Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip

E. Rijpkema, K. Goossens, A. Rădulescu,
J. Dielissen, J. van Meerbergen, P. Wielage,
and E. Waterlander

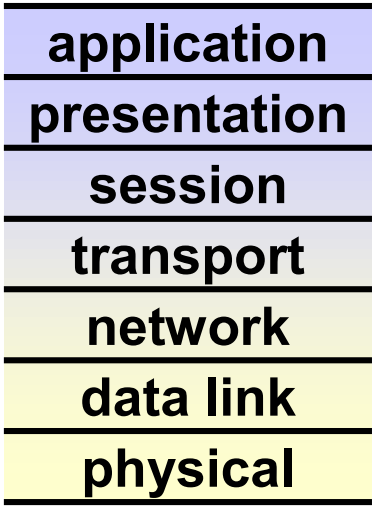
PHILIPS

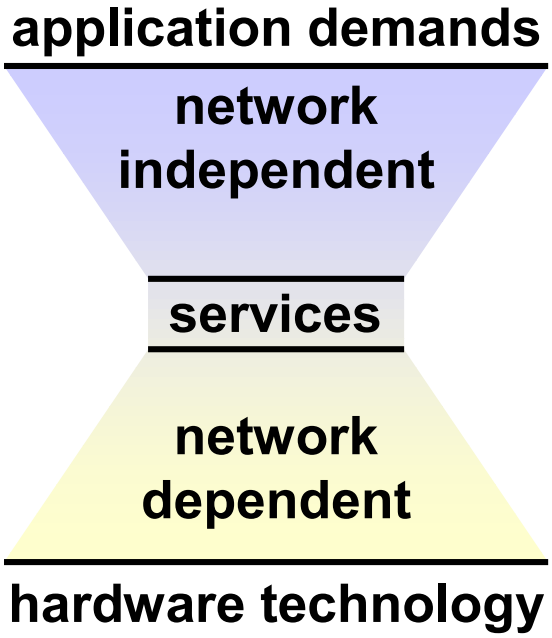
why Networks-on-Chip

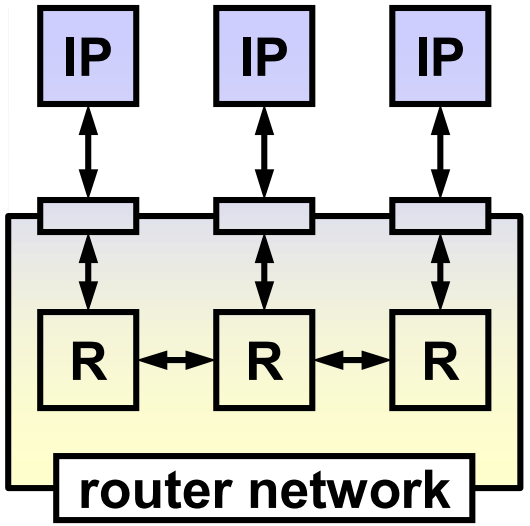
problems observed for SoC design

- deep sub micron
  - wire cost
  - timing closure
- design complexity
  - increasing # of IP blocks
  - increasing dynamism

decouple computation from communication



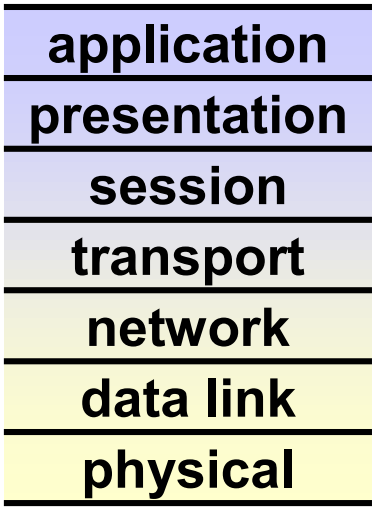


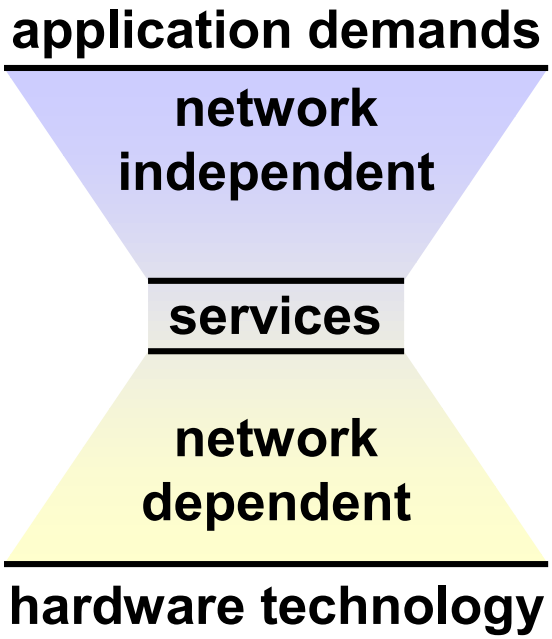


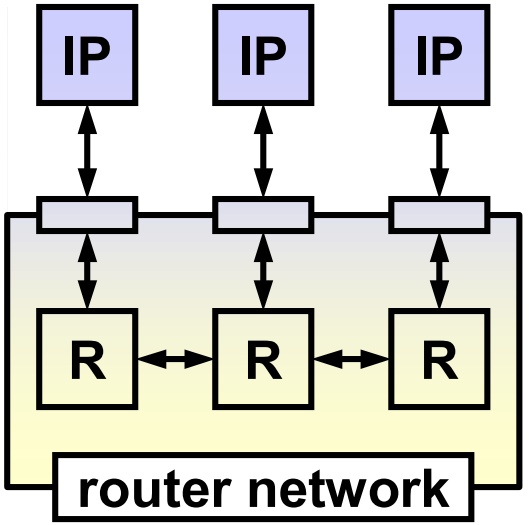
why Networks-on-Chip

- problems observed for SoC design
  - deep sub micron
  - design complexity

- key: decouple computation from communication





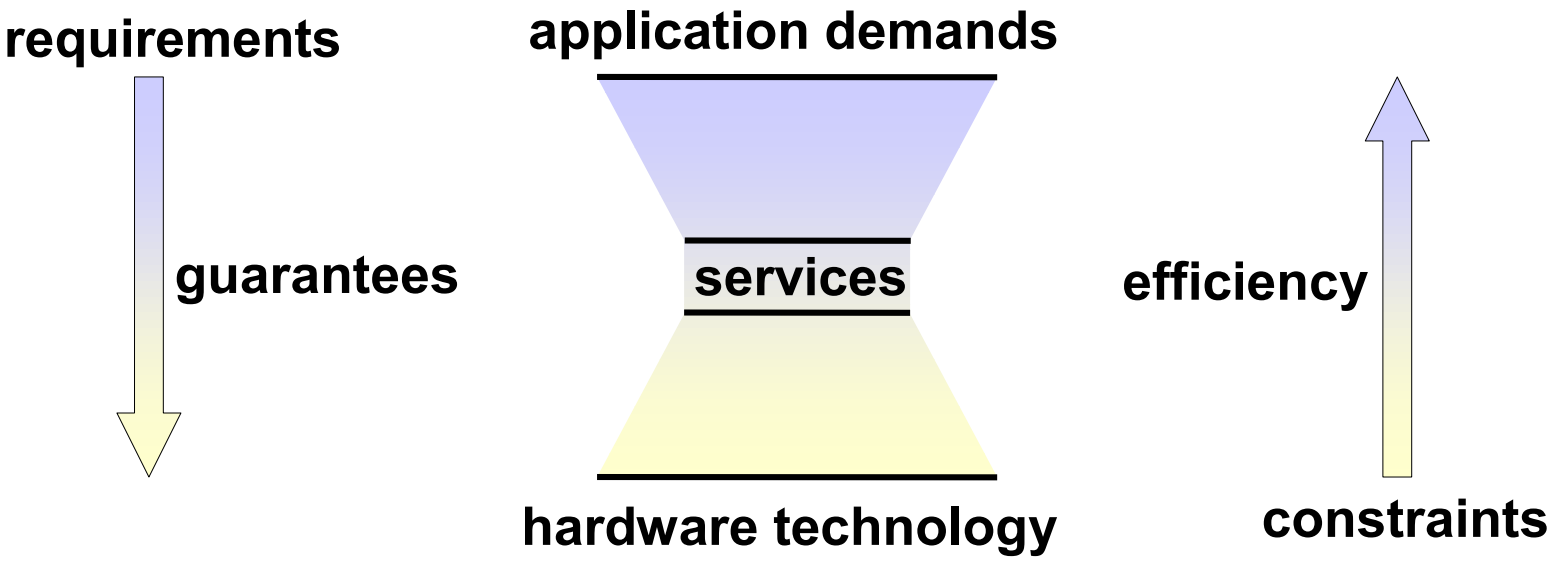


outline

- services
- combined router architecture
- guaranteed throughput router architecture
- best-effort router architecture
- router prototype
- conclusions

services I

- we need a network that is
  - predictable
  - cost effective



- build guarantees on top of guarantees
- efficient network is efficient at every layer

services II

- timeless guarantees
  - guaranteed data integrity
  - guaranteed data delivery
  - guaranteed in-order delivery

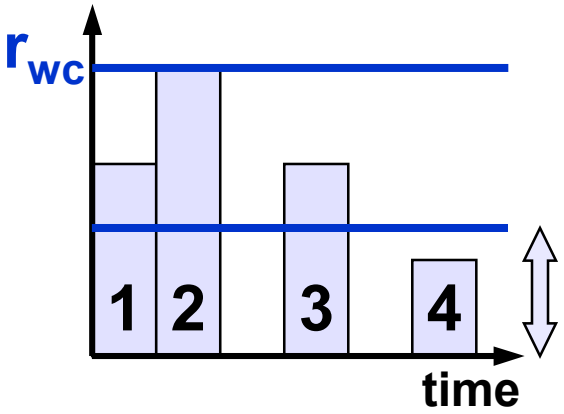
  } best-effort service (BE)

- time related guarantees
  (over bounded time interval)
  - guaranteed throughput
  - guaranteed latency

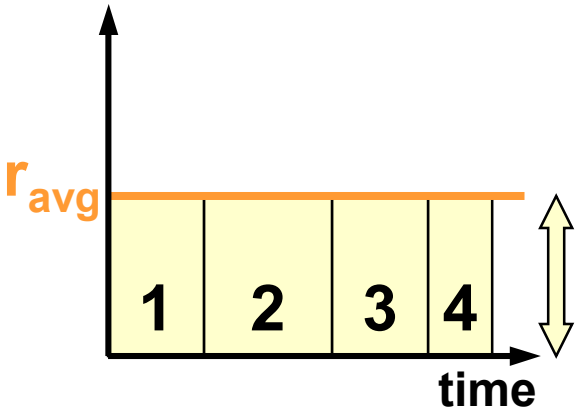
} guaranteed throughput service (GT)

guarantees vs. best-effort

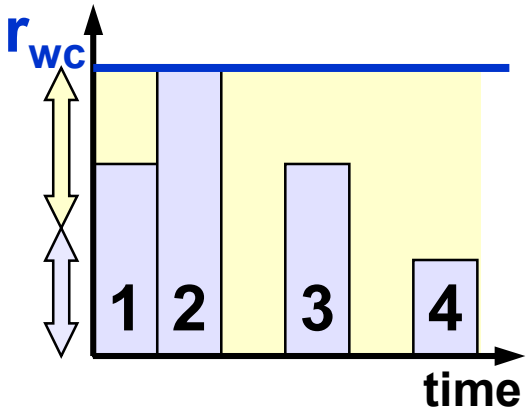
- GT requires dimensioning for guaranteed throughput
- BE requires dimensioning for average throughput



guaranteed throughput (bounded interval)



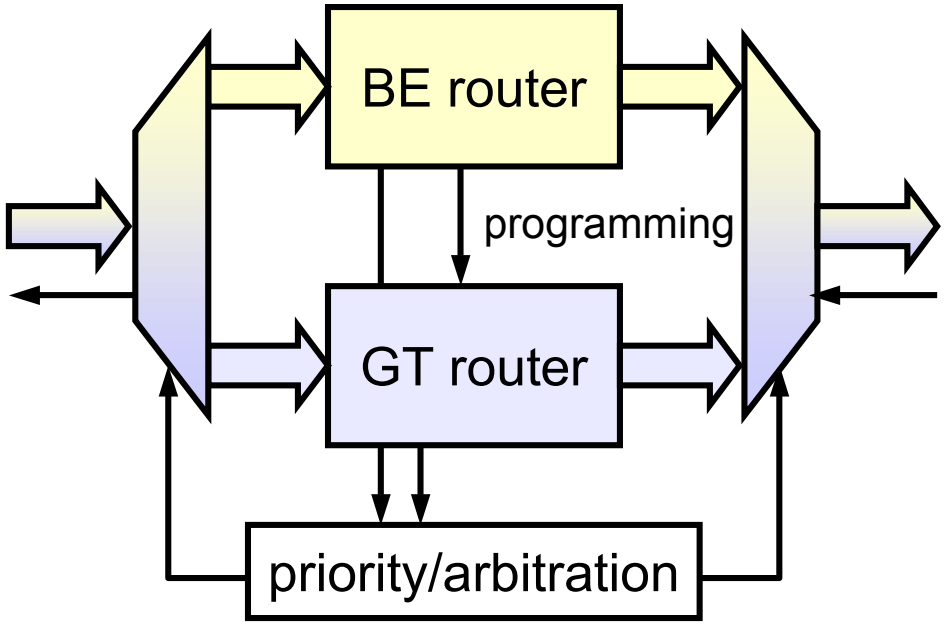
guaranteed delivery

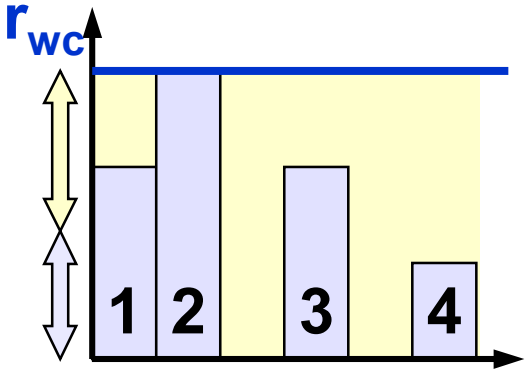


combination is beneficial

BE & GT combined architecture

- conceptually, two disjoint routers
  - a router with GT service class
  - a router with BE service class

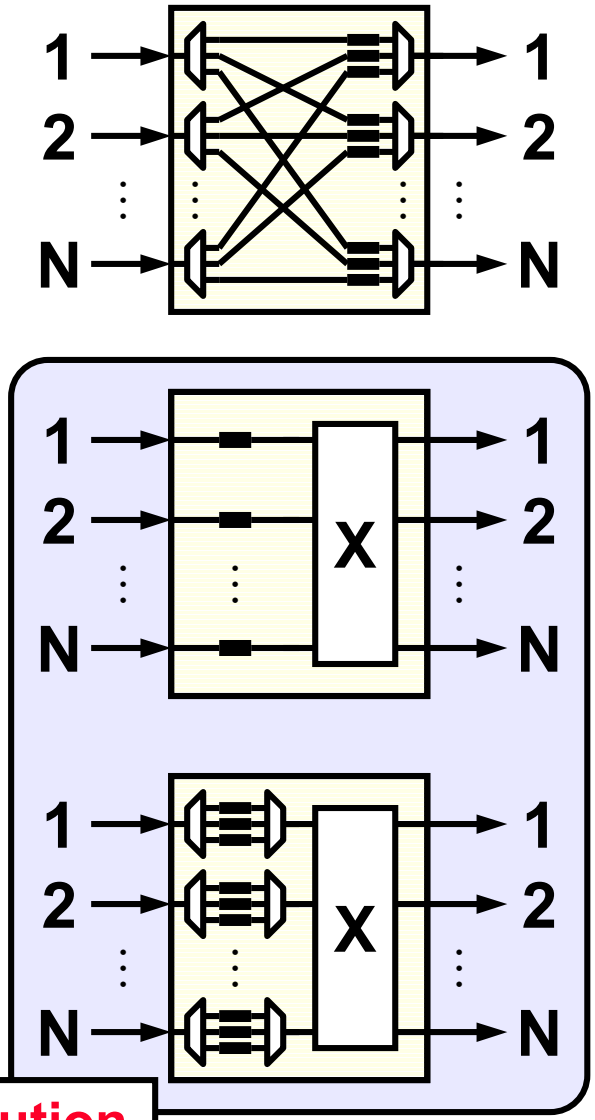




- to obtain an efficient combination routers must have similar architectures

buffering strategy

- output queuing
  - highest cost
  - highest performance
- input queuing
  - lowest cost
  - lowest performance
- virtual output queuing
  - moderate cost
  - high performance



contention

- links in network are shared resources
- contention occurs when multiple data request same link at same time
- GT and BE resolve contention differently

guaranteed throughput

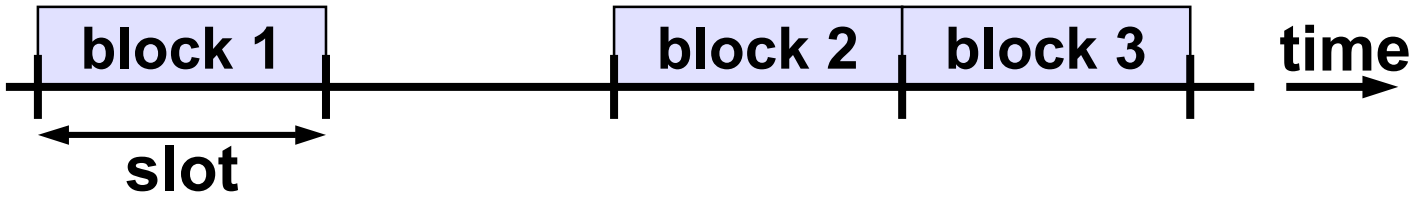
- to guarantee latency or bandwidth over finite interval
  - cannot drop data
  - must bound contention
- rate-based scheduling
  - has high buffer costs (deep fifos/output queuing)
- deadline-based scheduling
  - even higher buffer costs (deep priority queues)
- contention-free routing
  - low buffer costs (shallow fifos)

contention-free routing I

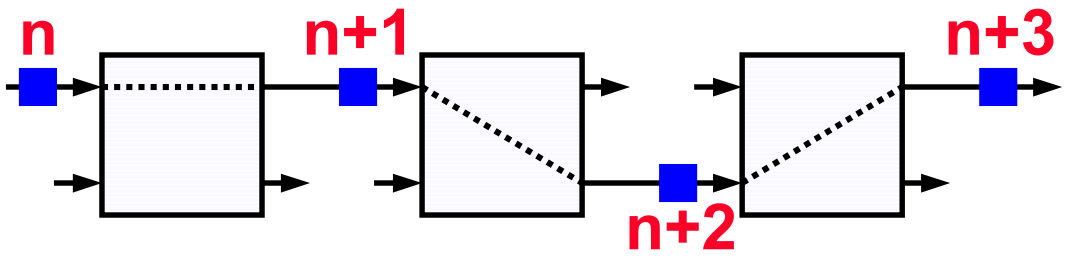
- **scheduling packet injection in network to avoid contention**
  - **in space: disjoint paths as in pure circuit switching**
  - **in time: time-division multiplexing as with a statically scheduled bus**
  - **in time and space: our solution**

contention-free routing II

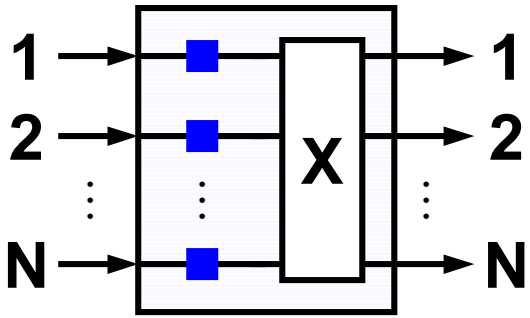
- divide time in slots



- a block is amount of data that fits in a slot
- block entering router in slot n enters next in slot n+1

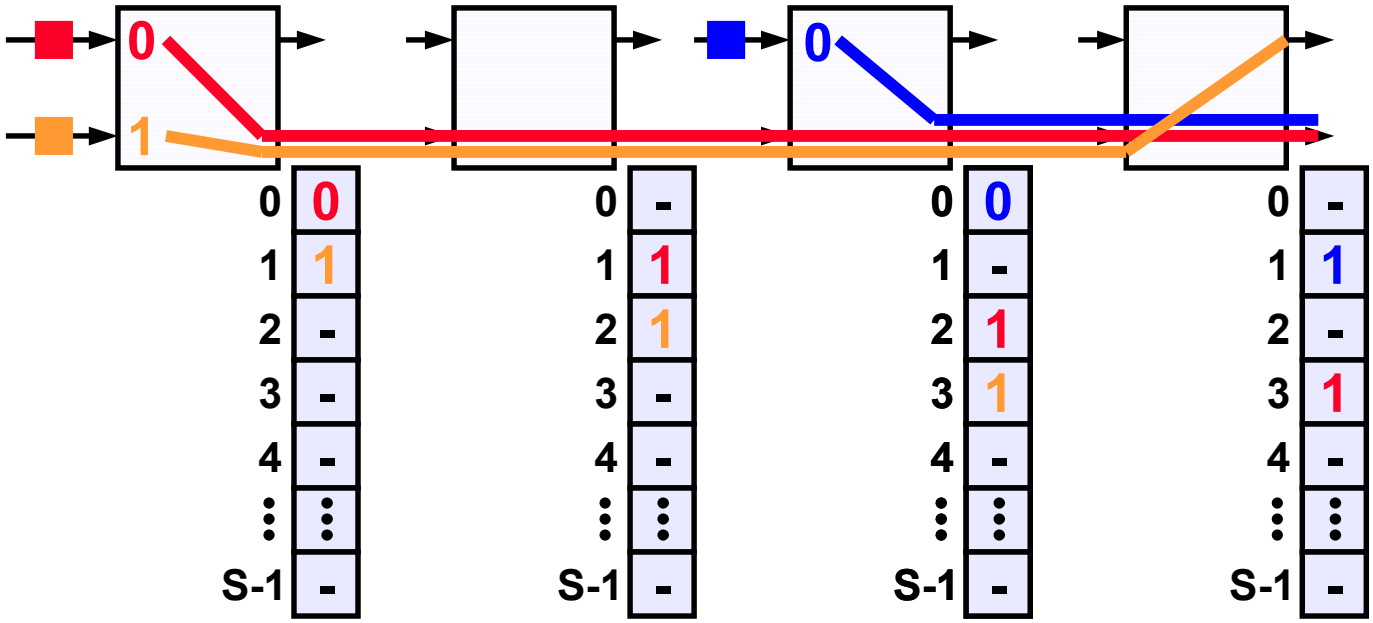


- matches with input queuing



contention-free routing III

- routers have tables that
  - store contention resolution & routing information
  - allow distributed programming



- small blocks → low buffering cost
  || → low latency
  small slots → throughput guarantee on smaller period

best-effort architecture

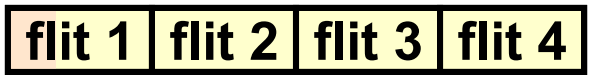
- to ensure high resource utilization
  - statistical multiplexing
  - packet-switching
  - but implement BE service class

- packet-switching
  - network flow control (routing mode)
  - contention resolution

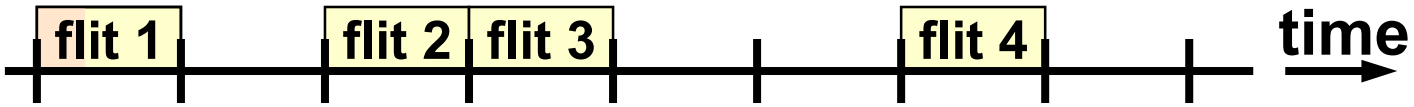
packets and flits

- packet = header + payload

- packet might be transmitted in smaller parts called flits



- flits divide time in iterations and must be scheduled



- smaller flit size → higher scheduling rate
  → lower latency
  → less storage

network flow control (routing mode)

network flow control \ performance/cost	per router	
	latency	storage

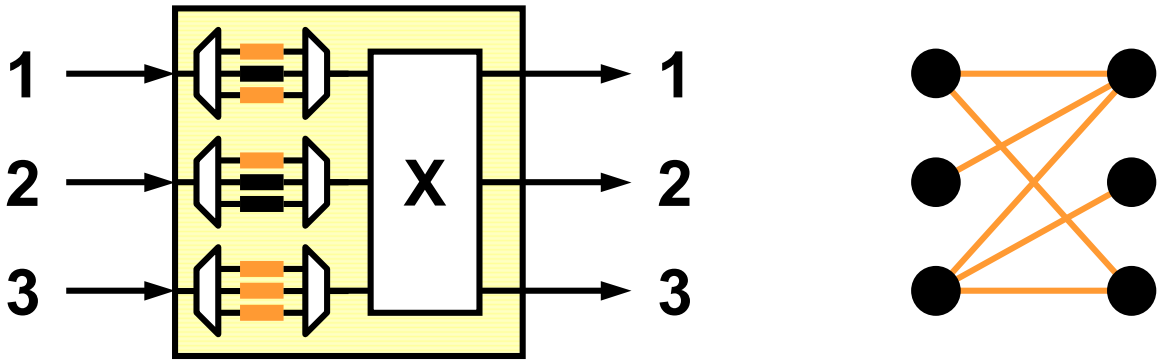
• store and forward routing
  • first receive whole packet
  • then transmit whole packet
	packet	packet

• virtual cut-through routing
  • send flit immediately
  • if next router can receive entire packet
	flit	packet

• wormhole routing
  • send flit immediately
  • if next router can receive that flit
	flit	flit

contention resolution

- queuing at input → set paths from inputs to outputs
- router has switch
- bipartite graph matching



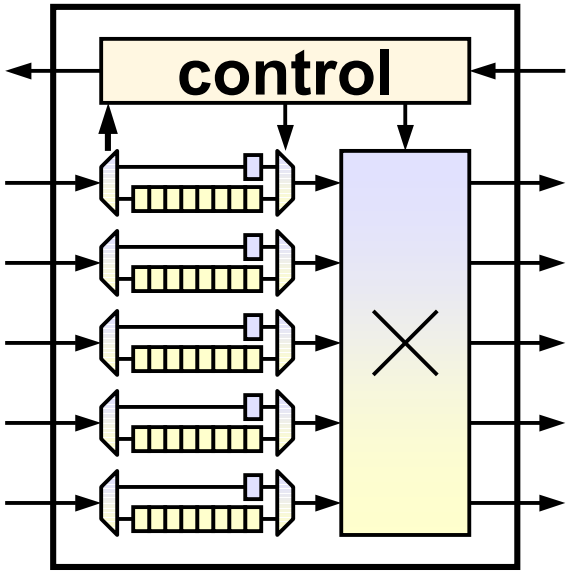
- algorithm must
  - be fair
  - have low complexity (to schedule at flit rate)
  - approximation of maximal matching

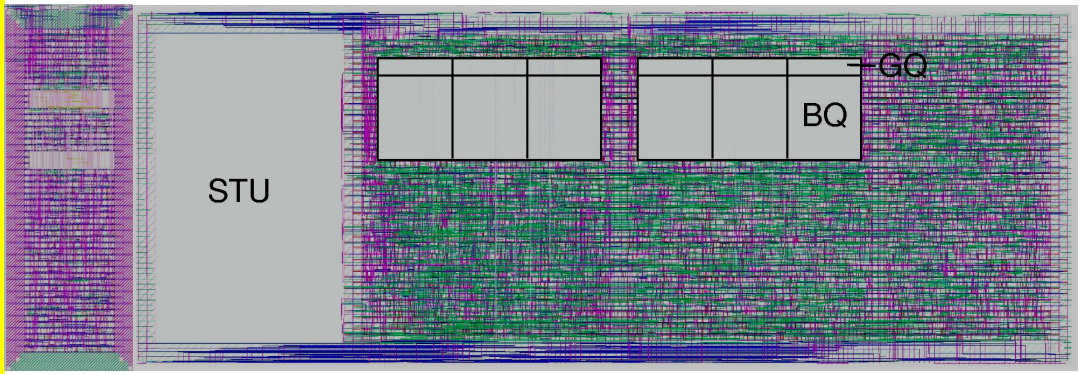
combining GT and BE

- links must be shared by GT and BE traffic
- grain size of interleaving must match
- block size = flit size
- smallest value for this is given by implementation
  - minimize scheduler latency → L
  - maximize data path speed → F
  - flit size = block size = F · L

router prototype

- snapshot of current prototype router:
  - input queuing
  - arity 5
  - 32 bits wide words
  - 8 flits deep BE queues
  - 256 slots





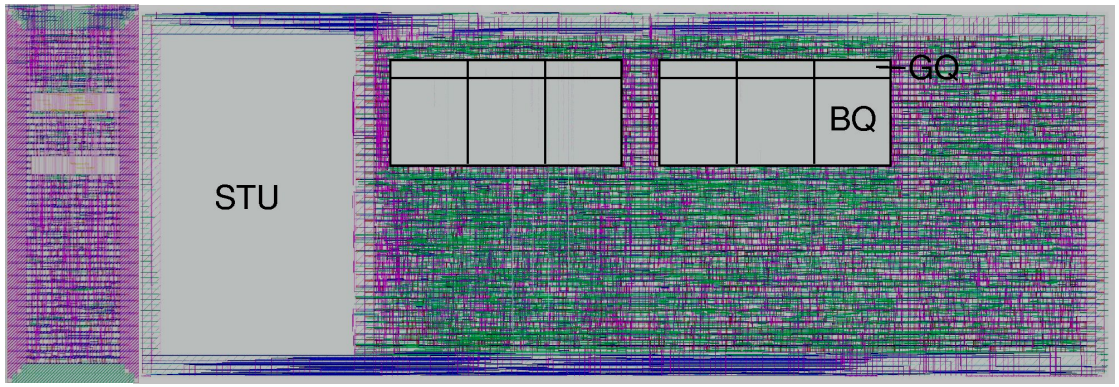
- 0.25 mm^2 CMOS12
- 500 MHz data path
- 166 MHz control path
- flit size is 3 words
- throughput per link: 500MHz·32bits = 16Gb/s

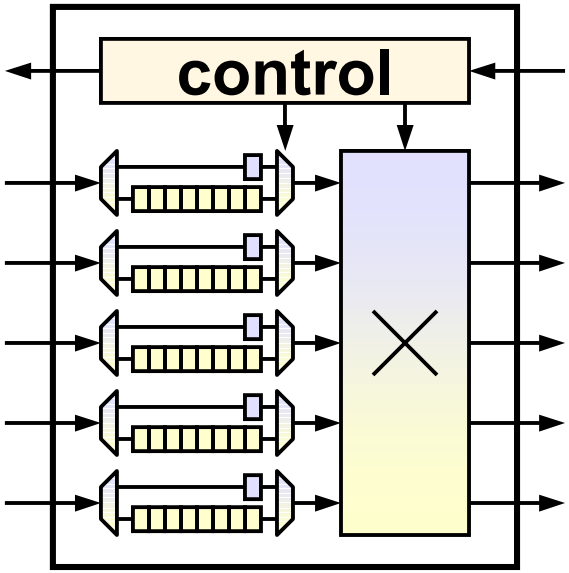
conclusions

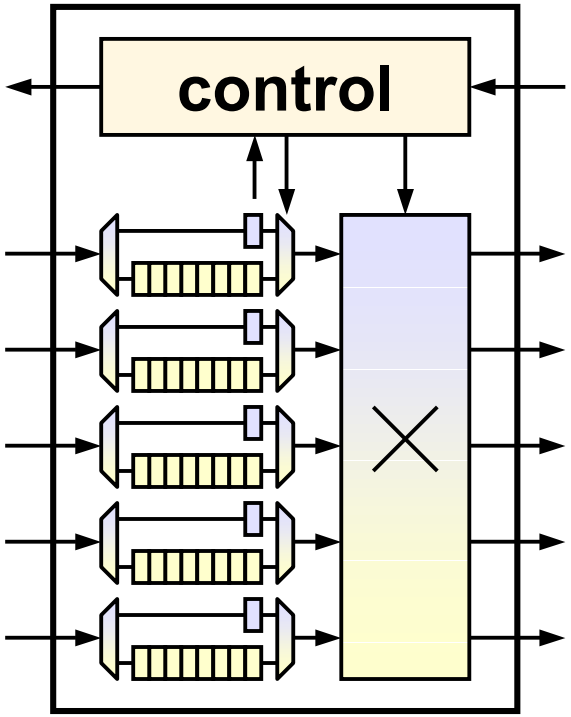
- for NoCs, guaranteed services are essential
- demonstrated the useful combination of:
  - BE service class → timeless guarantees
  - GT service class → BE + time related guarantees
- made trade-offs to come to efficient combined router
- proved feasibility with router prototype

router prototype

- **snapshot of current prototype router:**
  - **5 input and 5 output ports (arity 5)**
  - **0.25 mm^2 CMOS12**
  - **500 MHz data path, 166 MHz control path**
  - **flit size of 3 words of 32 bits**
  - **500x32 = 16 Gb/s throughput per link**
  - **256 slots & 5x1 flit fifos for guaranteed-throughput traffic**
  - **6x8 flit fifos for best-effort traffic**




control


control