buses~~ buses:
  - MIPS fast proc. bus
  - MIPS slow peripheral bus
  - TriMedia proc. bus
  - Memory bus (high bw)

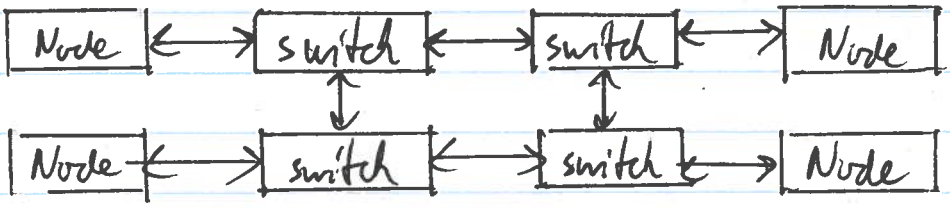
  (MIPS fast proc. bus — bridge — MIPS slow peripheral bus; TriMedia proc. bus — bridge — Memory bus (high bw); the whole group — bridge)

- address space is partitioned among the buses
- routing is built into the bridges
  - e.g. packet TriMedia → UART
    - option 1: TM bus — MIPS slow bus
    - option 2: TM bus — Mem bus — Fast bus — Slow bus
- pass-through bridges only work of the buses share speed, width, protocol; otherwise use store-and-forward

Network-on-Chip

typically crossbars

- network of switches connected by links
- one node per chip
- full-duplex links most common



- good scalability, but requires multiple hops (latency)

Links

- short ⇒ low capacitance ⇒ low power and high speed (~GHz)
- propagation delay P0 per hop: 1/f (1 cycle)

Issues

a) flow-control
b) routing
c) topologies

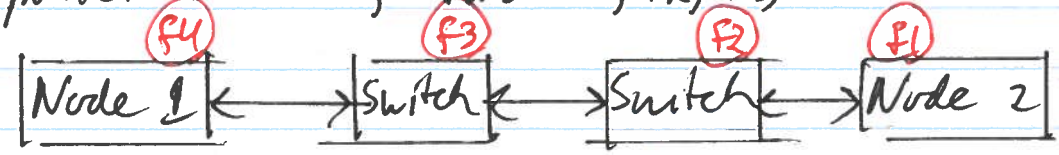
Differences with Off-Chip Networks

- synchronization is easier (common clock, short wires)
- wide links (vs PCIe, Ethernet, SATA - serial)
- buffers are relatively costly (area, power)

Wormhole Switching

- design goal: minimize buffering
- packets are subdivided into flits (flow control unit) (size = link width) that can be sent in 1 cycle
- switches buffer 1 flit
- packet "snakes" through the network

e.g. packet N1 → N2, flits f1, f2, f3, f4



hops, H = 3 # flits, L/W = 4

$$\text{latency} = H + L/W - 1$$
$$= 3 + 4 - 1 = 6$$

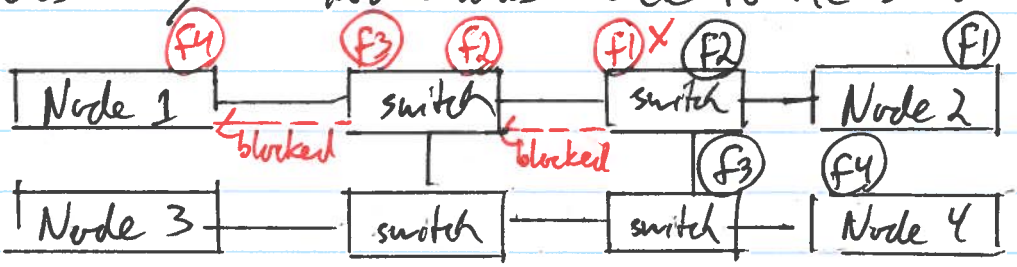
$$\Delta_{worm} = \frac{H + L/W - 1}{f}, \quad T_{worm}^{iso/col} = fW$$

Flow Control

- handling contention at switches: reduce blocking / maximize throughput
- there is no preemption of packet ~~flits~~ flits: once the route is established all flits follow uninterrupted

(1) Back-Pressure

- switches relay blocked status back to the sender



- flexible but requires additional buffering

(2) Virtual Circuit

- send an initial flit to reserve switch ports from source to destination
- after ACK is received at the sender, it sends the packet, a flit per cycle
- adds $2 \times H$ latency, so it only works well for long packets ($L/W >> H$) other it results in low throughput
  - L = message length
  - W = link width
  - (L/W = # flits)

(3) Offline Scheduling

- build a contention-free schedule offline
- nodes and switches follow the schedule table (~ like TDMA over the whole NoC)
- simplifies switches and is deterministic (good for real-time) but is inflexible

Routing

- goals: minimize contention, maximize bandwidth
- static: e.g. x-y routing
  - first route horizontally, then vertically
  - prevents deadlock
- dynamic: e.g. back-pressure routing
  - congested switches inform their neighbours which route packets away

Topologies

a) Mesh
b) Torus
c) Ring

Mesh Topology

- ~~Hacks~~ Handouts > on chip Connect : mesh & toras diagrams
- for a square mesh

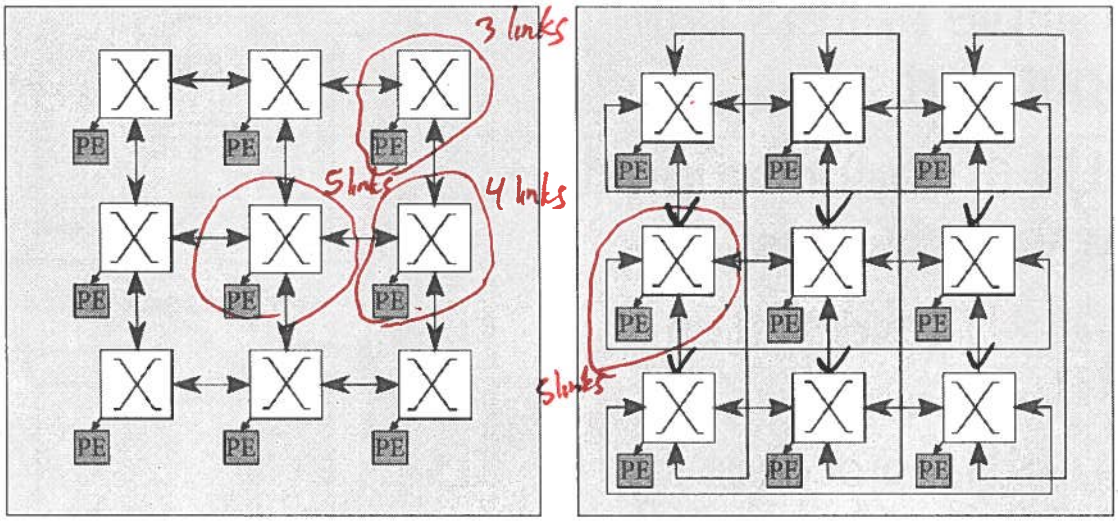
$$\#\text{ links} = \underbrace{N}_{\text{to each PE}} + \underbrace{\sqrt{N}(\sqrt{N}-1)}_{\#\text{rows} \quad \#\text{cols}} + \underbrace{\sqrt{N}(\sqrt{N}-1)}_{\#\text{cols} \quad \#\text{rows}}$$

maximal shortest route, $H_{max} = 2\sqrt{N}$ ($\sqrt{N}-1$ horz + $\sqrt{N}-1$ vert + to the PEs)

(sub-linear scaling)

through per node $(T^{col}_{worm}) = T^{iso}_{worm}$ only if nodes communicate to neights

Topologies

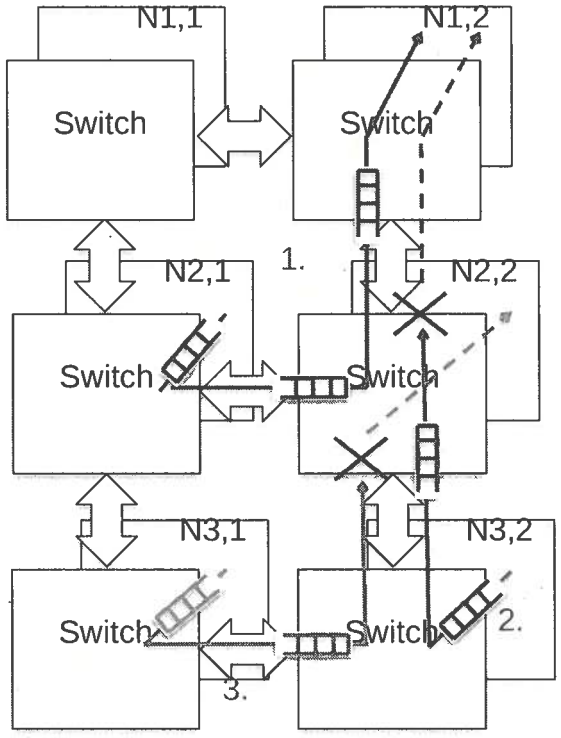


(a) Mesh (b) Torus

Vipin, "AsyncBTree: Revisiting Binary Tree Topology for Efficient FPGA-Based NoC Implementation," *International Journal of Reconfigurable Computing*, 2019.

Blocking

- single queue per input port
- $N_{2,1}$ -> $N_{1,2}$ is being transmitted (in/out ports reserved until packet done)
- $N_{3,2}$ -> $N_{1,2}$ blocks
- $N_{3,1}$ -> $N_{2,2}$ blocks because packet from $N_{3,2}$ is waiting for link to $N_{2,2}$



Torus Topology

- for square torus

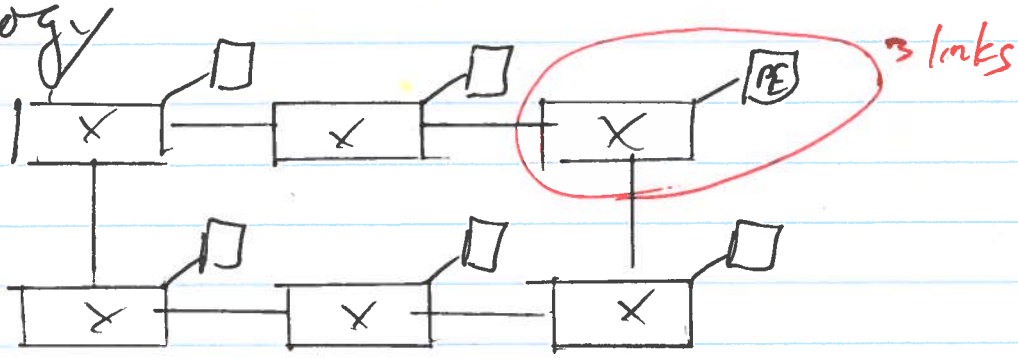
$$\#\text{links} = 3N = \left(1 + 4 \times \tfrac{1}{2}\right) \times N$$

(1 to each PE; 4 × ½ to each PE)

$$H_{max} = \underset{\text{from PE}}{1} + \underbrace{\lfloor \sqrt{N}/2 \rfloor}_{\text{horizontally}} + \underbrace{\lfloor \sqrt{N}/2 \rfloor}_{\text{vert.}} + \underset{\text{to PE}}{1}$$

- disadvantage: longer wires to connect end nodes/switches

Ring Topology



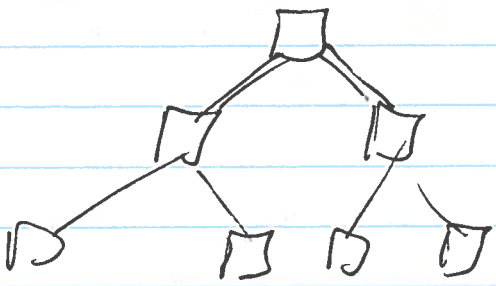
- route: choose shortest path: clockwise or counter

$$H_{max} = \lfloor N/2 \rfloor + 2$$

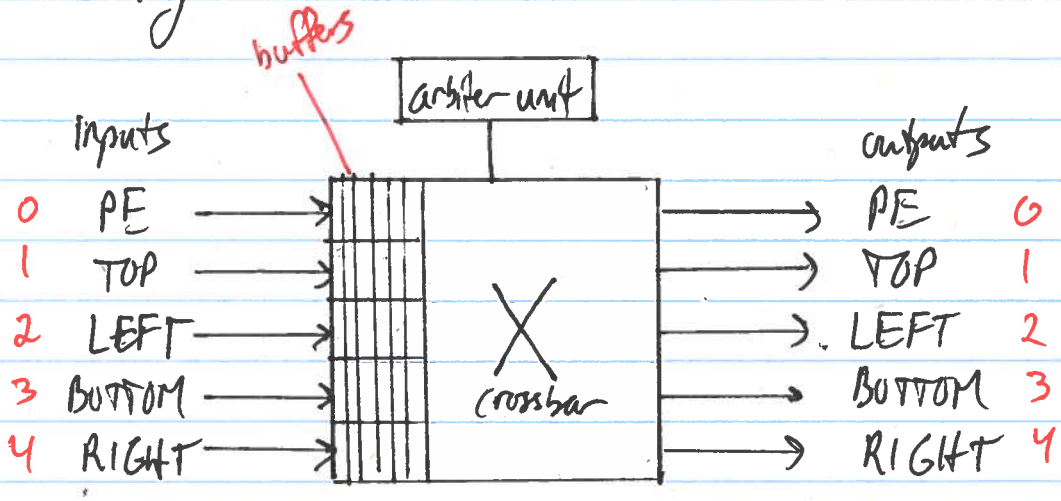
(+2: to/from PE)

$$\#\text{links} = 2N = \left(1 + 2 \times \tfrac{1}{2}\right) \times N$$

Other Topologies: linear (bus), tree, star



Switch Design

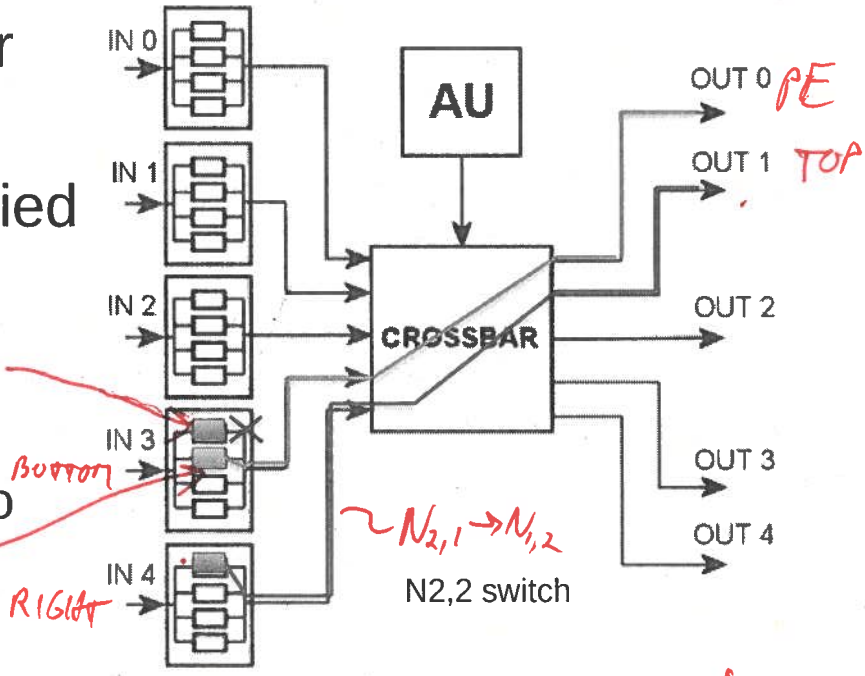


Virtual Channels

- see handout : Blocking
  - packets are blocked unnecessarily by other blocked packets
- increase total throughput with virtual channels
  - each input port has multiple queues/buffers
  - a packet is assigned to a virtual channel
  - see handout: Virtual Channels
- stochastic model showed that throughput increased level off around 4-8 virtual channels / input (see handout)

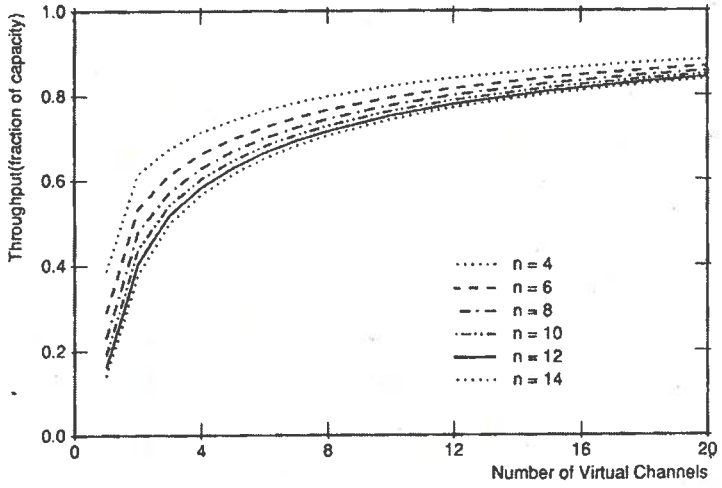
Virtual Channels

- Multiple buffers per input port
- OUT 1 (top) occupied by $N_{2,1}$ -> $N_{1,2}$ flits
- $N_{3,2}$ -> $N_{1,2}$ blocked on OUT 1
- $N_{3,1}$ -> $N_{2,2}$ proceeds to OUT 0 (node)



- same scenario as previous figure, but Packet 3 is not blocked

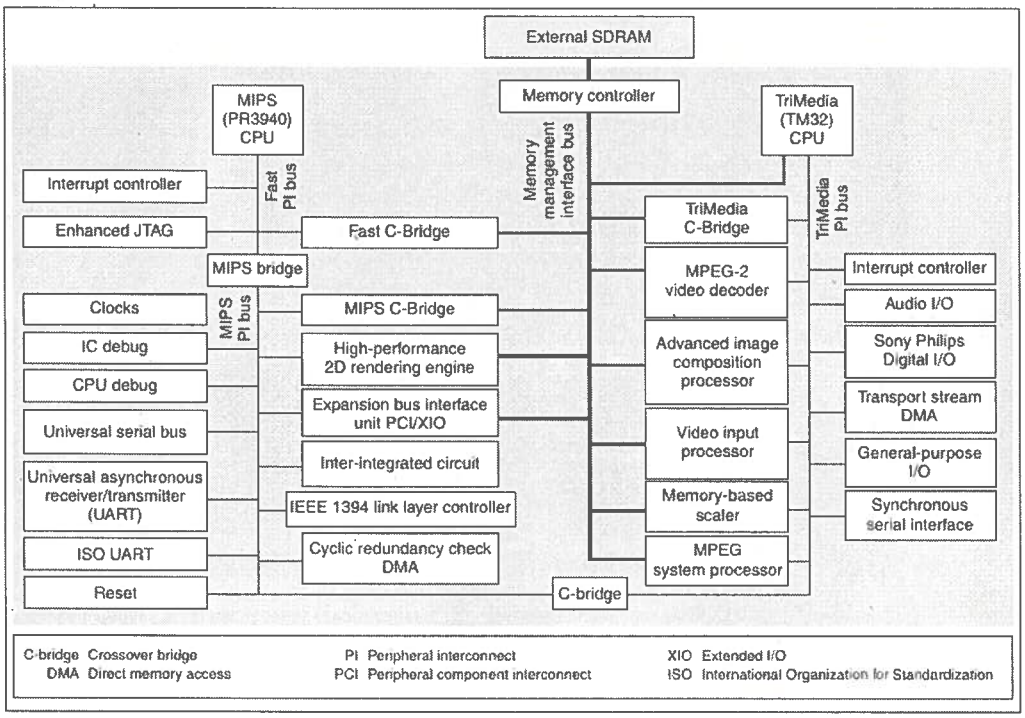
Stochastic Model: Throughput vs # Virtual Channels



- Assumptions: random source,destination for each packet
- Poisson arrival process on each node
- Fixed storage for 16 flits at each input port
- n is the number of hops

Dally, "Virtual-Channel Flow Control," *IEEE Transactions on Parallel and Distributed Systems*, 1992.

Phillips Nexperia

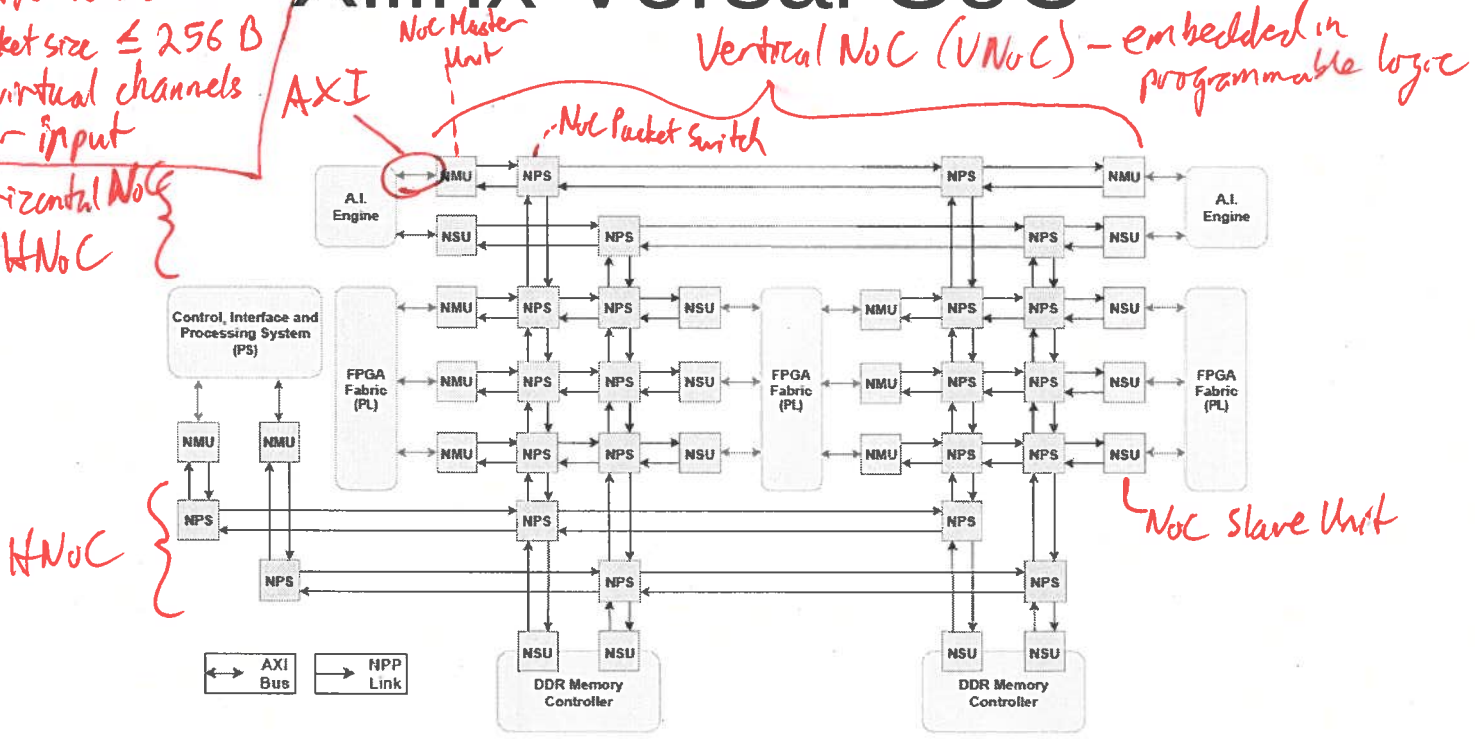


Dutta, Jensen, Rieckmann, "Viper: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems," *IEEE Design and Test of Computers*, 2001.

switches:
- wormhole routing
- 2 cycle latency
- buffer 16 128-bit flits
- packet size ≤ 256 B
- 8 virtual channels per input

Xilinx Versal SoC

released 2019



Lang, Kapre, Pellizzoni, "NOCS '21". "Worst-case propagation delay ..."