NoC (Networks on a Chip) Platform Design

周哲民 Jou, Jer Min

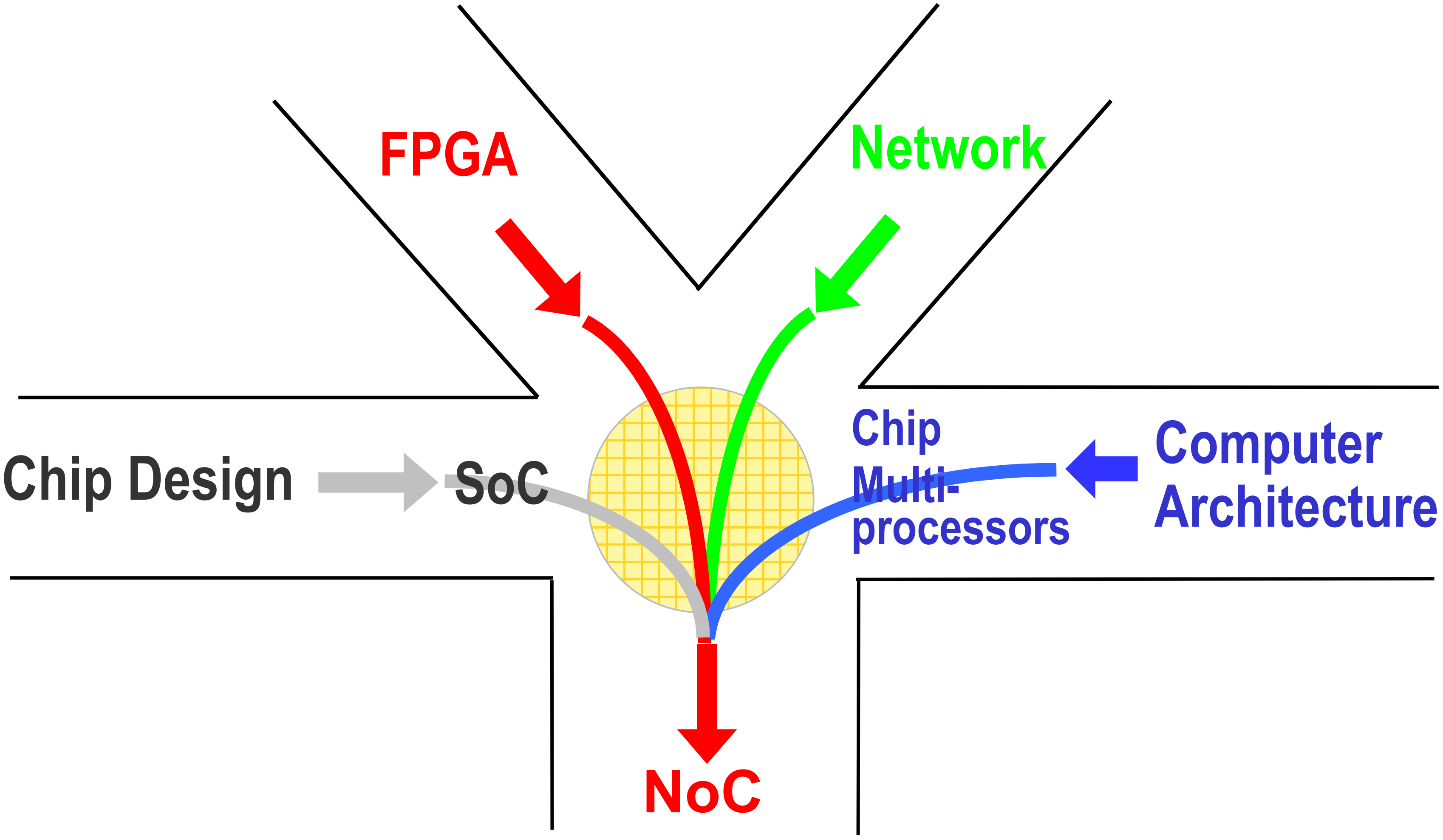
Department of Electrical Engineering
National Cheng Kung University
Tainan, Taiwan, R.O.C.

Email: jou@j92a21.ee.ncku.edu.tw

Outlines

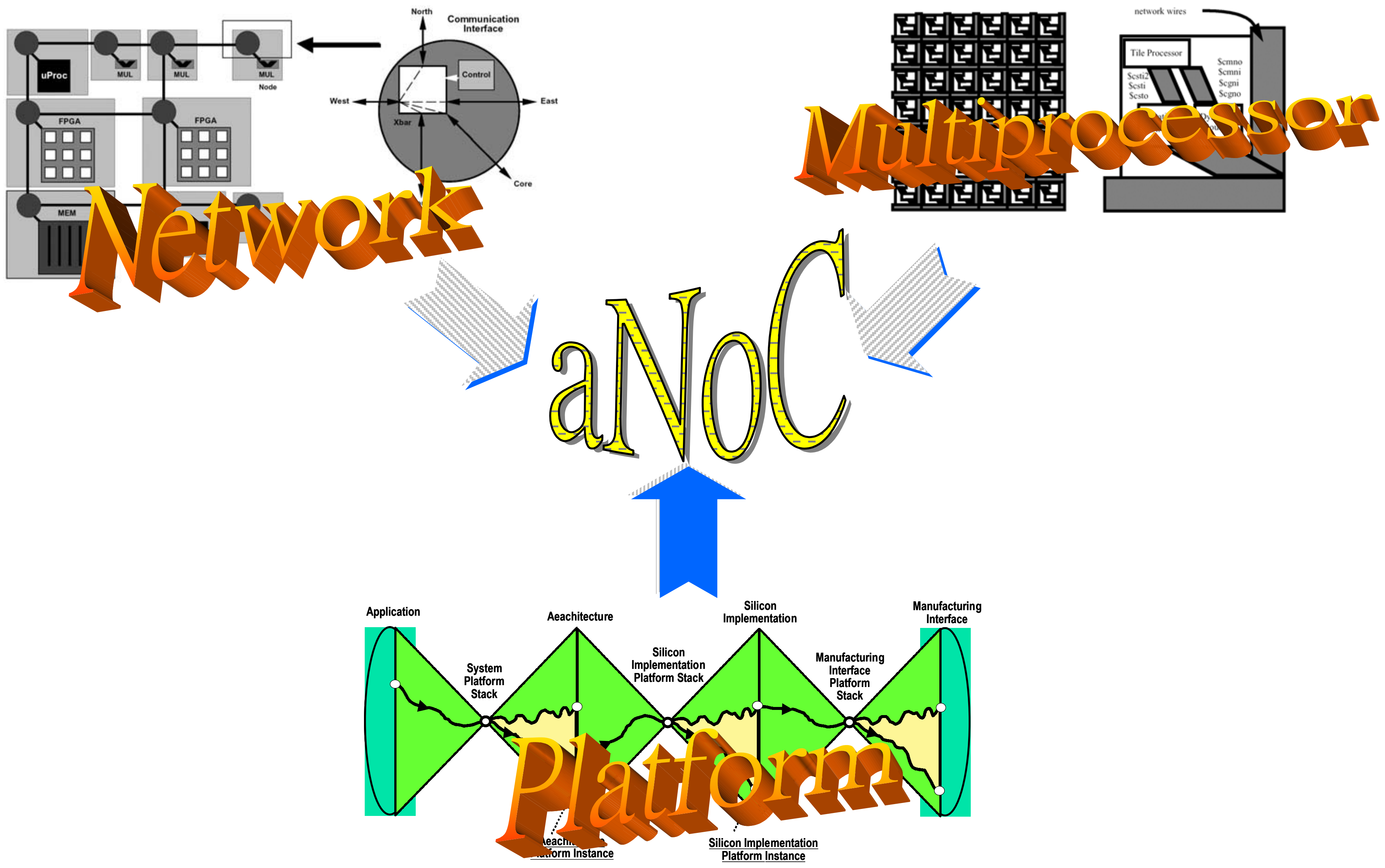
- Introduction of Networks-on-a-Chip (NoC) and NoC Platforms
- Current Communication Architectures
- NoC Challenges
- NoC Design Factors
- NoC Design Methodologies

An Interesting Cross-Road



(Networks-on-a-Chip;
On-chip interconnect networks)

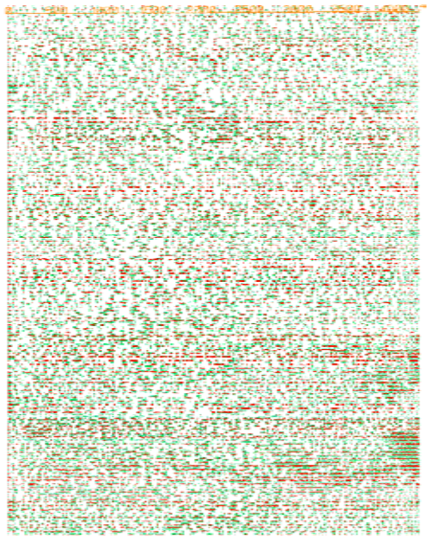
Perspectives of the NoC



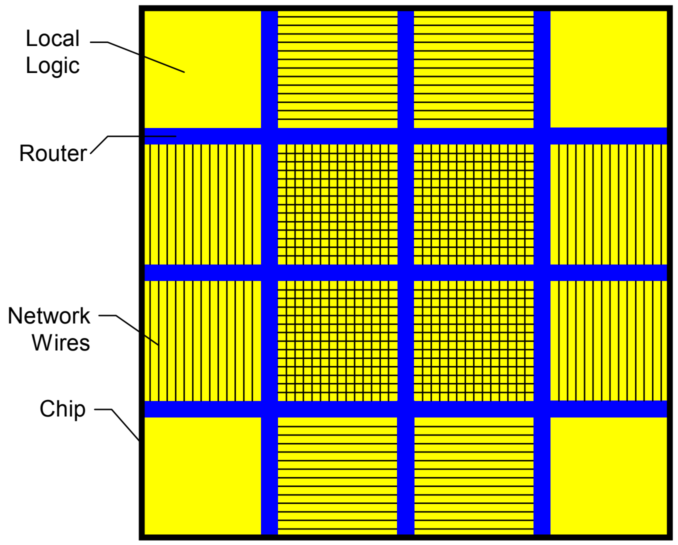
On-Chip Interconnection Networks

- Most chip wires are idle most of the time
- Dedicated global wiring ⇒ a shared network

Dedicated wiring



On-chip network



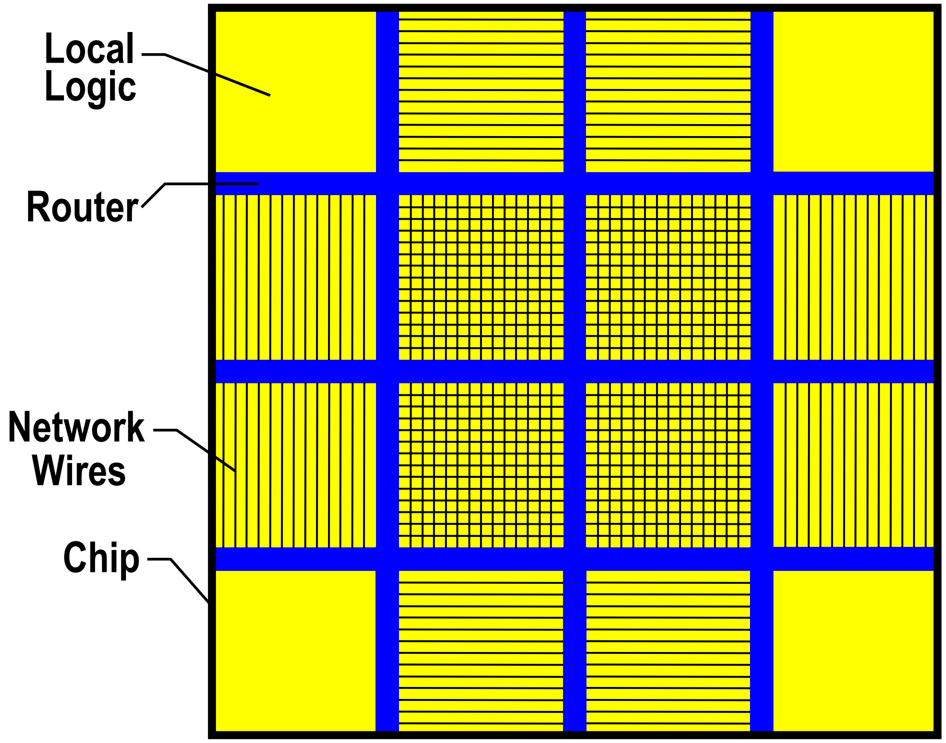
- Don't dedicate wires to signals, share wires across multiple signals
- Route *packets* not *wires*
- Organize global wiring as an on-chip interconnection network
  - allows the wiring to be shared, and keeps wires busy most of the time
  - allows a single global interconnect to be re-used on multiple designs
  - makes global wiring regular, and highly optimized

Dedicated Wires vs. Network*

Dedicated Wiring	On-Chip Network
Spaghetti wiring	Ordered wiring
Variation makes it hard to model crosstalk, returns, length, R & C.	No variation, so easy to exactly model XT, returns, R and C.
Drivers sized for 'wire model' – 99% too large, 1% too small	Driver sized exactly for wire
Hard to use advanced signaling	Easy to use advanced signaling
Low duty factor	High duty factor
No protocol overhead	Small protocol overhead

On-Chip Interconnection Networks

- Many chips, same global wiring
  - Carefully optimized wiring
  - Well characterized
  - Optimized circuits
    - 0.1x power 0.3x delay
- Efficient protocols
  - dynamic routing with pipelined control
  - statically scheduled
    - Static network
- Standard interface



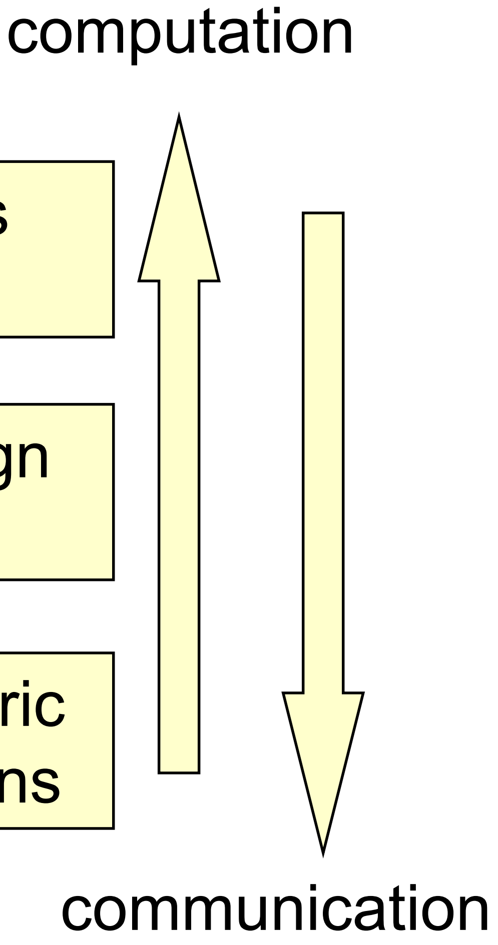
NoC

Emerging Platforms & Architectures

Functions on a chip	Chips on PCBs External Connections External storage
Algorithms on a chip	Hardwired computation Hardwired interconnectivity Centralised storage
System on a chip	Programmable computation Hardwired interconnectivity Partially distributed storage
Networks on a chip	Programmable computation Programmable interconnectivity Fully distributed storage

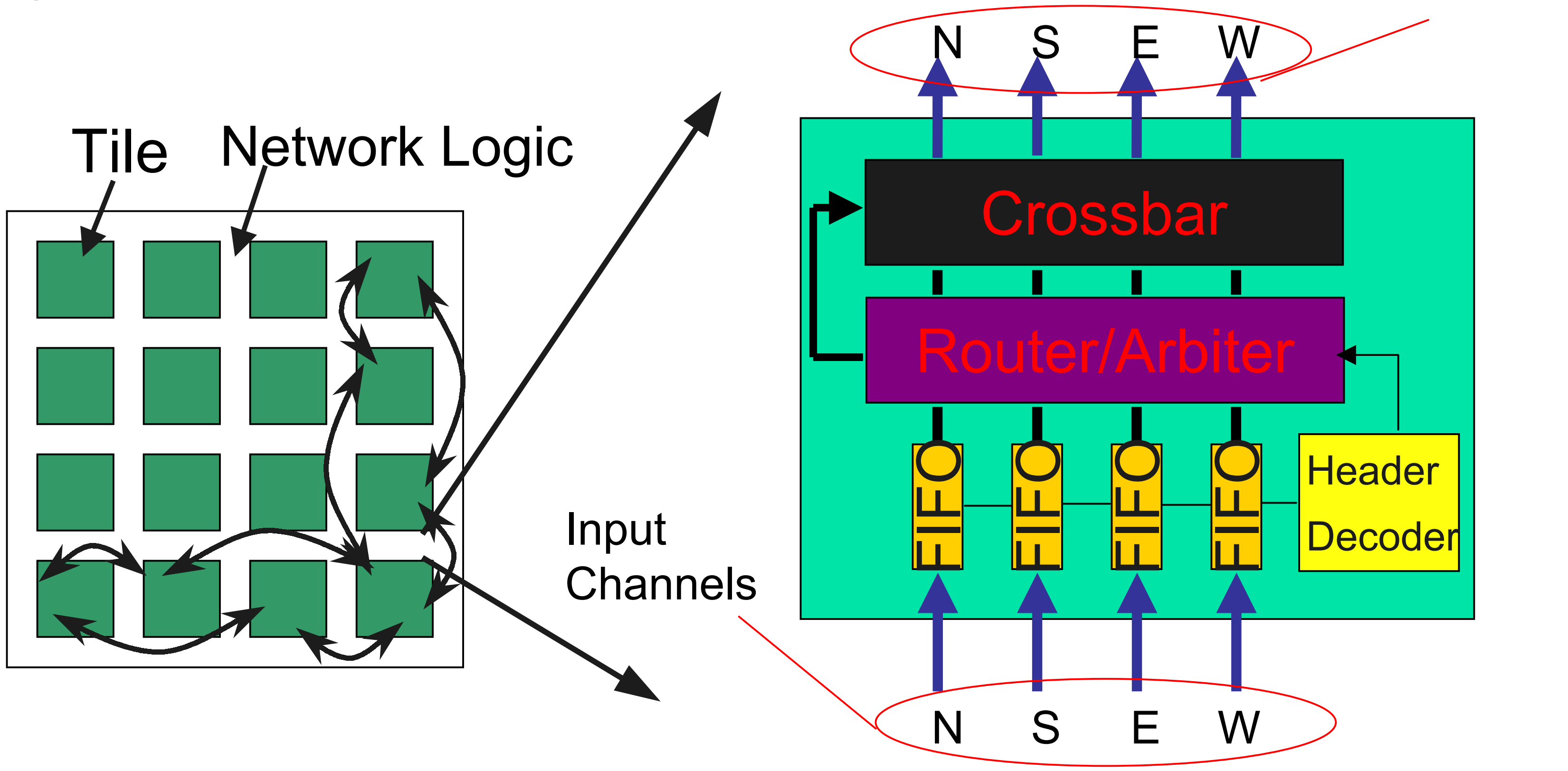
Levels of IC architecture

Embedded processors	HLS + SW-compilers ILP, VLIW
Multi-processor architectures	Platform based design Reuse of IP
Networks-on-Chip	Communication centric separation of concerns

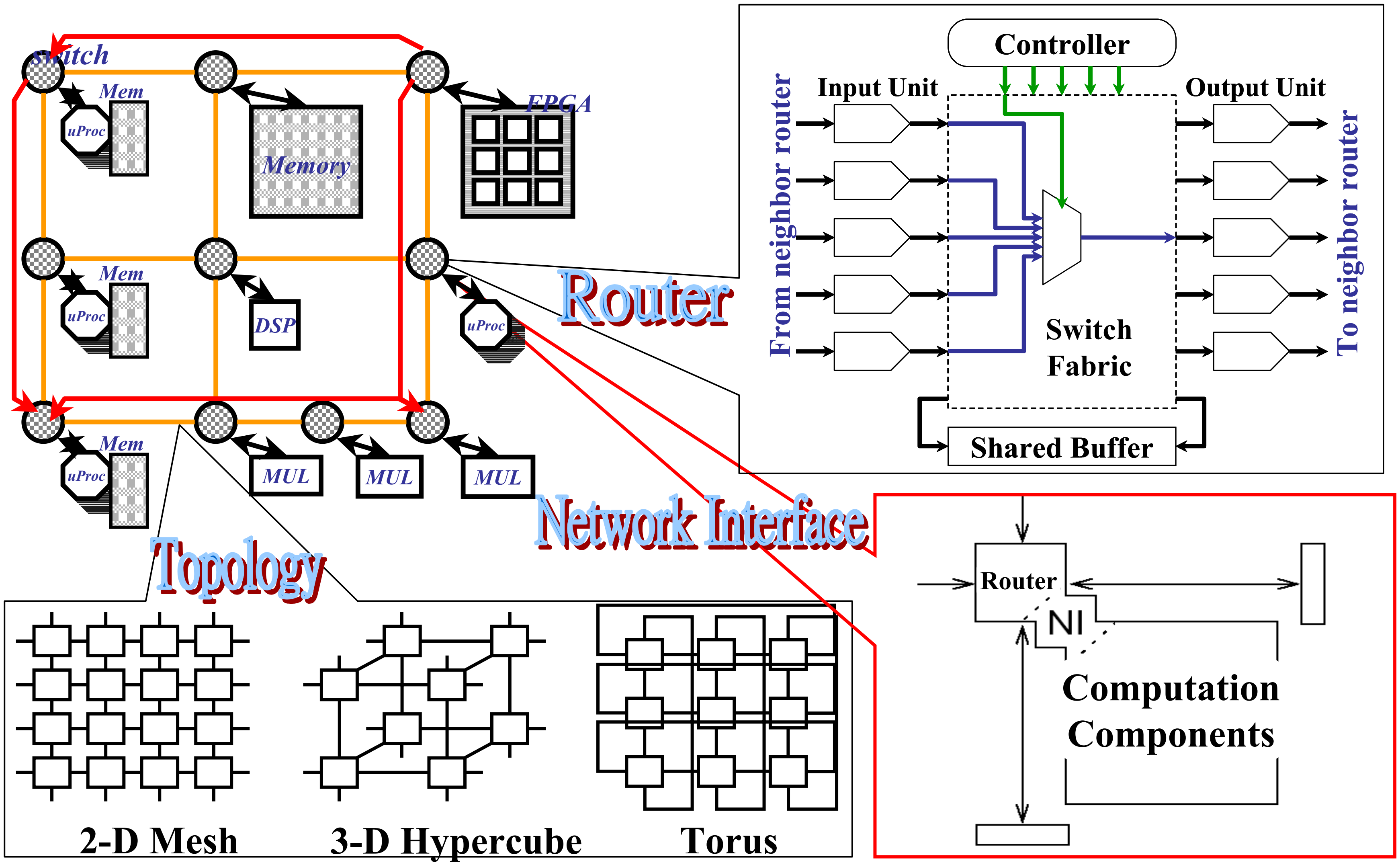


An Architecture of NoC

- Tile-based architecture
  - Well-controlled electrical parameter
  - Reliable interconnection
  - High performance

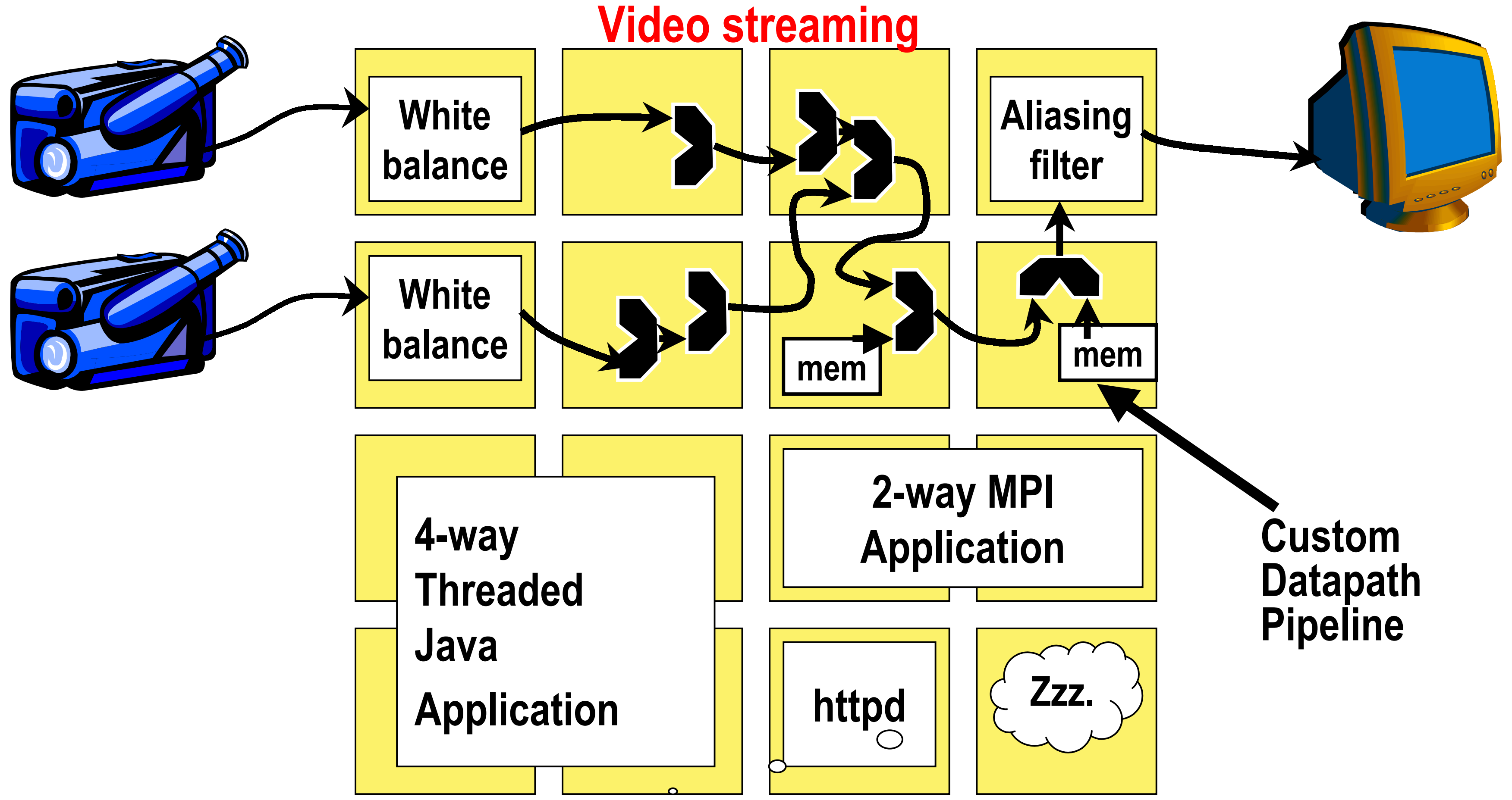


NoC Communication Components



NoC is Composable (examples)

- Mix and match types of parallelism



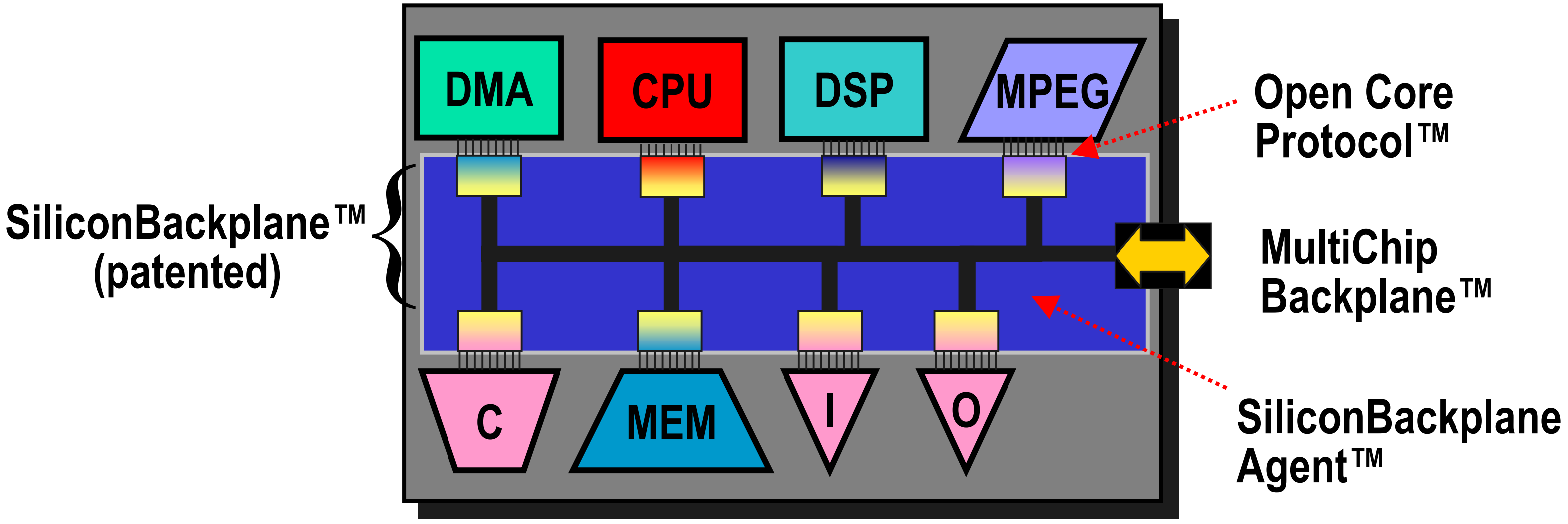
It is expected that most of the future applications for NoCs are going to provide audio and video user interface

NoC Platform

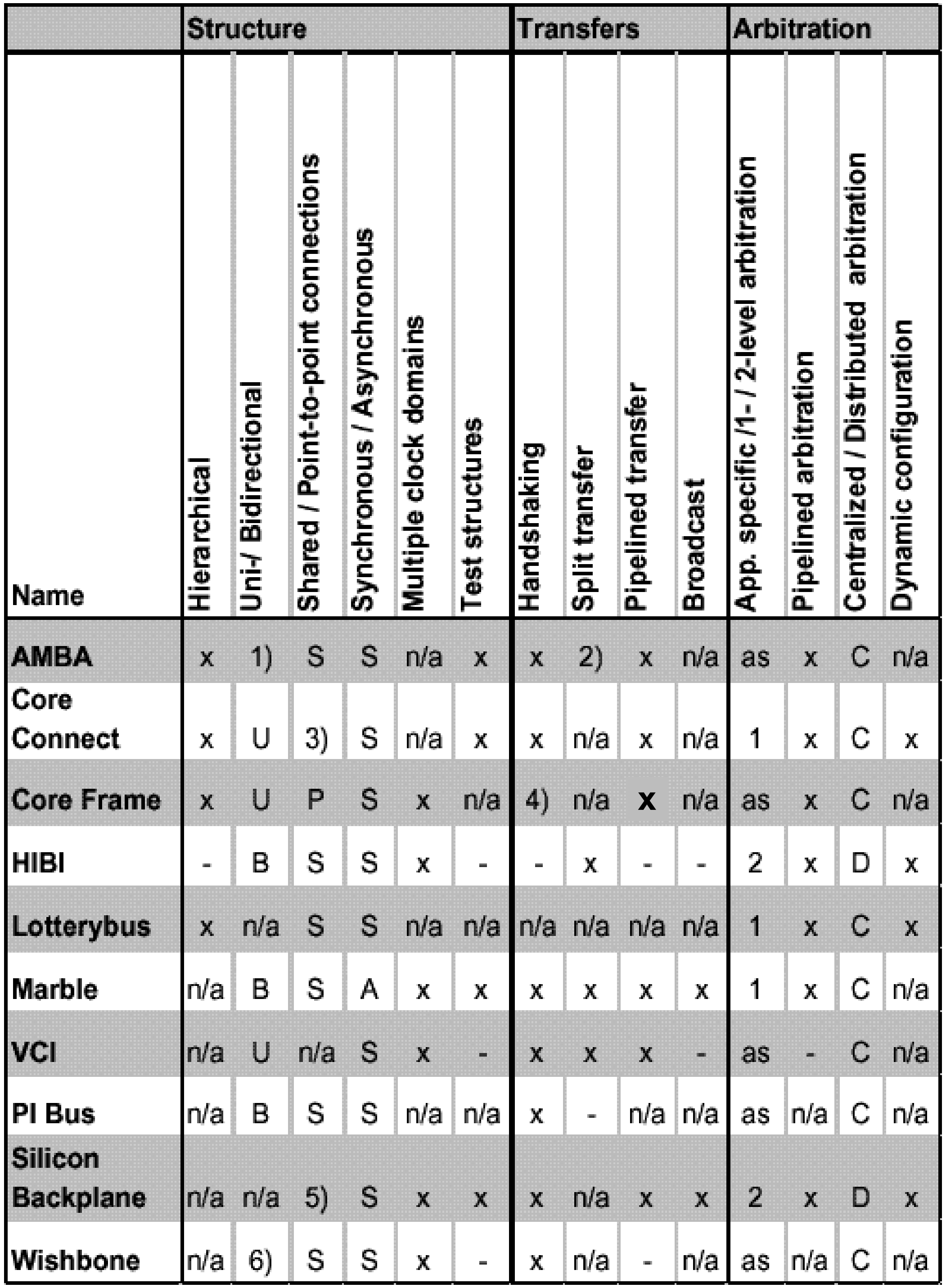
- Platform based design (PBD) methodology: reuse of components, and also reuse of system architectures and topologies
- NoC platform: <u>an architectural template</u> consisting of <u>a set of resources interconnected</u> through <u>a topology of routers</u> (<u>switches</u>), for complex embedded products – a system design platform
  - for designs which are the basis for several product variants;
  - for applications with a heterogeneous task mix;
  - for applications with stringent TTM requirements;
  - for products where reuse both at the block and the function and system level
  - to reuse at all levels of system design
- Intensive communication requirement is very critical for NoC platforms
  - it will depend on the design of the routers and routing algorithms

Present: Current Communication Architectures

- Virtual Socket Interface (VSI) Alliance (VSIA); Virtual component interface (VCI): PVCI, BVCI, AVCI
- ARM's AMBA
  - AHB - CPU, DMA, Memory
  - APB - low power, parallel I/O
- Standard bus key element of strategy for SoC-oriented ASIC vendors (such as IBM CoreConnect, Palmbus Core Frame)
- Sonics SoC integration architecture



Summary of On-Chip Buses Properties



	Structure						Transfers				Arbitration			
Name	Hierarchical	Uni-/ Bidirectional	Shared / Point-to-point connections	Synchronous / Asynchronous	Multiple clock domains	Test structures	Handshaking	Split transfer	Pipelined transfer	Broadcast	App. specific /1- / 2-level arbitration	Pipelined arbitration	Centralized / Distributed arbitration	Dynamic configuration
AMBA	x	1)	S	S	n/a	x	x	2)	x	n/a	as	x	C	n/a
Core Connect	x	U	3)	S	n/a	x	x	n/a	x	n/a	1	x	C	x
Core Frame	x	U	P	S	x	n/a	4)	n/a	**x**	n/a	as	x	C	n/a
HIBI	-	B	S	S	x	-	-	x	-	-	2	x	D	x
Lotterybus	x	n/a	S	S	n/a	n/a	n/a	n/a	n/a	n/a	1	x	C	x
Marble	n/a	B	S	A	x	x	x	x	x	x	1	x	C	n/a
VCI	n/a	U	n/a	S	x	-	x	x	x	-	as	-	C	n/a
PI Bus	n/a	B	S	S	n/a	n/a	x	-	n/a	n/a	as	n/a	C	n/a
Silicon Backplane	n/a	n/a	5)	S	x	x	x	n/a	x	x	2	x	D	x
Wishbone	n/a	6)	S	S	x	-	x	n/a	-	n/a	as	n/a	C	n/a

Table exceptions

- 1) AHB and APB unidirectional, ASB bidirectional
- 2) AHB and ASB allow split transfers
- 3) Data lines shared, control lines point-to-point ring
- 4) Palmbus uses handshaking, Mbus does not
- 5) Data lines shared, control lines point-to-point
- 6) Both possible, unidirectional recommended
- 7) CoreFrame is a nonpipelined bus
- 8) Silicon Backplane uses (TDMA) based arbitration

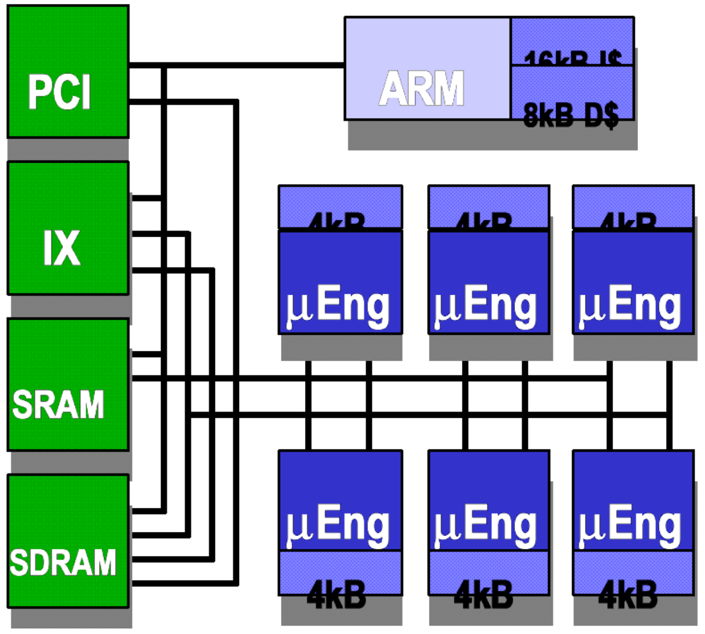
Bus-versus-Network Arguments

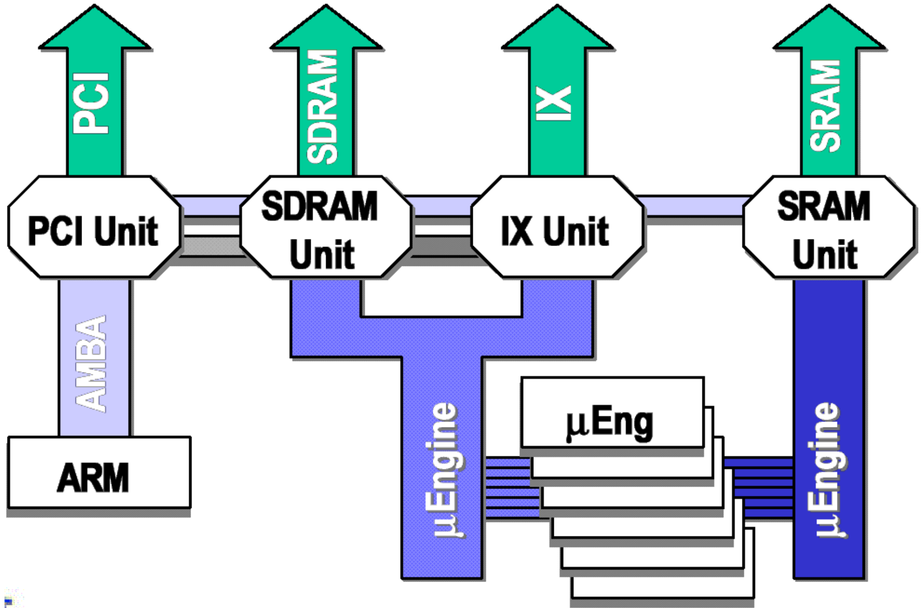
Bus Pros & Cons			Network Pros & Cons
Every unit attached adds parasitic capacitance, therefore electrical performance degrades with growth.	-	+	Only point-to-point one-way wires are used, for all network sizes.
Bus timing is difficult in a deep submicron process.	-	+	Network wires can be pipelined because the network protocol is globally asynchronous.
Bus testability is problematic and slow.	-	+	Dedicated BIST is fast and complete.
Bus arbiter delay grows with the number of masters. The arbiter is also instance-specific.	-	+	Routing decisions are distributed and the same router is reinstanciated, for all network sizes.
Bandwidth is limited and shared by all units attached.	-	+	Aggregated bandwidth scales with the network size.
Bus latency is zero once arbiter has granted control.	+	-	Internal network contention causes a small latency.
The silicon cost of a bus is near zero.	+	-	The network has a significant silicon area.
Any bus is almost directly compatible with most available IPs, including software running on CPUs.	+	-	Bus-oriented IPs need smart wrappers. Software needs clean synchronization in multiprocessor systems.
The concepts are simple and well understood.	+	-	System designers need *reeducation* for new concepts.

Bus based architecture is not scalable

IXP1200's Communication Architecture

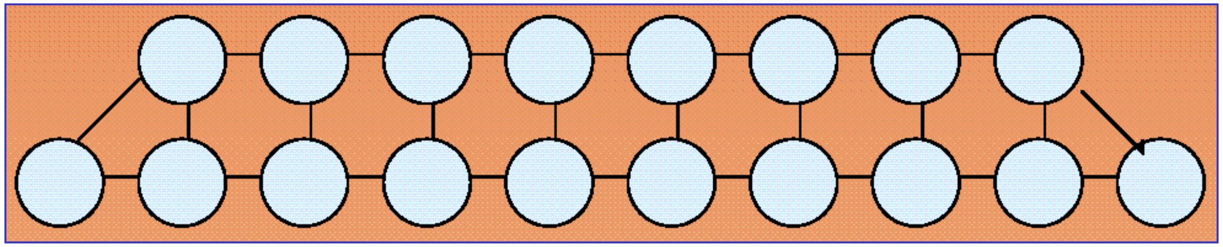
- 4 External Buses
  - PCI
  - IX
  - SRAM
  - SDRAM
- 5 Internal Buses
  - ARM AMBA
  - µEngine to SRAM
  - µEngine to SDRAM & IX
  - IX to SDRAM
  - PCI to SDRAM
- Communication-centric view
  - Interface units look like network switches
  - Interconnect different physical bus implementations
  - Implement protocols that dictate how data is moved between buses

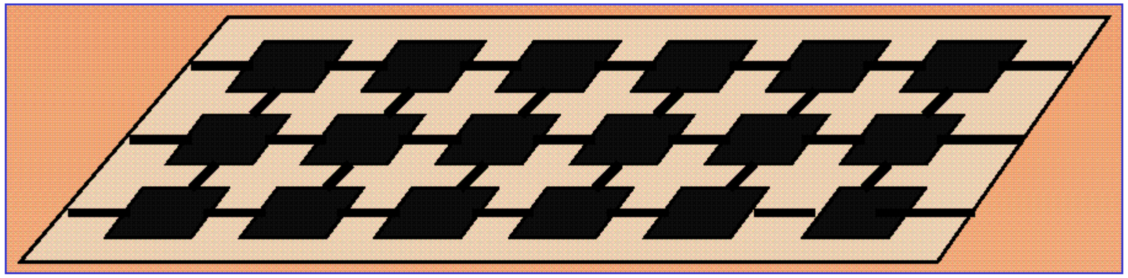


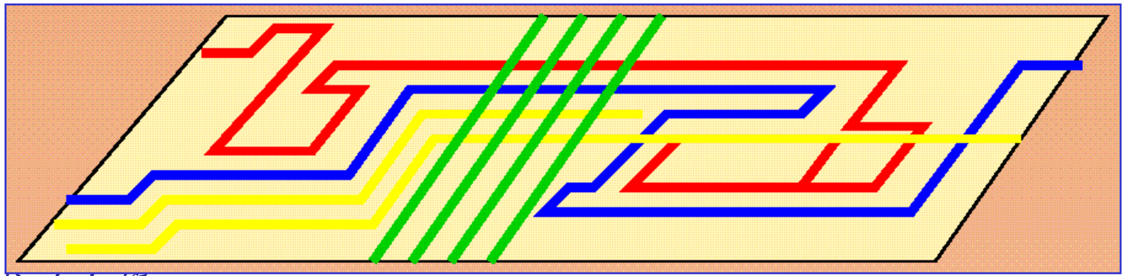


NoC Challenges: Functions, Architecture, and Physics

# Gates	# Processors	Year
6M	4	2000
24M	16	2003
96M	64	2006
384M	256	2009

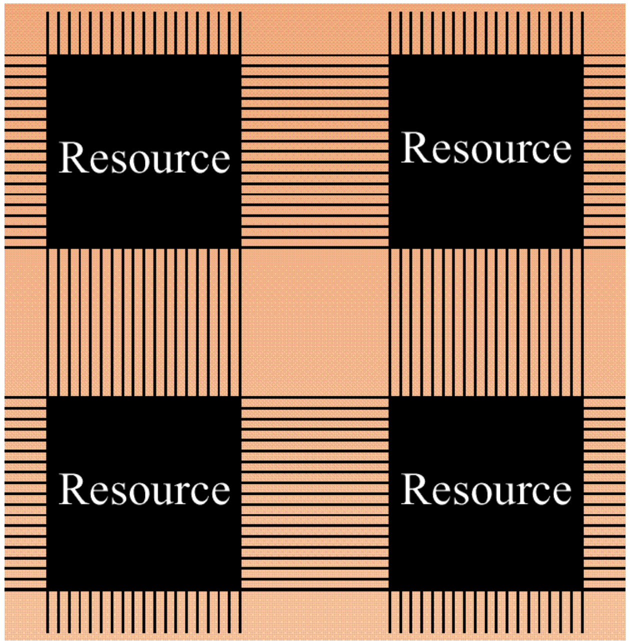

Concurrent processes


Large number of resoucres


Physical issues

Physical Issues Challenges

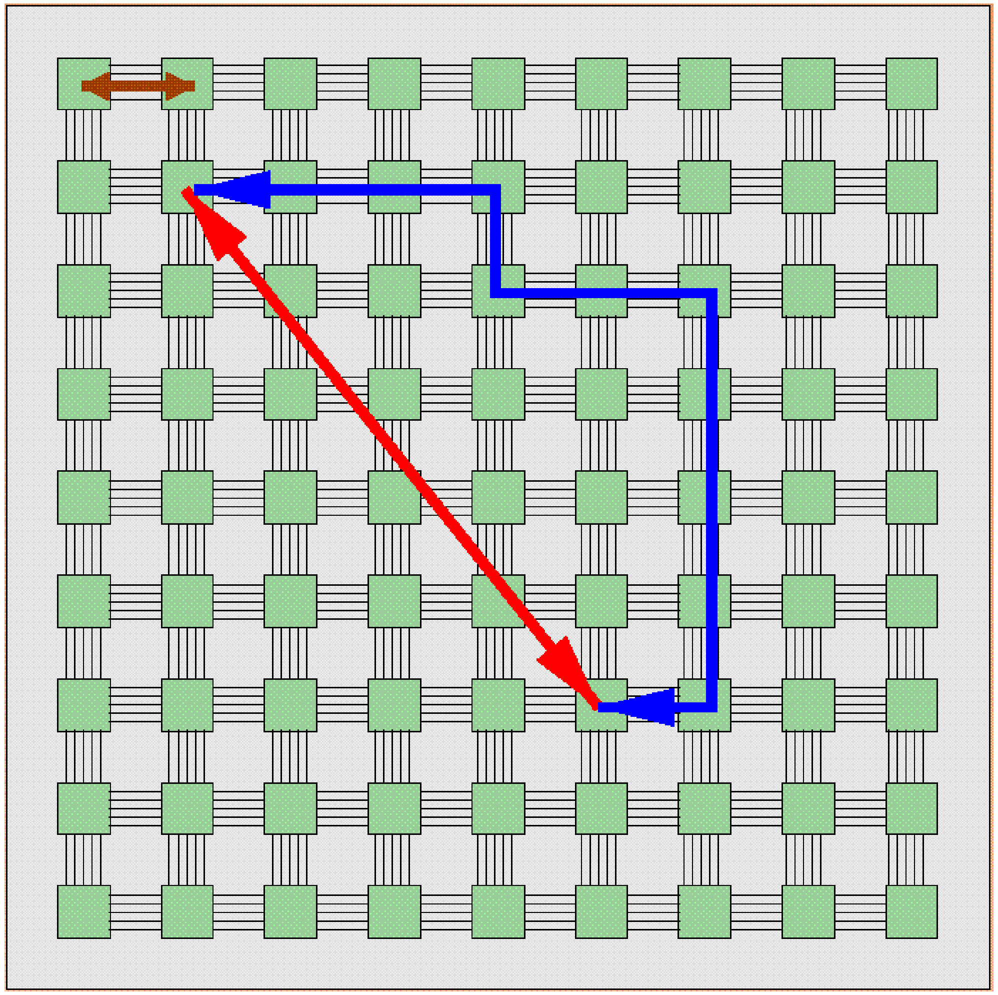
- Deep submicron effects, noise, signal integrity: cross-talk, electro-magnetic interference (EMI)
- Interconnect and global communications
- Power consumption, power delivery
- Clock distribution: timing errors
- Memory integration (50-80% of the chip)



Scenario:

- 60 nm CMOS
- 22 × 22 mm Chip size
- 2 × 2 mm resource size
- 300 nm minumum wire pitch
- 6600 wires between two resources on each metal layer

Architecture Challenges

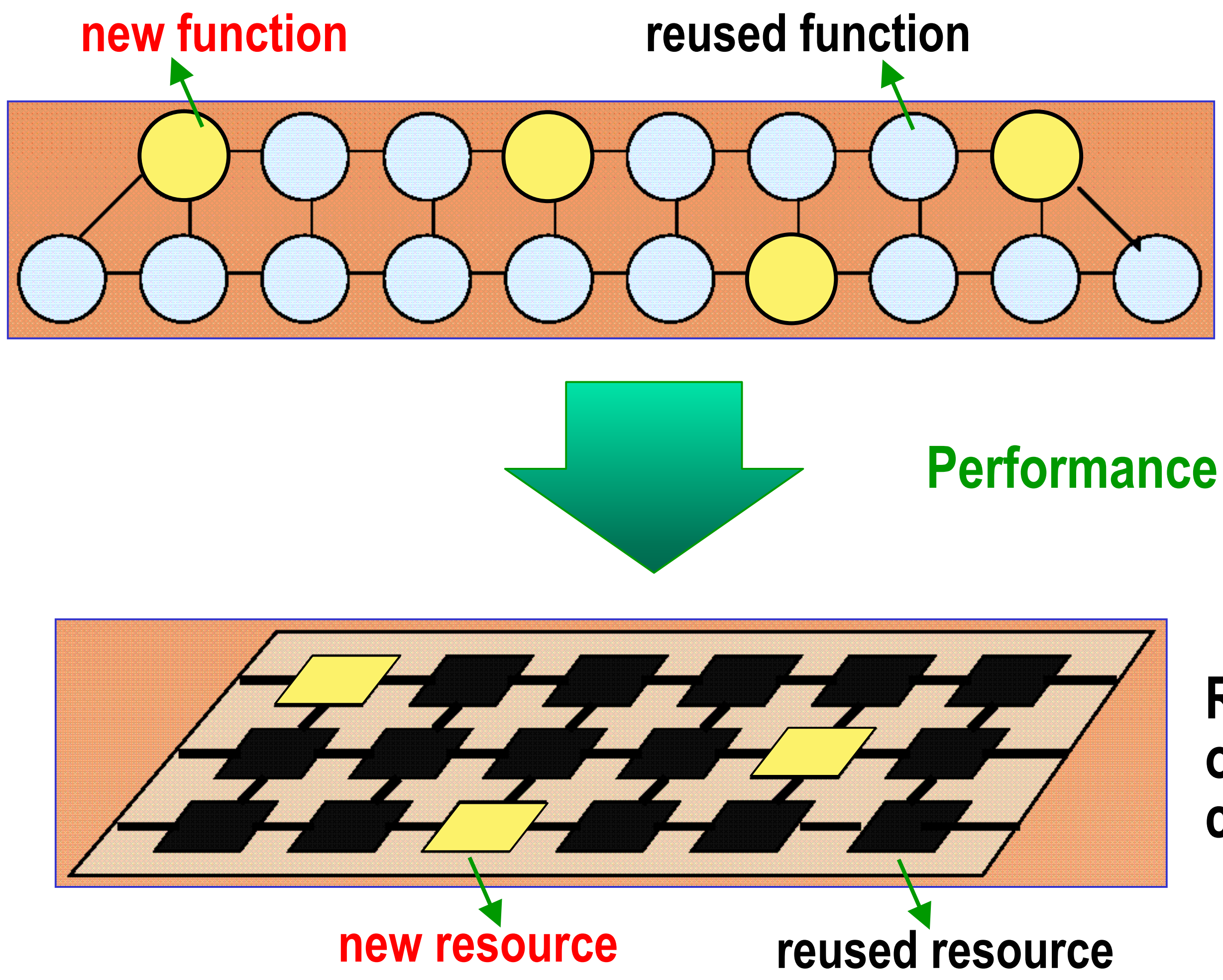


- Communication infra-structure
  - Physical layer
  - Data link layer
  - Network layer
  - Transport layer
- Topology
- Type and size of resources (Micro processor, DSP, memory, FPGA, etc.)
- Resource usage

On-chip Network Topology Considerations

- On-chip networks are constrained by on-chip bisection bandwidth, or the number of available communication channels on the chip
  - Designers must design for the optimum network (in terms of latency, throughput, and path diversity) by trading channel bandwidth (area devoted to communication channels) for processing grain size (area devoted to the processing tiles)
- 7 measures of quality for a network topology:
  - Load balance between the network and the processing unit, and among processing units (tiles)
  - Its topology regularity
  - Spatial locality
  - Its ability to map to a 2-D layout
  - Low power consumption
  - Its fault tolerance
  - Base measures of a network's quality: latency, throughput, and path diversity

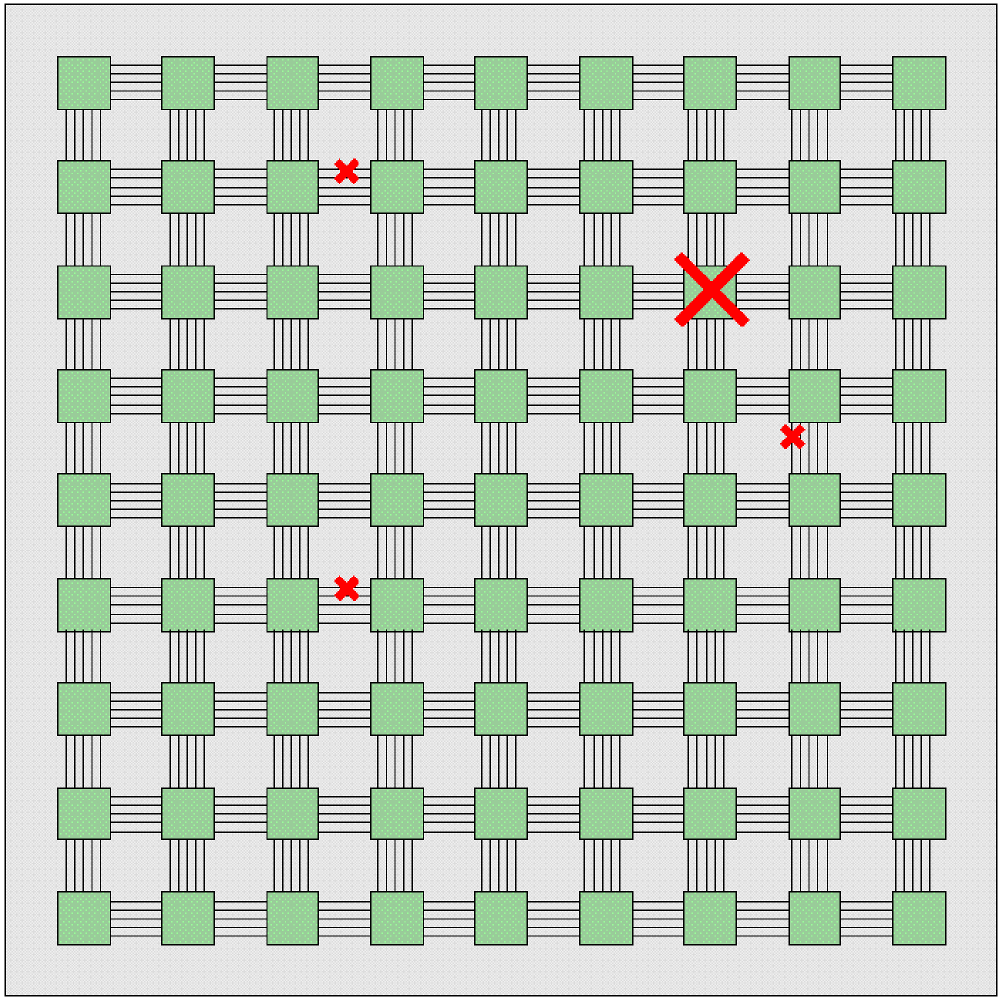
Design Methodology Challenges



Specification techniques of concurrent activities

Reuse and integration of both functions and components

Run Time Services Challenges



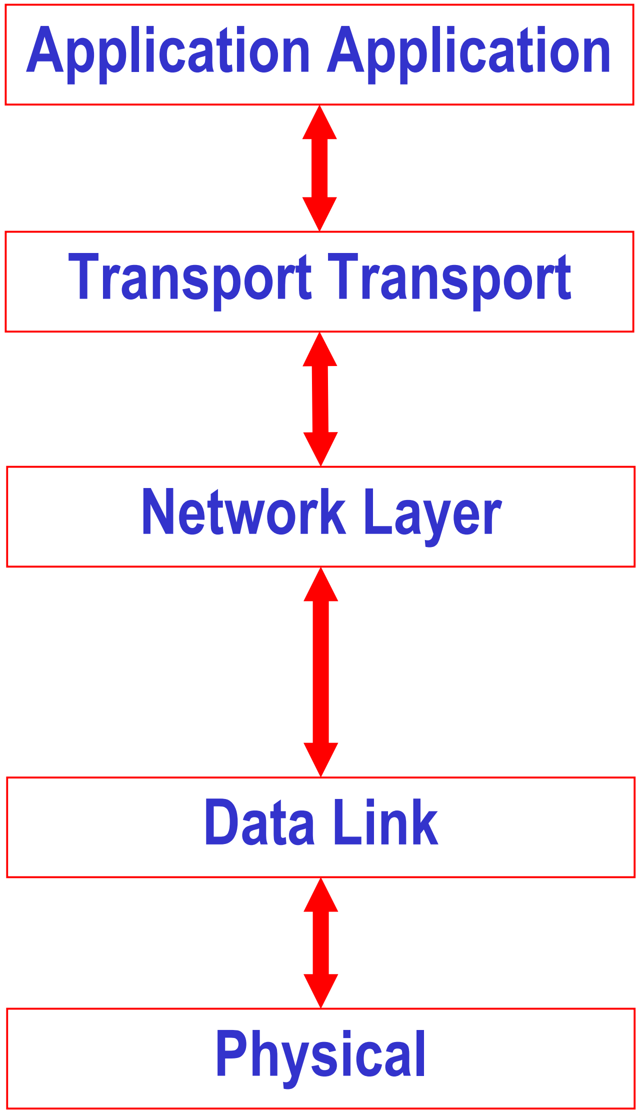
- Monitoring
- Fault-tolerance,
- Diagnostics
- Fault recovery
- Dynamic resource management

Configurability Challenges

A sensible trade-off between efficiency and generality is critical

- Configurability of communication resources from the data link to the application layer
- Configurability of resources (processors, DSPs, FPGAs, etc.)
- When and who?
  - Design-time configuration: Platform $\Rightarrow$ Product
  - Static product configuration: Once for a product
  - Dynamic reconfiguration: Programming of the product
- Network reconfigurability will be key in providing plug-and-play component use
- Programmable
  - Programmable architectures are an enabling technology for On-Chip Networks
  - Many communication stack features can be implemented in software
  - System can be reprogrammed for a different application and a different application model

Ideas from Networking and VLSI



- Application model independence
  - MPI*
  - Shared Memory
  - Other models of computation

programming

- Prioritization of communication events
- Usage statistics
- Protocol semantics

- Routing data across an arbitrary interconnect topology
- Implementation independence
- Hierarchical buses
- Dynamic connections

VLSI design

- Power savings: pulse shaping
- Channel coding: error control for unreliable channels

- Source coding: data compression
- Reliability: CRC, ECC*

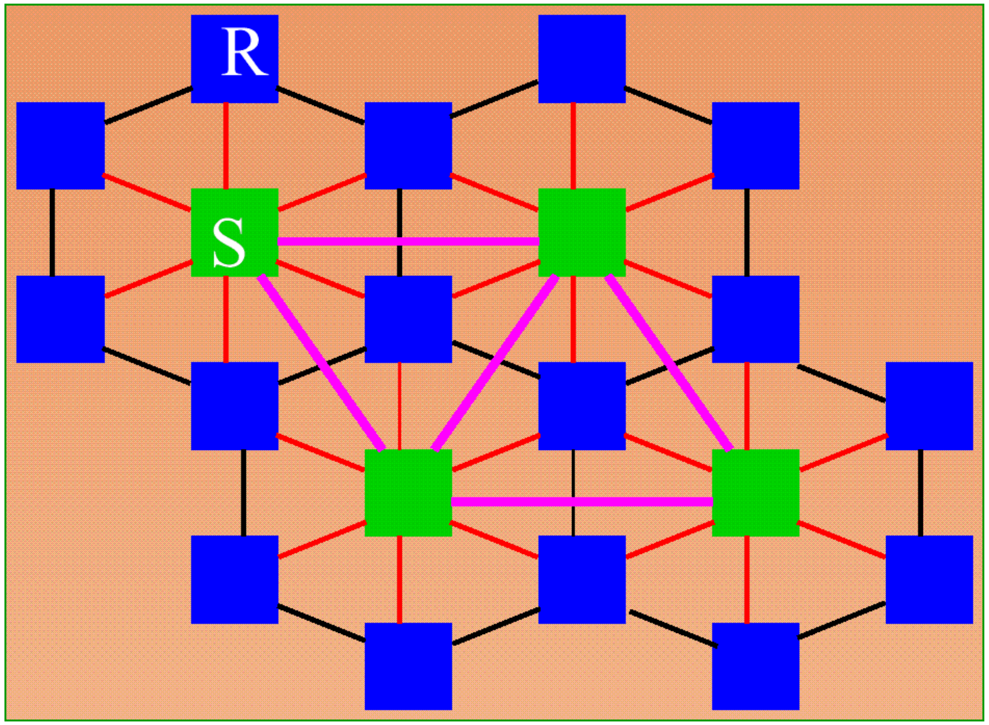
NoC Design Factors

- Architectures
  - Topology: honeycomb, meshes, butterfly, octagon, …
  - Shared medium
  - Direct/indirect: switch (or router) or not
  - Hybrid
  - Regions
- Physical Layer
- Data link layer
- Network layer
- Transport layer
- Application layers: NoC OS

Architecture for On-Chip Networks

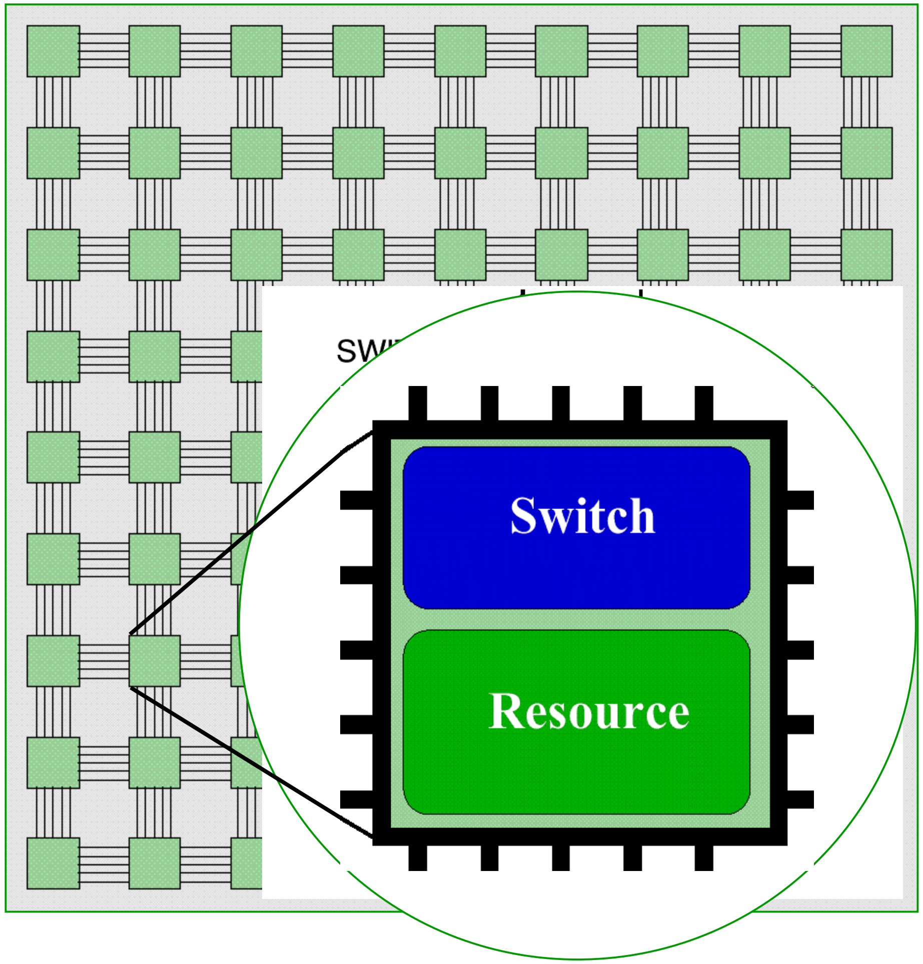
- Topology - different constraints than off-chip networks
  - buffering is expensive, bandwidth is cheap
  - more wires between ‘tiles’ than needed for one channel
    - multiple networks, higher dimensions, express channels
- Flow-control
  - flit-reservation flow control, pipeline control ahead of data
    - latency comparable to statically scheduled networks
    - minimum buffering requirements
  - run static, statically scheduled, and dynamic networks on one set of wires
- Interface Design
  - standard interface from modules to network
    - pinout and protocol
    - independent of network implementation

Topology: Honeycomb



- Resource-to-switch ratio: 3
- A switch is connected to 6 resources and 6 switches
- A resource is connected to 3 switches and 3 resources
- Wiring intense topology
- Max number of hops grows with n/3

Topology: Mesh*



- Resource-to-switch ratio: 1
- A switch is connected to 4 switches and 1 resource
- A resource is connected to 1 switch
- Max number of hops grows with 2n

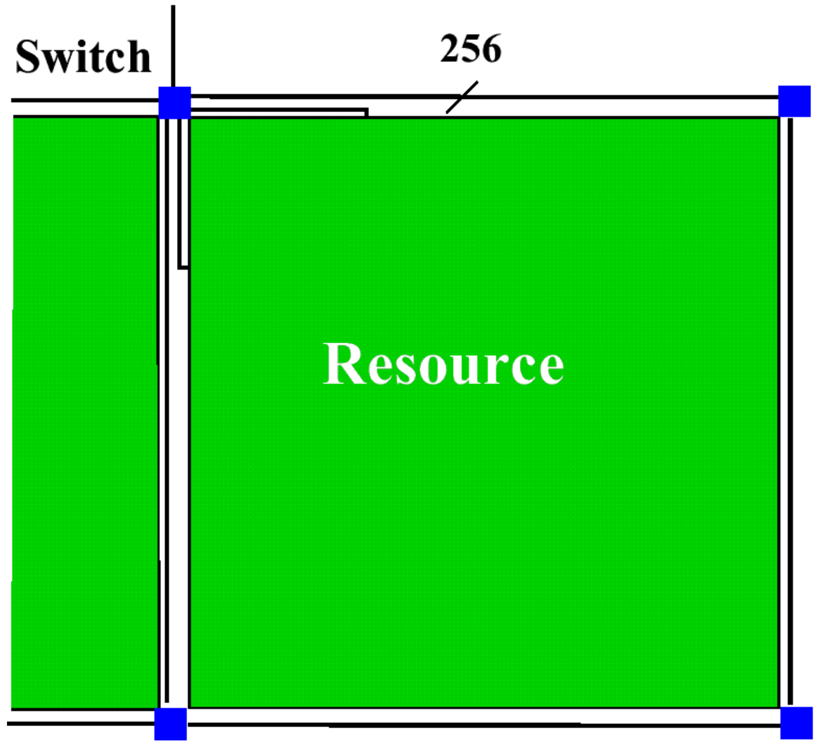
rni : resource network interface, 🟩

The node in a mesh

- How large are resources and the switches?
- What is the best geometry of switch and resource?
- How many wires and how long?

Square Switch

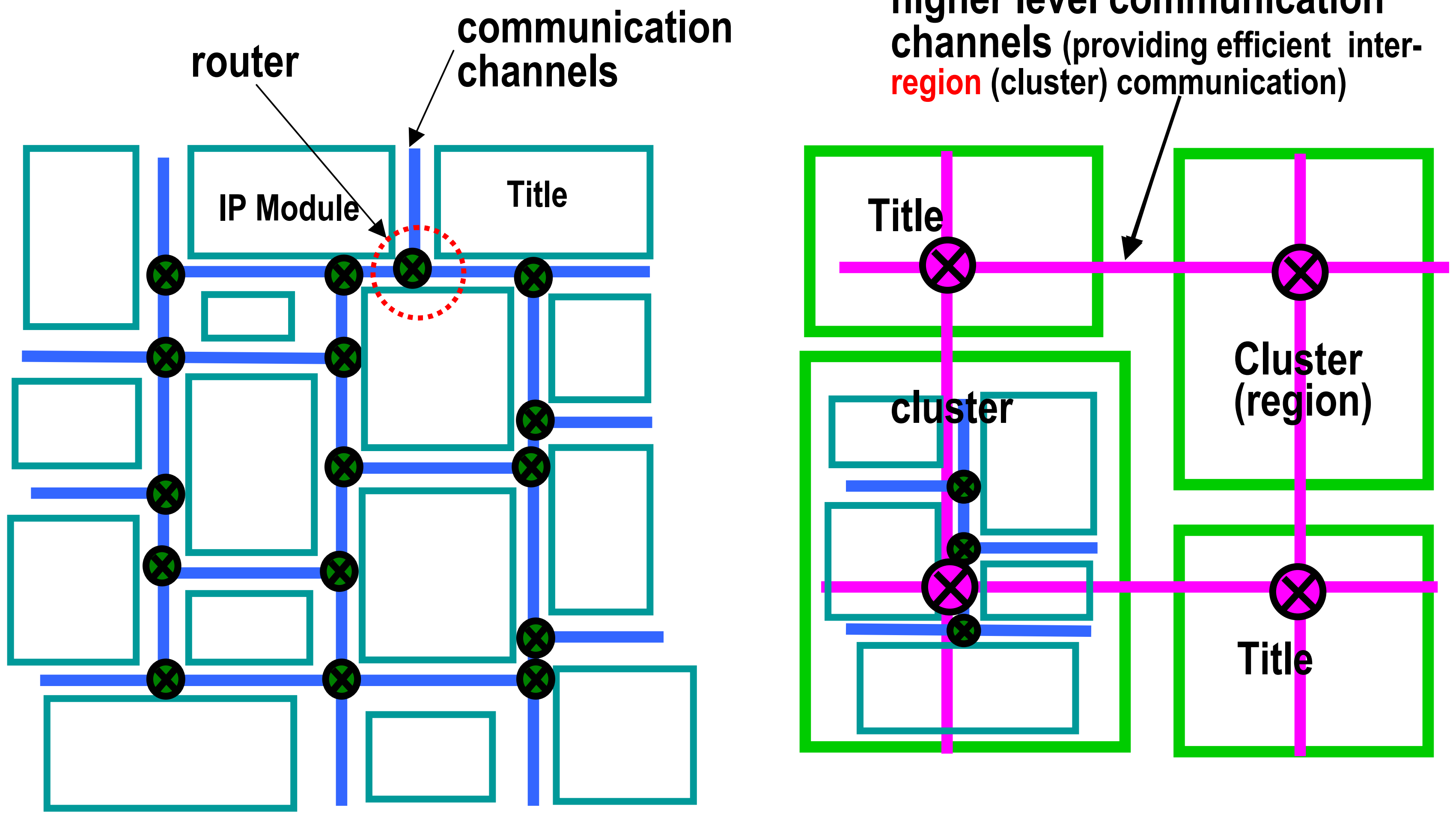
- 2 distinct switch layouts, called “thin switch” and “square switch”
  - Thin switch: the functionality is distributed around a resource and wires are routed across the resources, and is superior with respect to layout area.
  - Square switch is placed on the crossings in dedicated channels left between resources: superior with respect to performance and bandwidth



256 switch scenario:

- 60nm CMOS
- 22mm × 22mm chip size
- 300nm minimal wire pitch
- 2mm × 2mm resource
- 100μm × 100 μm switch
- switch-to-switch connection: 256 wires
- switch-to-resource connection: 256 wires

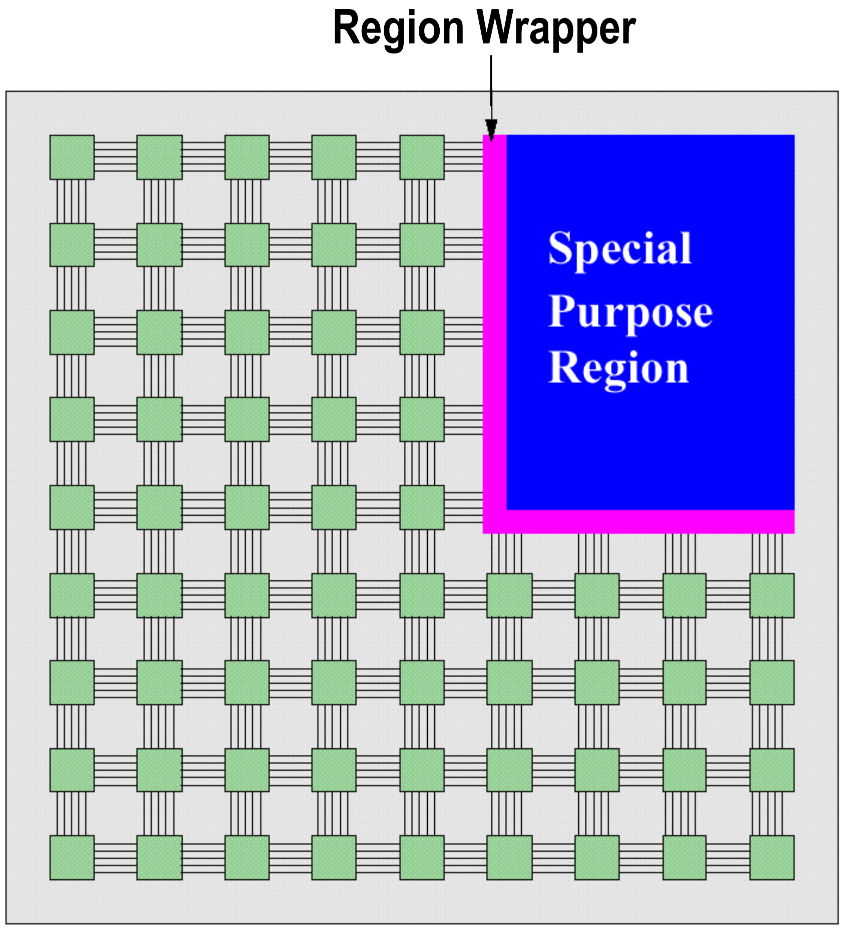
Generalized and Hierarchical Mesh


router
communication channels
IP Module
Title
higher level communication channels (providing efficient inter-region (cluster) communication)
Title
cluster
Cluster (region)
Title

A NoC with heterogeneous IP cores

Regions

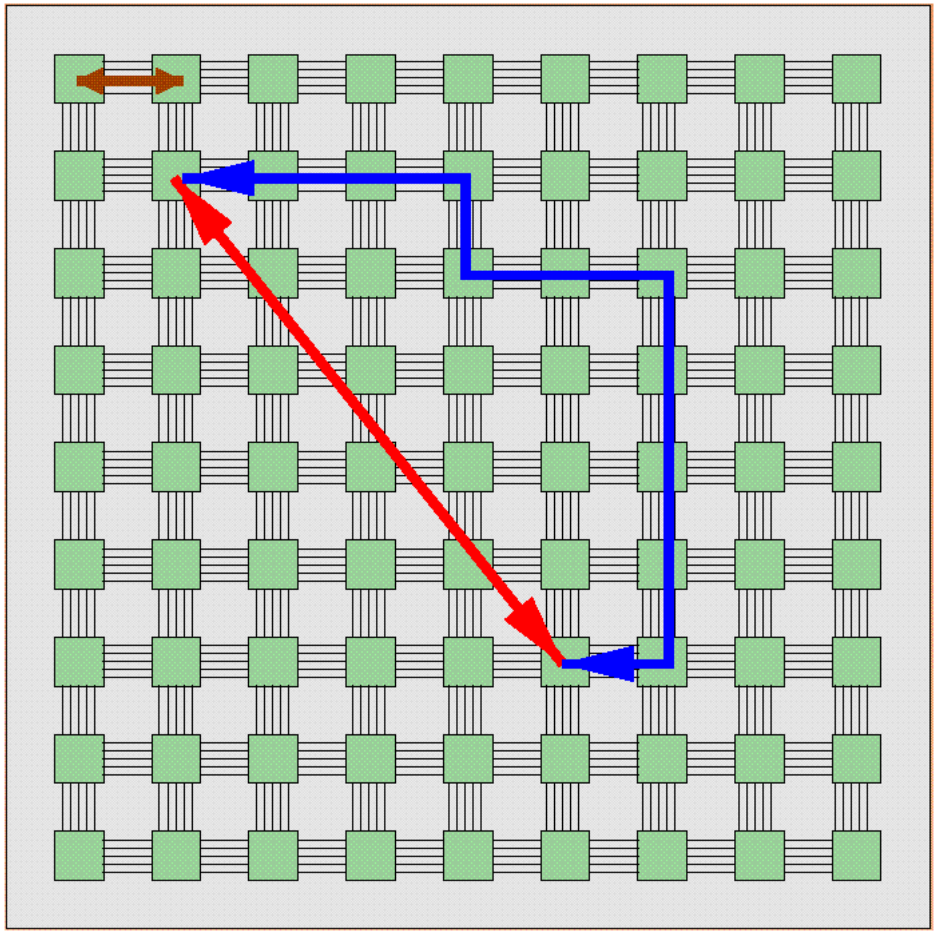
A region: covers an arbitrary number of switches and resources and allows to accommodate larger specific and heterogeneous components or resources such as FPGA areas and memory



Parameters:

- Not all resources have the same size; e.g.
  - Memory
  - FPGA regions
  - Special purpose architectures like multiprocessors and streaming-oriented data processing
  - Mixed signal parts
- A region wrapper makes the region transparent
  - It maintains the illusion of an undisturbed network to the outside by (for instance routing packets around rather than through the region if the packet's destination is on the other side)
- The region wrapper can be
  - at several protocol layers,
  - in hardware or software
  - local or distributed

Communication is Key on Several Levels

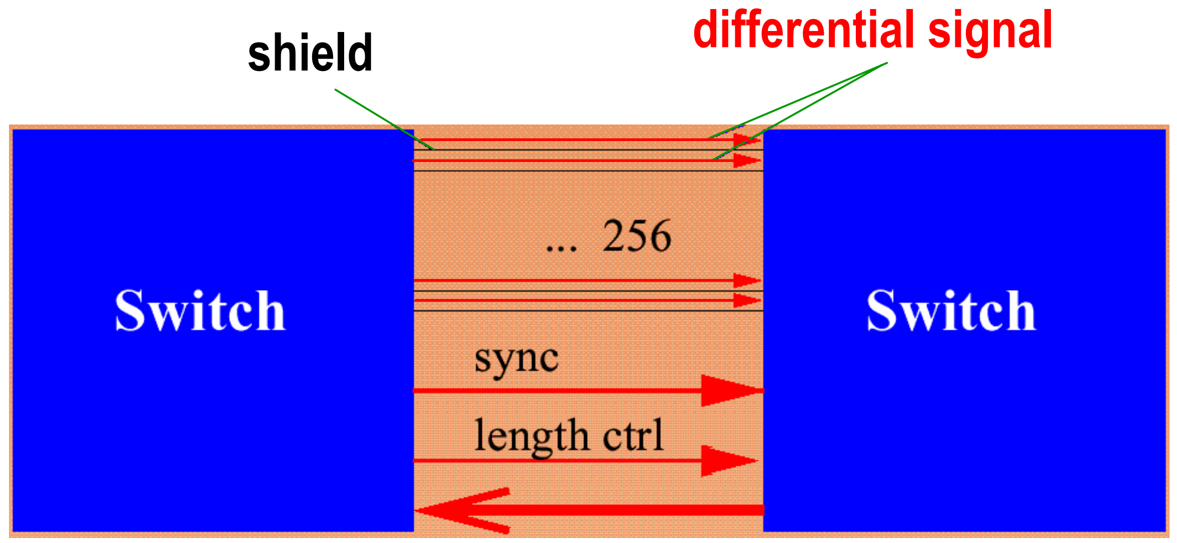


Communication Layers and units of communication:

- Physical layer: Word
- Data link layer: Cell
- Network layer: Packet
- Transport layer: Message
- Application layer
  - System software
  - Application software

Physical Layer

Determines the number and length of wires connecting resources and switches.



2 x 300 x 4 = 2400 wires

Parameters:

- Physical distance
- Number of lines, sizing
- Voltage levels
- Driver design
- Physical routing
- Activity control
- Buffers and pipelining

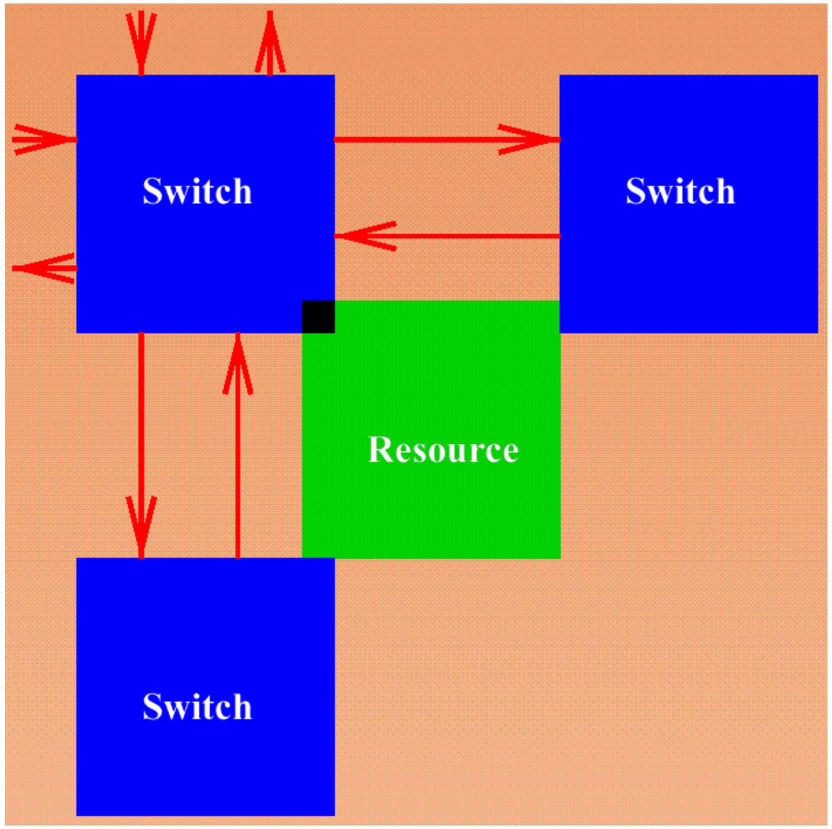
dependent on the technology

Physical Layer Phenomenon

- Limitations come from interconnect physics
  - RLC effects on wires -> propagation time
  - Delay on global wires and delay uncertainty
- Electric signaling
  - Trade-off noise immunity vs. energy vs. speed
  - Small swings -> low energy and fast transitions
- Challenge: synchronization across large chips
  - Is synchronization possible at high clock rates?
  - What is the probability of synchronization failure?
- Reliability of information
  - Information transfer is inherently unreliable
  - Major causes:
    - Cross-talk
    - Electro-magnetic interference (EMI)
    - Timing errors
    - Soft errors
  - The problem will get more and more acute as technology scales down

Data Link Layer

Defines the protocol to transmit a *cell* between a resource and a switch and between two switches



Parameters:

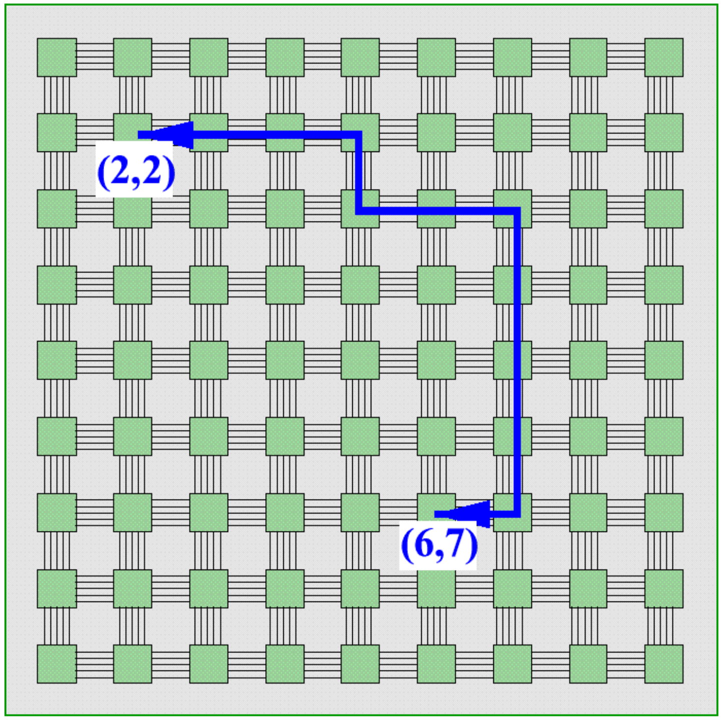
- Line frequency versus switch frequency (word vs cell)
- Buffering
- Error correction
  - Due to contention (its resolution is by synchronization)
  - Uses packetizing and error-correcting codes
- Power optimization; e.g. avoid activity and power optimized (bus) encoding

Advanced Bus Techniques*

- CDMA on bus
  - Support multiple concurrent write on bus and against noise
  - Challenge: electrical design of drivers using charge pumps
- Split transactions
  - 128 bit split-transaction bus:
    - Many simultaneous transaction requests
      - Varying size, some large
      - Varying priority
      - Transactions IDs associated with transactions
    - Goal: minimize average latency
  - Round-robin arbitration
- Error-resilient coding for deep submicon effects of cross-talk and EMI

Network Layer

Defines how a *packet* is transmitted from an arbitrary sender to an arbitrary receiver directed by the receiver's network address -- end-to-end delivery control with many communication channels.



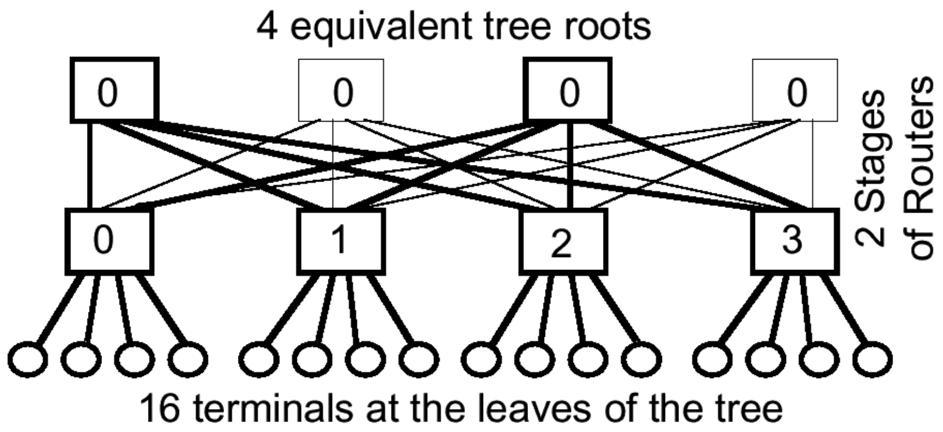
Parameters:

- Cell size versus packet size
- Network address scheme, e.g. 4 + 4 bit for 16 × 16 resources
- Switching: circuit, packet, cut-through
- Routing algorithm
  - Deterministic and adaptive routing
  - Decentralization of routing decisions, robustness, and fault tolerance
- Priority classes: e.g. 2 classes:
  - 1. high priority, fixed delay cells
  - 2. low priority, best effort delay cells
- Error correction
- Trade-off: delivery time and channel utilization vs. predictability
- Choice depends on traffic regularity and on availability of computing nodes

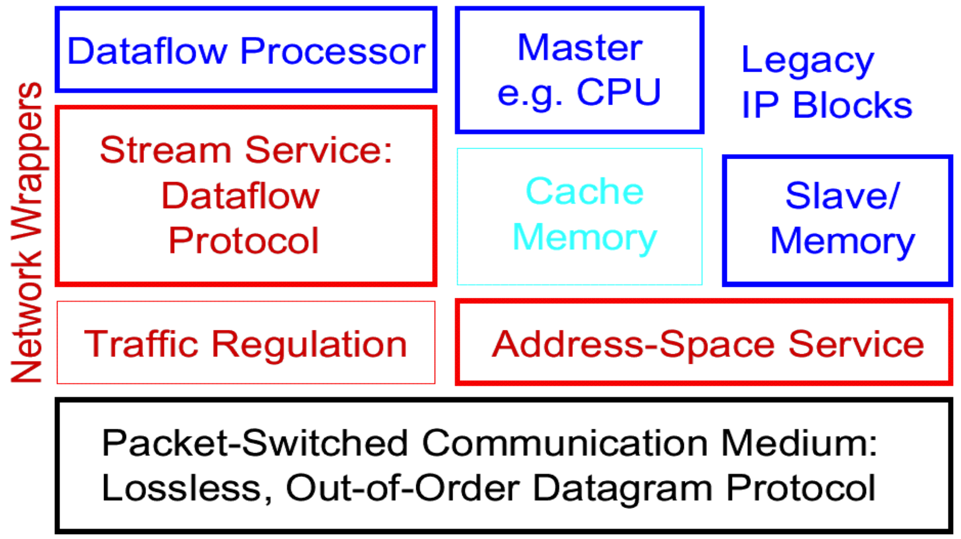
SPIN NoC*

A Scalable, Programmable, Integrated Network (SPIN) on a chip

- 32-bit packets
  - header: destination
  - trailer: checksum
- Fat-tree network architecture
- Cut-through switching
- Deterministic routing
  - Tree routing (for fat tree)



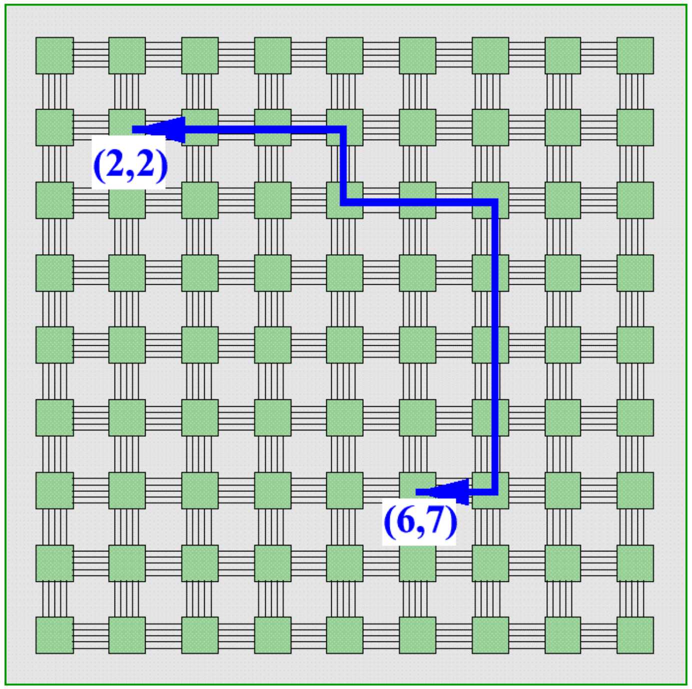
A fat-tree network



Hardwired protocol stack

* P. Guerrier and A. Grenier, "A Generic Architecture for On-Chip Packet-Switched Interconnections," *Proc.* DATE 2000, pp. 250-256.

Transport Layer



Parameters:

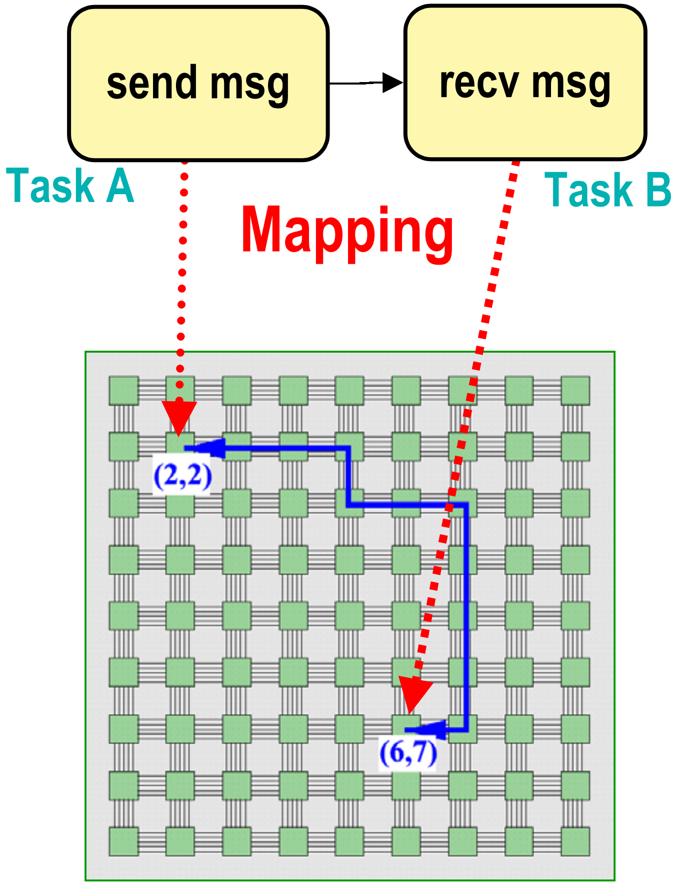
- Message size versus cell size: granularity
- Virtual channels with traffic profiles
- Signaling
- Priority classes of channels, e.g.
  - 1. constant bit rate traffic
  - 2. varying bit rate traffic
- Network resource management
  - Admission/congestion
- Error correction

- RNI: all 4 layers towards the network. The switch-to-switch interfaces: only the 3 lower protocol layers
- It is possible to add additional protocols (without using message passing) on top of the transport layer to provide for instance a virtual shared memory abstraction,
  - For data and computation intensive application

Application Program Layer

- Performance and energy to realize a function on a platform depends on software
  - Different algorithms to embody a function (e.g., sorting)
  - Different coding styles
  - Different instruction streams (e.g. assembly)
- Software production tools need to address both performance and energy goals
- Software synthesis
  - Source-code generation
  - Source-level optimizing transformations
- Application-specific compilation
  - Choice of instructions, registers, and schedule
- Software design tools need network awareness to be effective
  - Balance computation, storage and communication

Application (Software) Layers



Interprocess communication at the task level:

- send/receive for individual messages
- open; write/read; close for channel based communication

Mapping issues:

- Assigning tasks to resources
- Translating task addresses to resource/task addresses
- Establishing and closing channels
- Static allocation versus dynamic allocation

System software: NoC OS

- OS, RTOS, run-time scheduler
- Component and network dependent

Application program

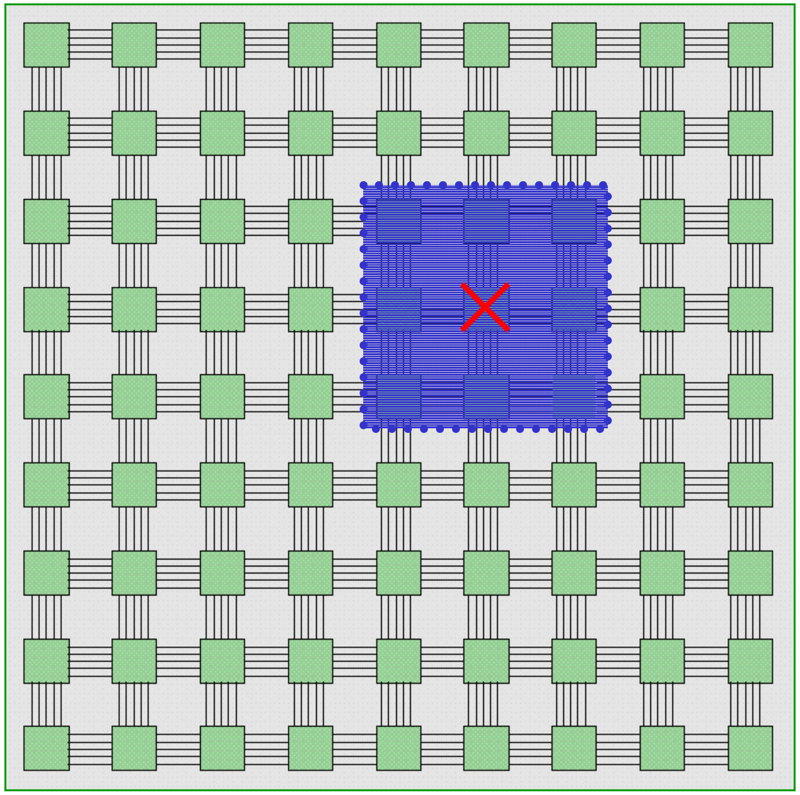
Network-aware System Software

- Lightweight, modular, distributed system
  - Supports component DPM* at different levels
  - Supports network re-configuration
  - Adapts to different modes of operation and environmental factors
- Challenges:
  - Distributed environment factors: scalability, re-configurability, robustness, ...

NoC Operating System

- Monitoring
  - Utilization of resources and switches
  - Power consumption
  - Statistics (errors, cells, etc.)
- Communication services (transport, presentation and application layers)
- Resource allocation and load migration
- Diagnostics and fault recovery
- Power management: dynamic
- Development support services
  - Libraries for run-time services
  - Compilers, linkers, and simulators
- A modular distributed System
  - Each programmable component will be provided with system software to support its own operation, manage its communication with the on-chip network, and interact with neighboring components' system software.
  - System software configuration: manual adaptation $\rightarrow$ automatic configuration
  - On-chip communication protocols are programmable at the system software level to adapt the underlying layers to the components' characteristics.

Dynamic Fault Handling Example



Assumptions:

- NoC-OS makes run-time diagnostics
- NoC-OS can identify faulty parts
- Network layer protocol can be reconfigured at run-time

If a resource becomes faulty:

- 1. NoC-OS detects a faulty resource
- 2. NoC-OS defines a new region to isolate the faulty resource
- 3. NoC-OS reconfigures the Network layer protocol

Basic Requirements for NoC Design Methodology

- Reuse
  - of intellectual property blocks
    - best performance/energy ratio
    - best mapping to application characteristics
- Reuse
  - of hardware (and architecture) and NoC platform
    - best complexity/cost and performance/cost ratio
    - only way to even dream of achieving time-to-profit requirements
- Reuse
  - of design methods and tools
    - only way to deal with heterogenuous application set
- Partitioning of problems
  - by encapsulation and hiding of the complexity of the overall system

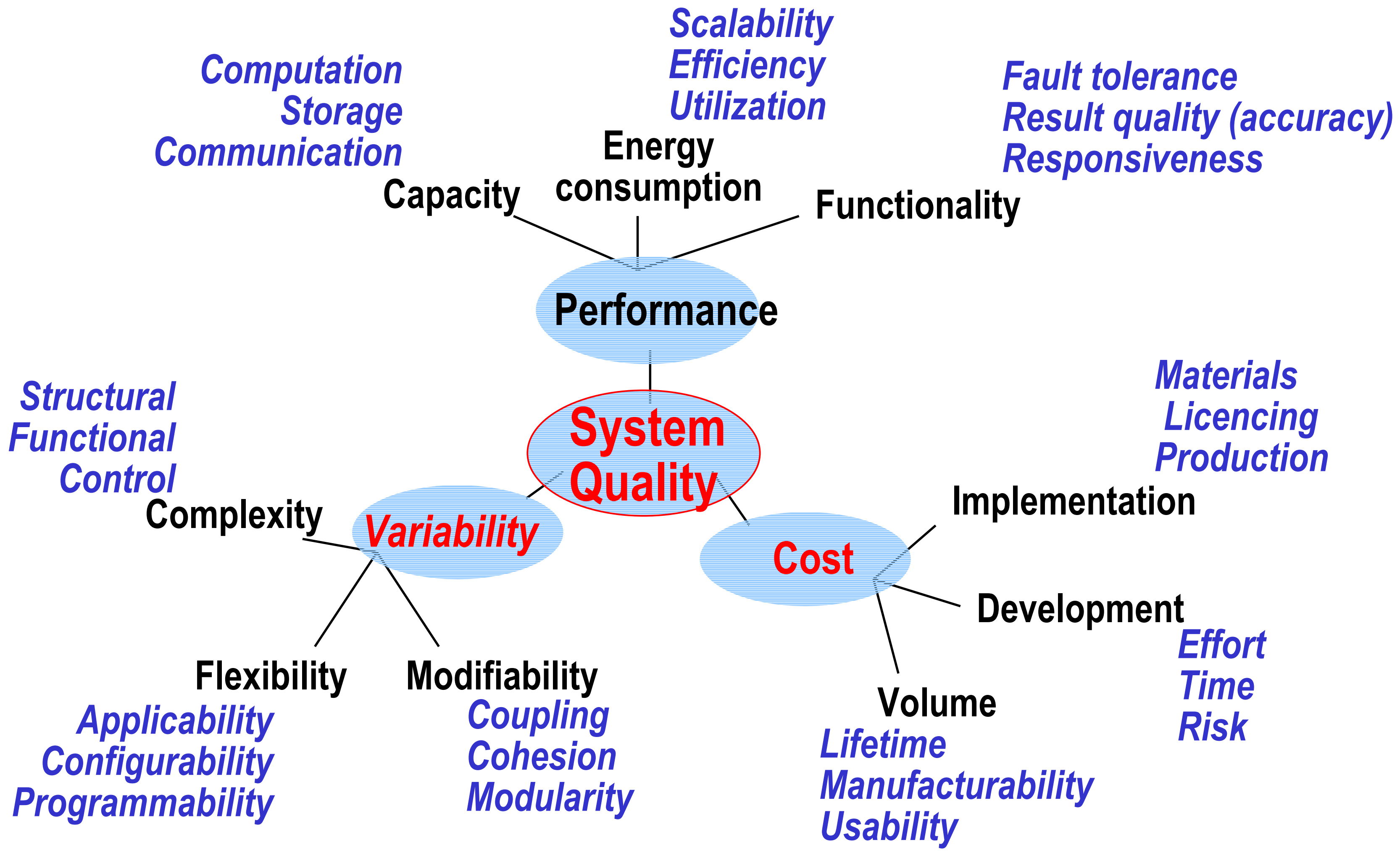
NoC Design Methodology

- NoC design methodology consists of 2 phases:
  - 1-st phase: a concrete architecture is derived from the general NoC template
    - The concrete architecture defines the number of switches and shape of the network, the kind and shape of *regions*, and the number and kind of resources
  - 2-nd phase: it maps the application onto the concrete architecture to form a concrete product
- Design methodology must support product family management:
  - Tolerance of incomplete specifications
  - Management of configurations and modifications
  - Supporting for multiple languages and methods
  - Capability to handle different abstraction levels and the network protocol stack model

NoC Platform Development

- Scaling problem
  - How big NoC is needed? What are the application area requirements?
- Region definition problem
  - What kind of regions are needed? What kind of interfaces between regions? What are the capacity requirements for the regions?
- Resource design problem
  - What is needed inside resources? Internal computation type and internal communication?
- Application mapping flow problem
  - What kind of languages, models and tools must be supported? How to validate and test the final products?

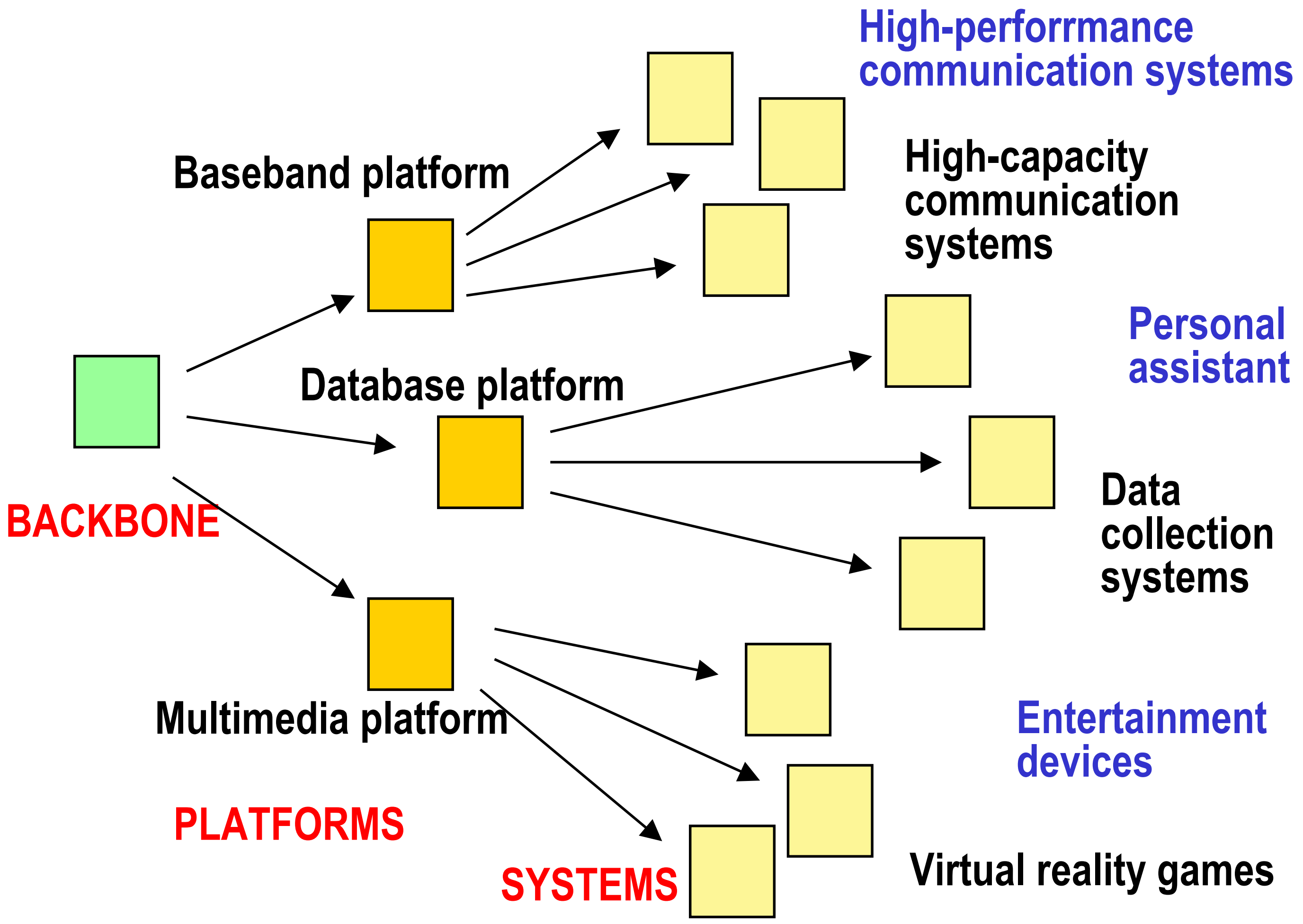
Figure of Merit for NoC based Systems



NoC Application Development

- Mapping problem
  - How to partition applications for NoC resources? How to allocate functionality effectively? Is the performance adequate? Is the resource usage in balance?
- Optimisation problem
  - How to perform global optimisation of heterogenuous applications? How to define right optimisation targets? How to utilise application/resource type specific tools?
- Validation problem
  - Are the contraints met? Are the communication bottlenecks or power consumption hot spots? How to simulate 10000 GIPS system? How to test all applications?

Development of NoC based Systems



NoC Based System Design - I

- 1. Backbone design
  - A development platform for all NoC based systems
  - It encapsulates topological and communication issues such as channels, switches, and network interfaces
  - Its design focus is the network communication resources, e.g. switches and interfaces, and NoC system services and performance of different region topologies
  - Definition of *region* requires that potential applications are analyzed and modeled

NoC Based System Design - II

- 2. Platform design: creates a computation platform for an intended application area.
  - Main activities: scaling of the network, definition of regions, design of the resource nodes, and definition of the system control.
  - First, understand the target systems' functionality thoroughly
  - Do not use exact applications as a starting point for architecture requirement definition due to the platform nature.
  - Use of optimized virtual components and knowledge of application-area requirements
  - Characterizes application area domain and architecture, and estimates system quality

NoC Based System Design - III

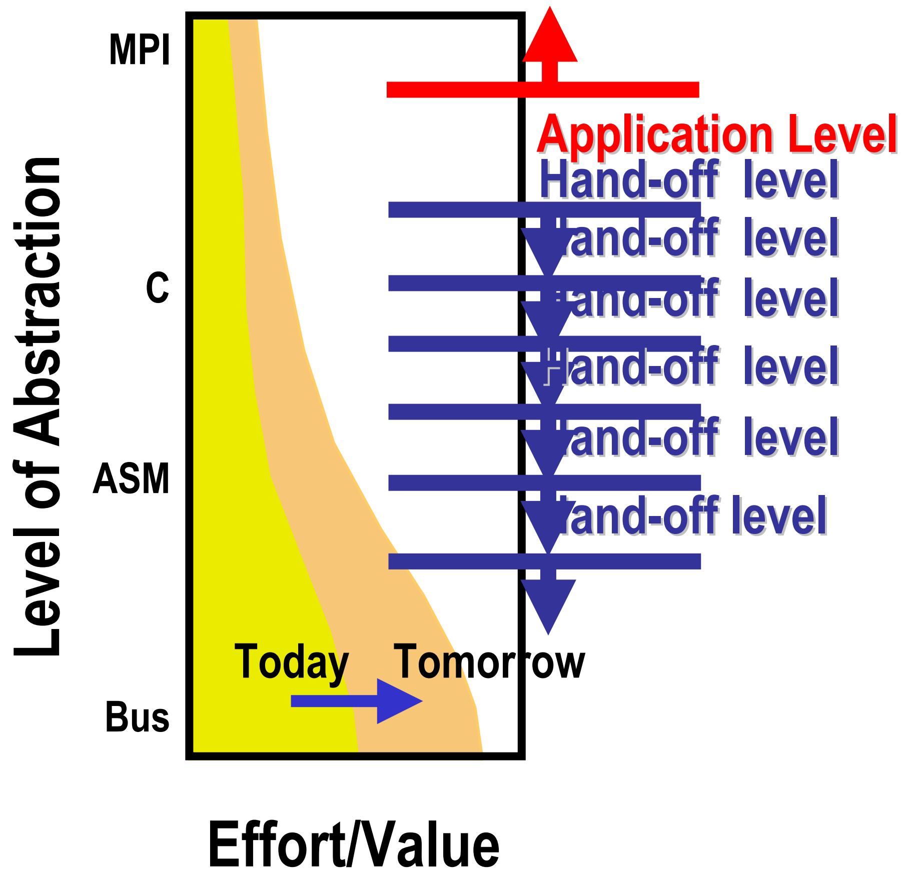
- 3. System design
  - Application mapping: application functionalities are mapped to the resources.
    - Must support both dynamic and static mapping
      - Main problems: are the resource allocation, optimization of network usage and verification of performance and correctness.
      - Several modeling languages should be supported
  - Main tasks: what to put into the NoC as resources, how to map functionality into those resources (resource selection), and how to validate the decisions.
    - Mappability of algorithms and architectures
    - Analysis of network behavior is a critical part
  - The *development and verification environments* provides a virtual machine and development environment for software development, and tools for hardware design

Summary of NoC Systems Design

Instance	Responsibilities during design
Backbone development	Region (specific resource) types Communication channels and switches Network interfaces of resources Communication protocols (specification)
Platform development	Region scaling Resource design (units, interconnections) Dedicated hardware blocks System level control (implementation of communication, diagnostics, monitoring)
Application development	Resource level control (OS) Functionality of resources (SW, configurable HW) Control of the network Functionality of the network

NoC Extending the Stack Model

- Grow stack to support message passing semantics


Level of Abstraction
MPI
C
ASM
Bus
Today
Tomorrow
Effort/Value
Application Level
Hand-off level
Hand-off level
Hand-off level
Hand-off level
Hand-off level
Hand-off level