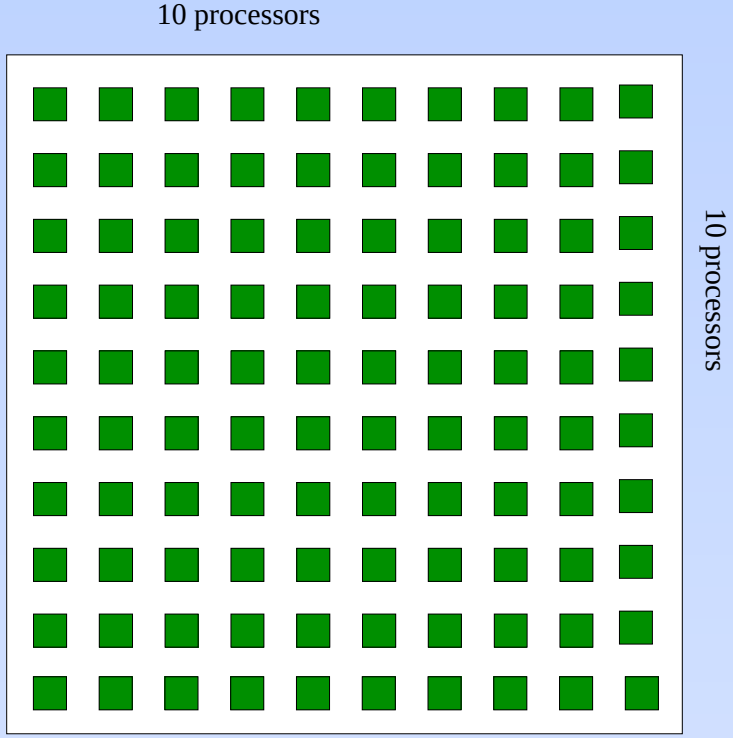
Networks on Chip



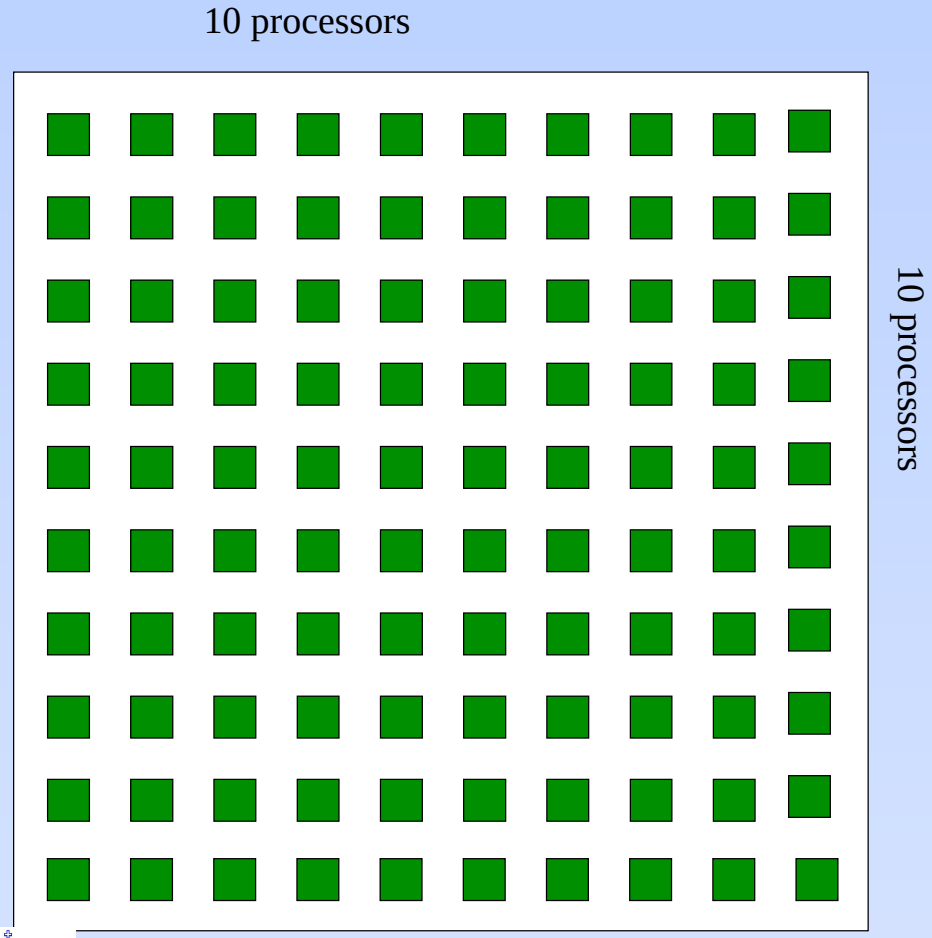
Axel Jantsch
Royal Institute of Technology, Stockholm

December 11, 2002

Overview

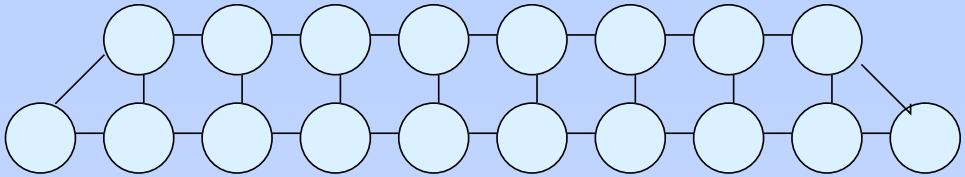
- Introduction
- NoC Architecture
- NoC Design Methodology

The Challenge

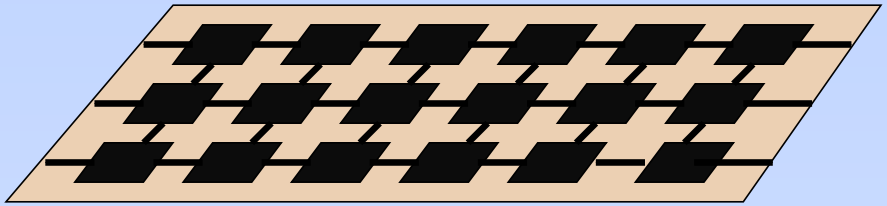


# Gates	# Processors	Year
6 M	4	2000
24 M	16	2003
96 M	64	2006
384 M	256	2009

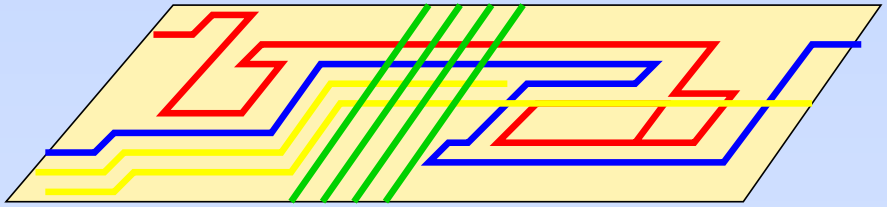
Functions, Architecture, and Physics



Concurrent processes



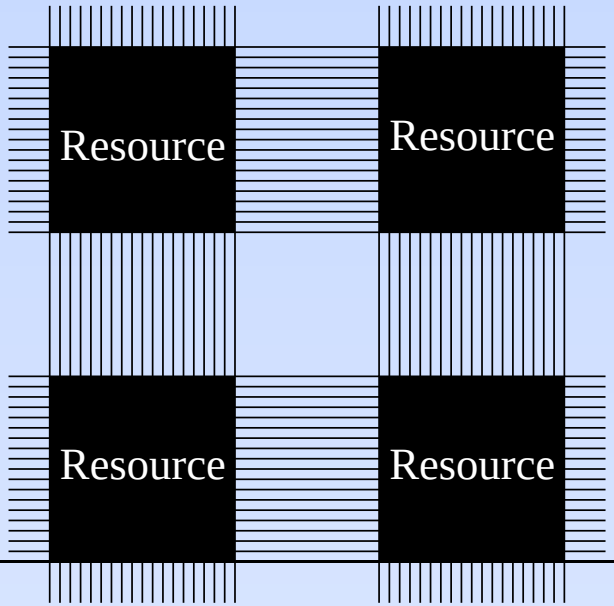
Large number of resoucres



Physical issues

Challenge Areas: Physical Issues

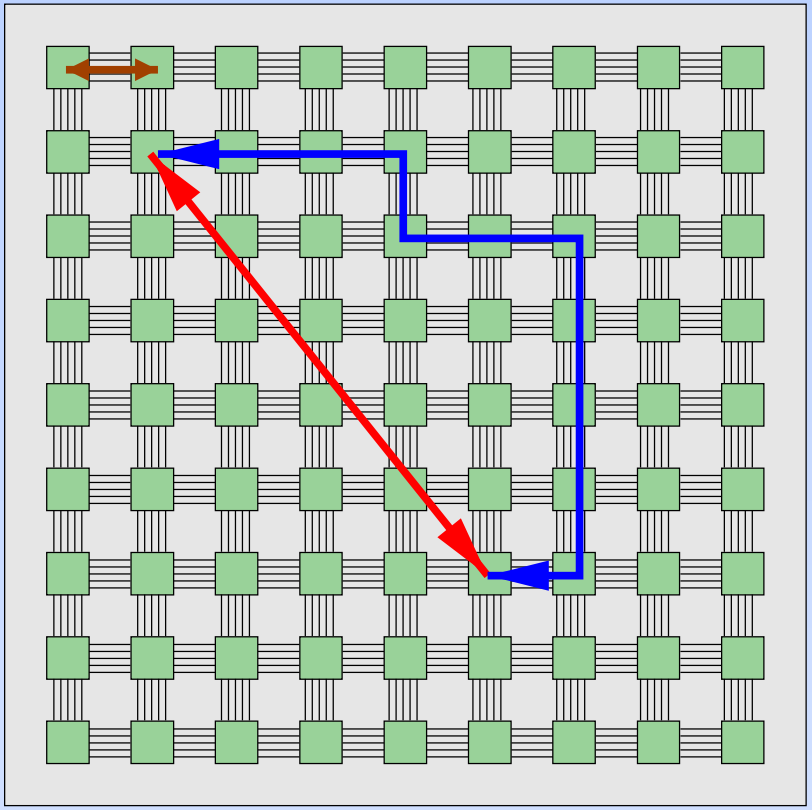
- Deep submicron effects, noise, signal integrity
- Interconnect
- Power consumption, power delivery
- Clock distribution
- Memory integration (50-80% of the chip)



Scenario:

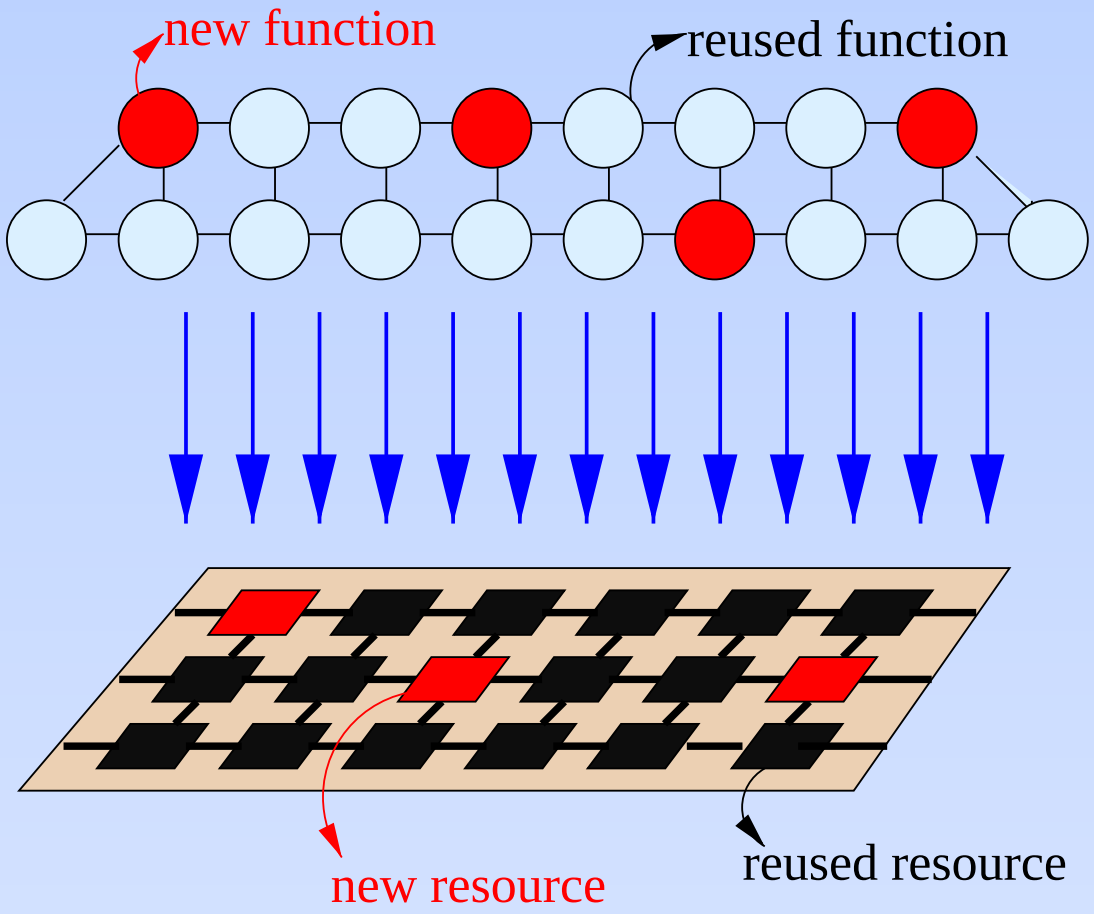
- 60 nm CMOS
- 22×22 mm Chip size
- 2×2 mm resource size
- 300 nm minumum wire pitch
- 6600 wires between two resources on each metal layer

Challenge Areas: Architecture



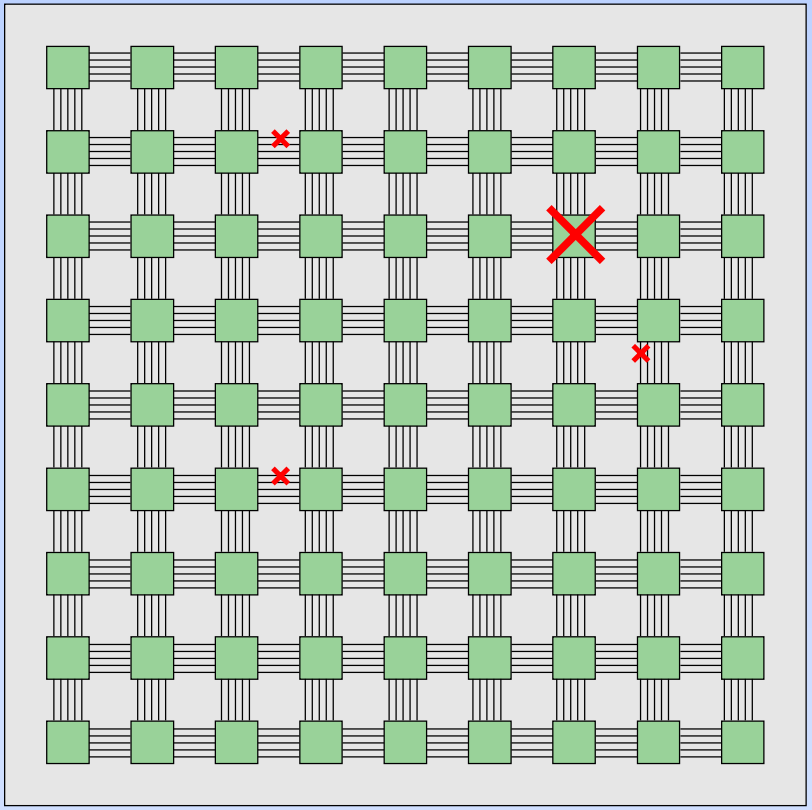
- Communication infra-structure
  - ⋆ Physical layer
  - ⋆ Data link layer
  - ⋆ Network layer
  - ⋆ Transport layer
- Type and size of resources (Micro processor, DSP, memory, FPGA, etc.)
- Resource usage

Challenge Areas: Methodology



- Specification techniques of concurrent activities
- Performance analysis
- Reuse and Integration of both functions and components

Challenge Areas: Run Time Services



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

Challenge Areas: Configurability

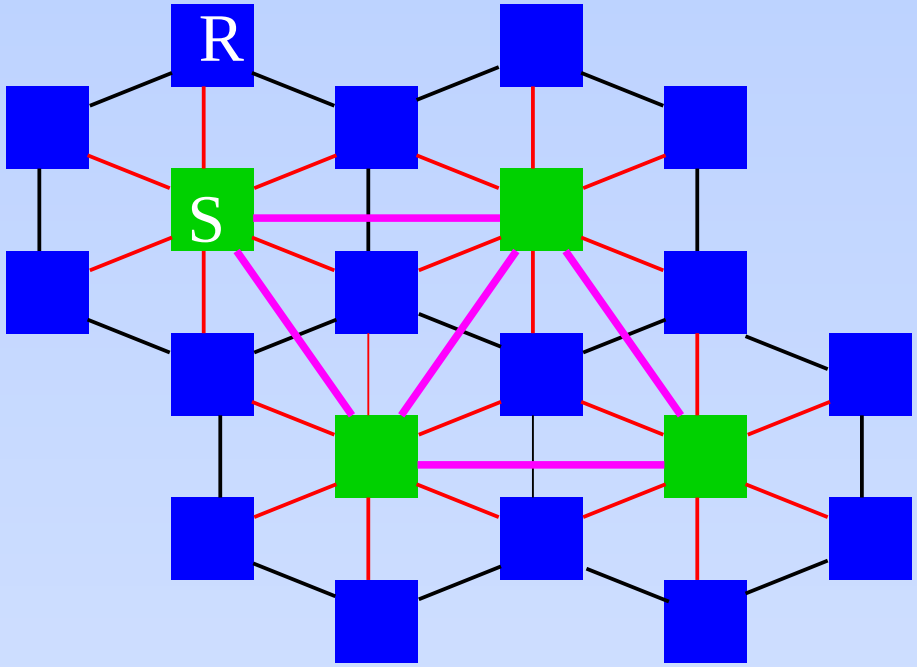
A sensible trade-off between efficiency and generality is critical.

- 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 ⇒ Pruduct
  - ⋆ Static product configuration: Once for a product
  - ⋆ Dynamic reconfiguration: Programming of the product

NoC Architecture

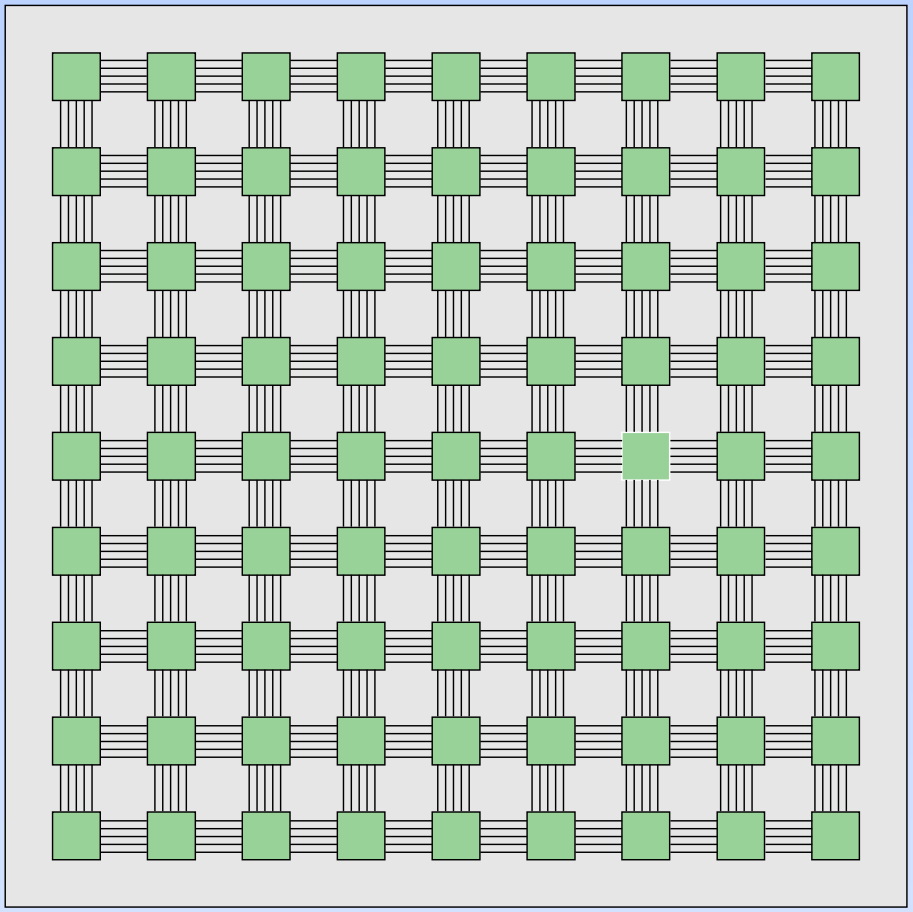
- Topology
- Switch Architecture
- Data link layer
- Network layer
- Transport layer
- Application layers
- Regions

Topology: Honeycomb



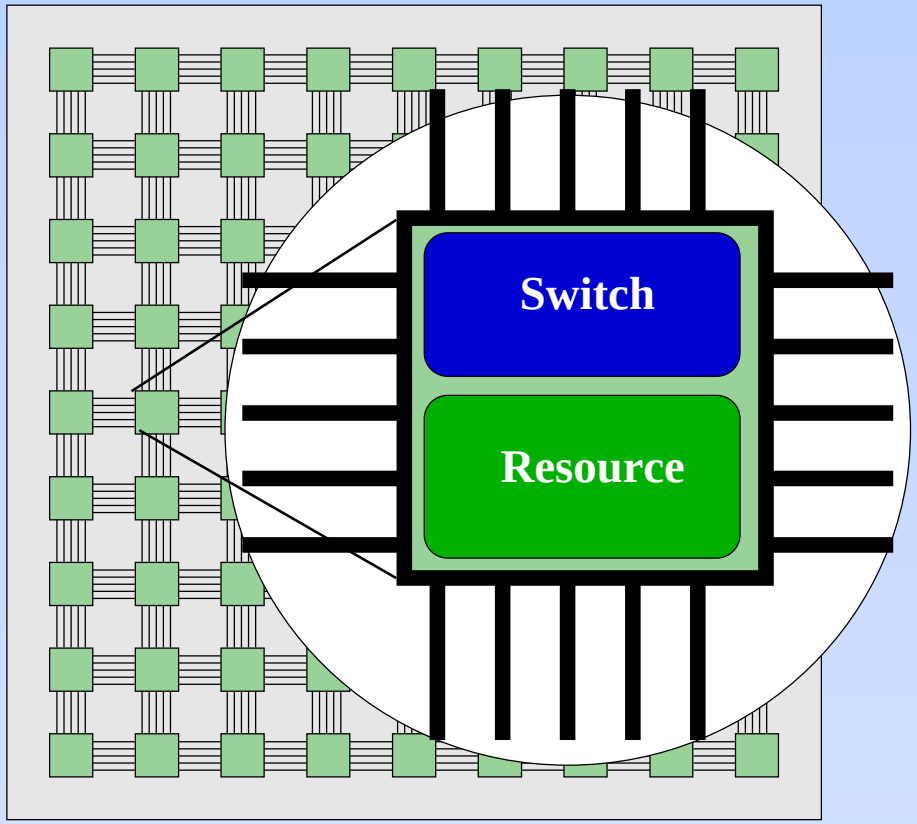
- Resource-to-switch ratio: 3
- A switch is connected to 6 resources and 6 switches
- A resource is connected to 3 switches and 3 resources
- Wiring intense topology
- Max number of hops grows with $n/3$

Topology: Mesh



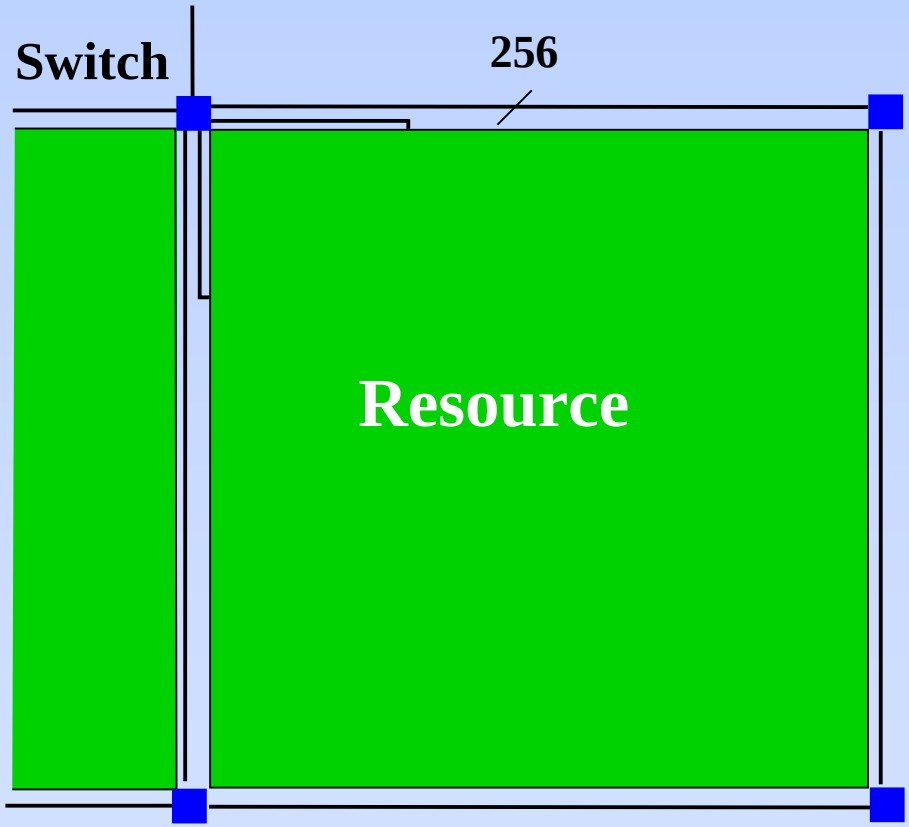
- Resource-to-switch ratio: 1
- A switch is connected to 4 switches and 1 resource
- A resource is connected to 1 switch
- Max number of hops grows with $2n$

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?

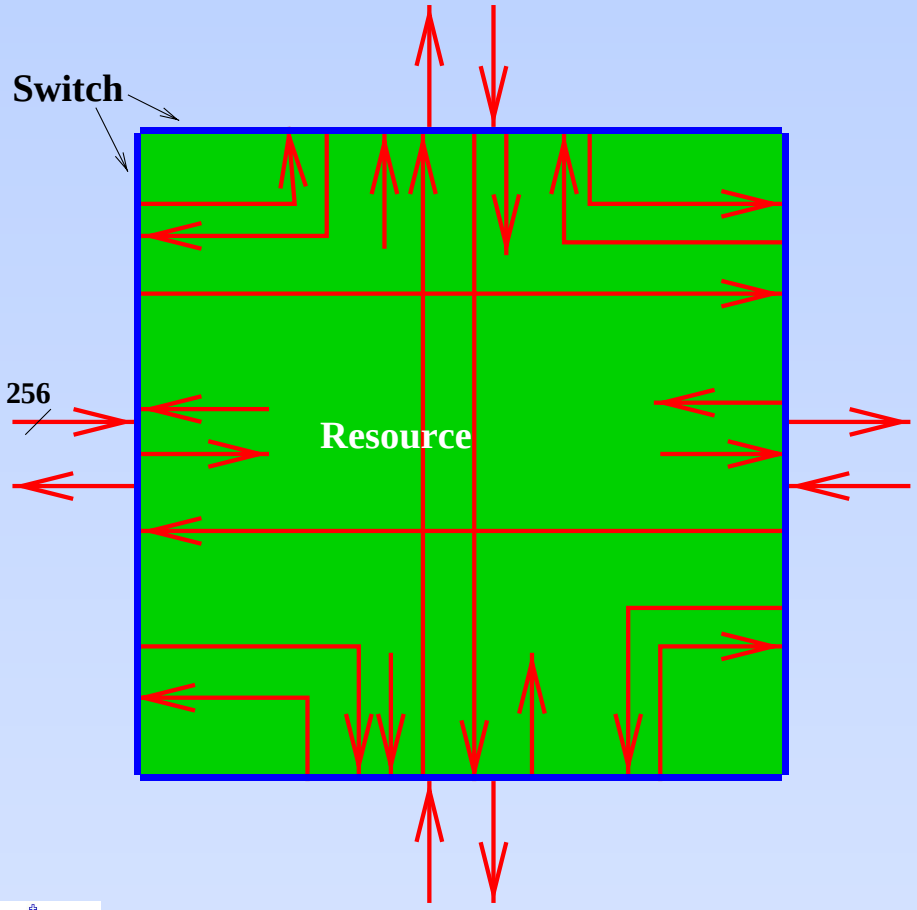
Square Switch



Scenario:

- $60nm$ CMOS
- $22mm \times 22mm$ chip size
- $300nm$ minimal wire pitch
- $2mm \times 2mm$ resource
- $100\mu m \times 100\mu m$ switch
- switch-to-switch connection: 256 wires
- switch-to-resource connection: 256 wires

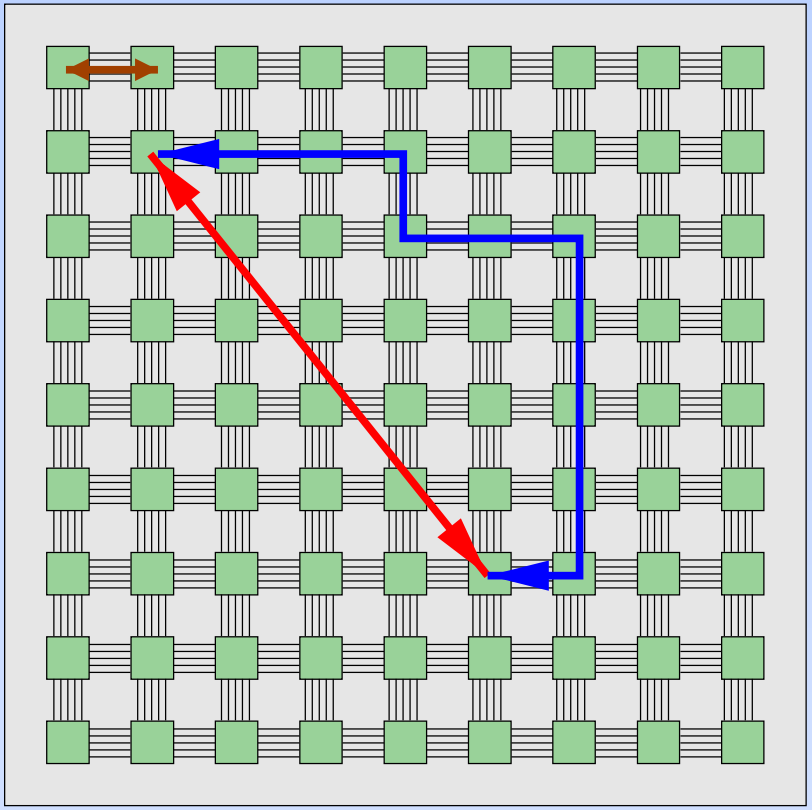
Thin Switch



Scenario:

- $60nm$ CMOS
- $22mm \times 22mm$ chip size
- $600nm$ minimal wire pitch for top layers
- $2mm \times 2mm$ resource
- $4 \times 50\mu m \times 2000\mu m$ switch
- switch-to-switch connection: 512 wires
- switch-to-resource connection: 512 wires

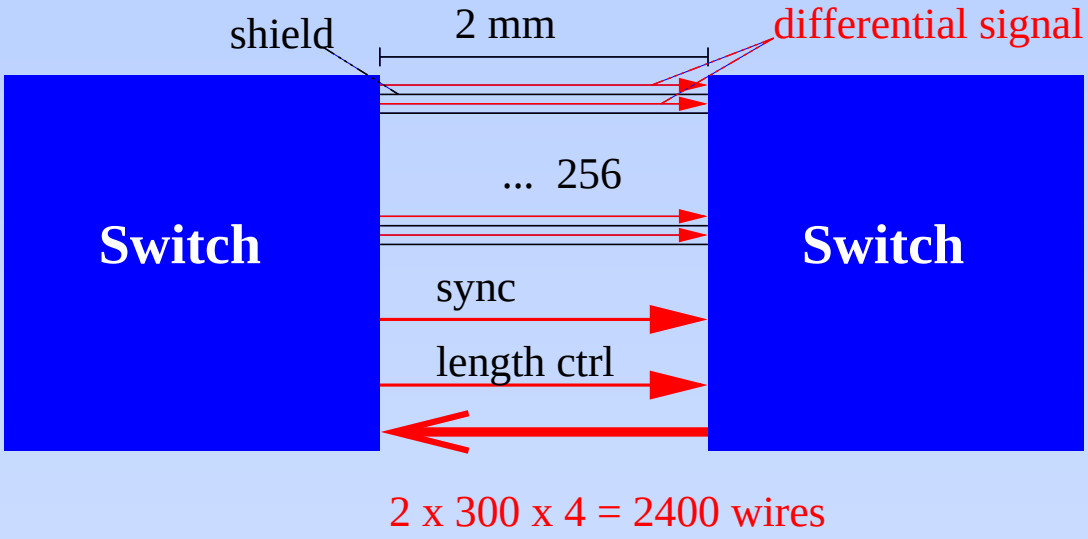
Communication is Key on Several Levels



Communication Layers and unit of communication:

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

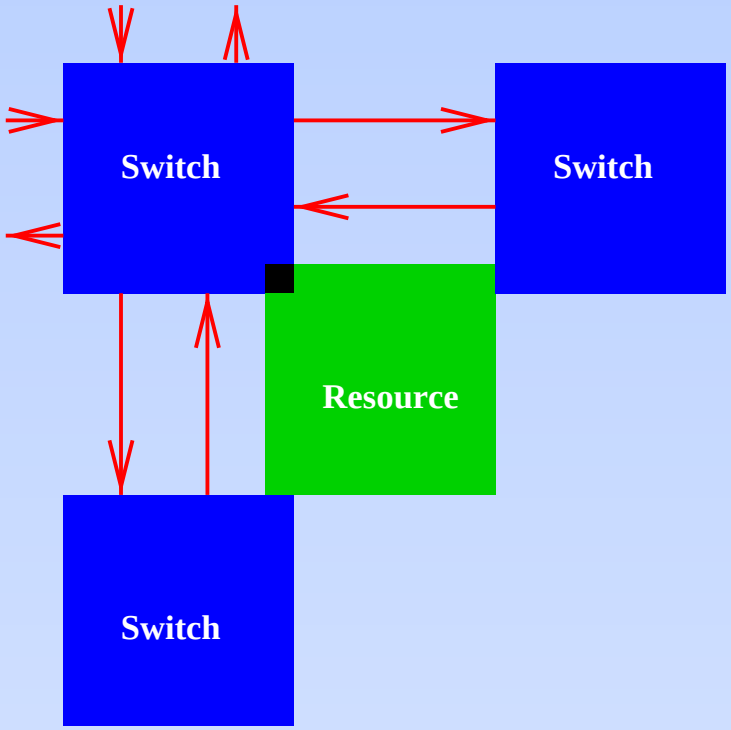
Physical Layer



Parameters:

- Physical distance
- Number of lines
- Activity control
- Buffers and pipelining

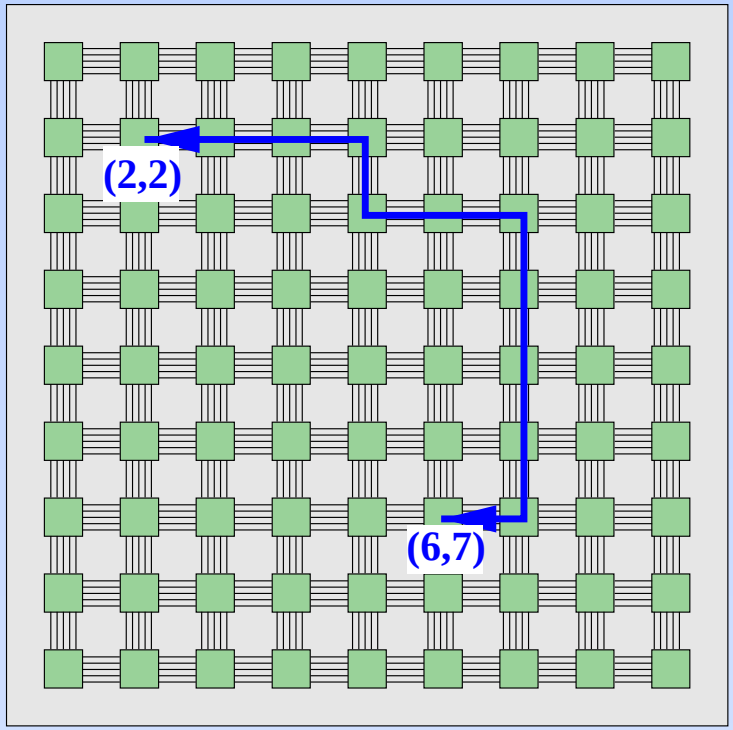
Data Link Layer



Parameters:

- Line frequency versus switch frequency (word versus cell)
- Buffering
- Error correction
- Power optimization; e.g. avoid activity and power optimized encoding

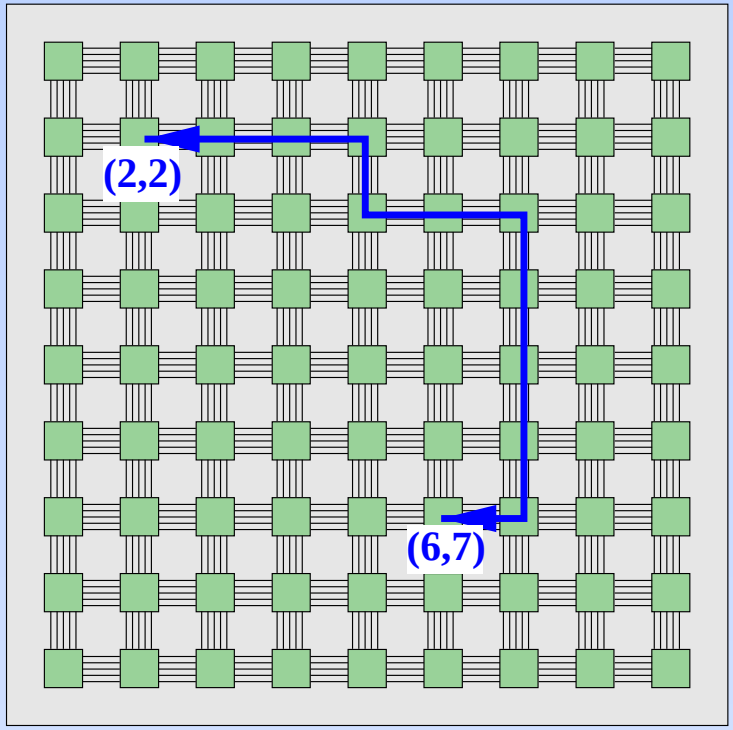
Network Layer



Parameters:

- cell size versus packet size
- Network address scheme,
  e.g. $4 + 4$ bit for 16×16 resources
- Routing algorithm
- Priority classes: e.g. 2 classes:
  1. high priority, fixed delay cells
  2. low priority, best effort delay cells
- Error correction

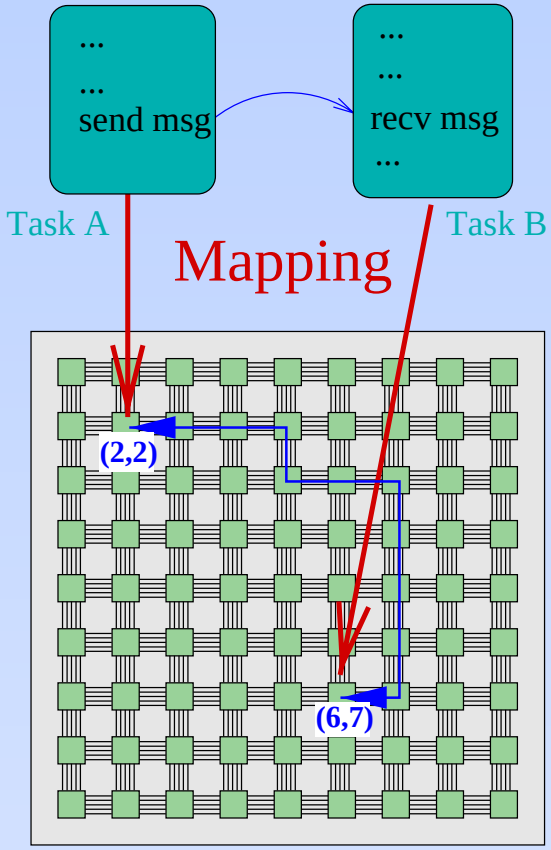
Transport Layer



Parameters:

- Message size versus cell size
- 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
- Error correction

Application Layers



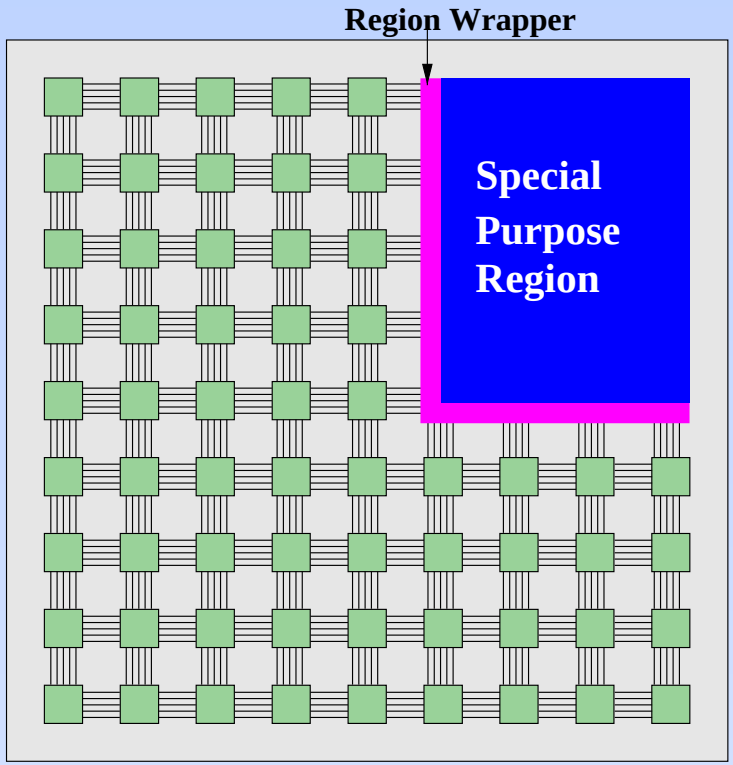
Interprocess communication at the task level:

- `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

Regions



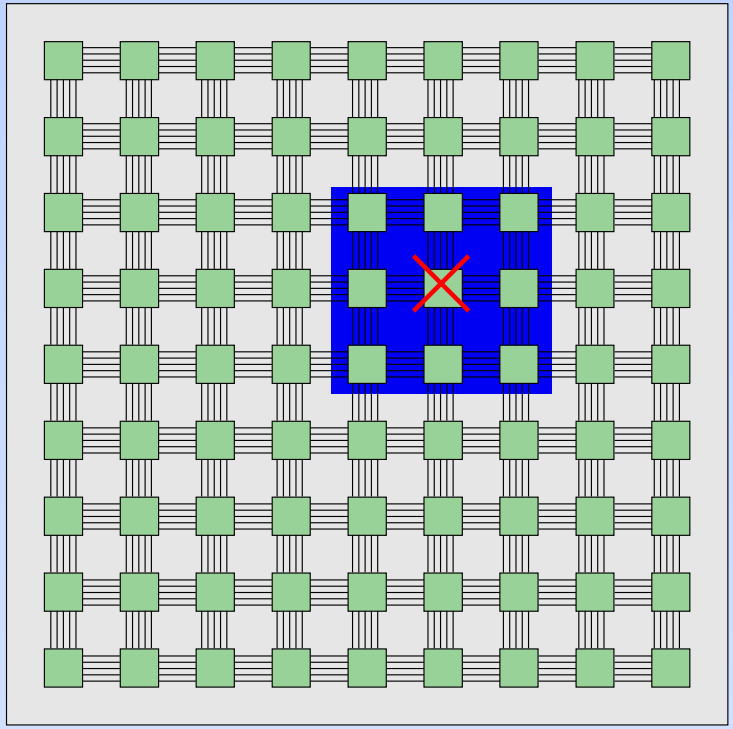
Parameters:

- Not all resources have the same size; e.g.
  - ⋆ Memory
  - ⋆ FPGA regions
  - ⋆ Special purpose architectures like multiprocessors
  - ⋆ mixed signal parts
- A region wrapper makes the region transparent
- The region wrapper can be
  - ⋆ at several protocol layers,
  - ⋆ in hardware or software
  - ⋆ local or distributed

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
- Development support services
  - ⋆ Libraries for run-time services
  - ⋆ Compilers, linkers and simulators

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 to route around the faulty resource

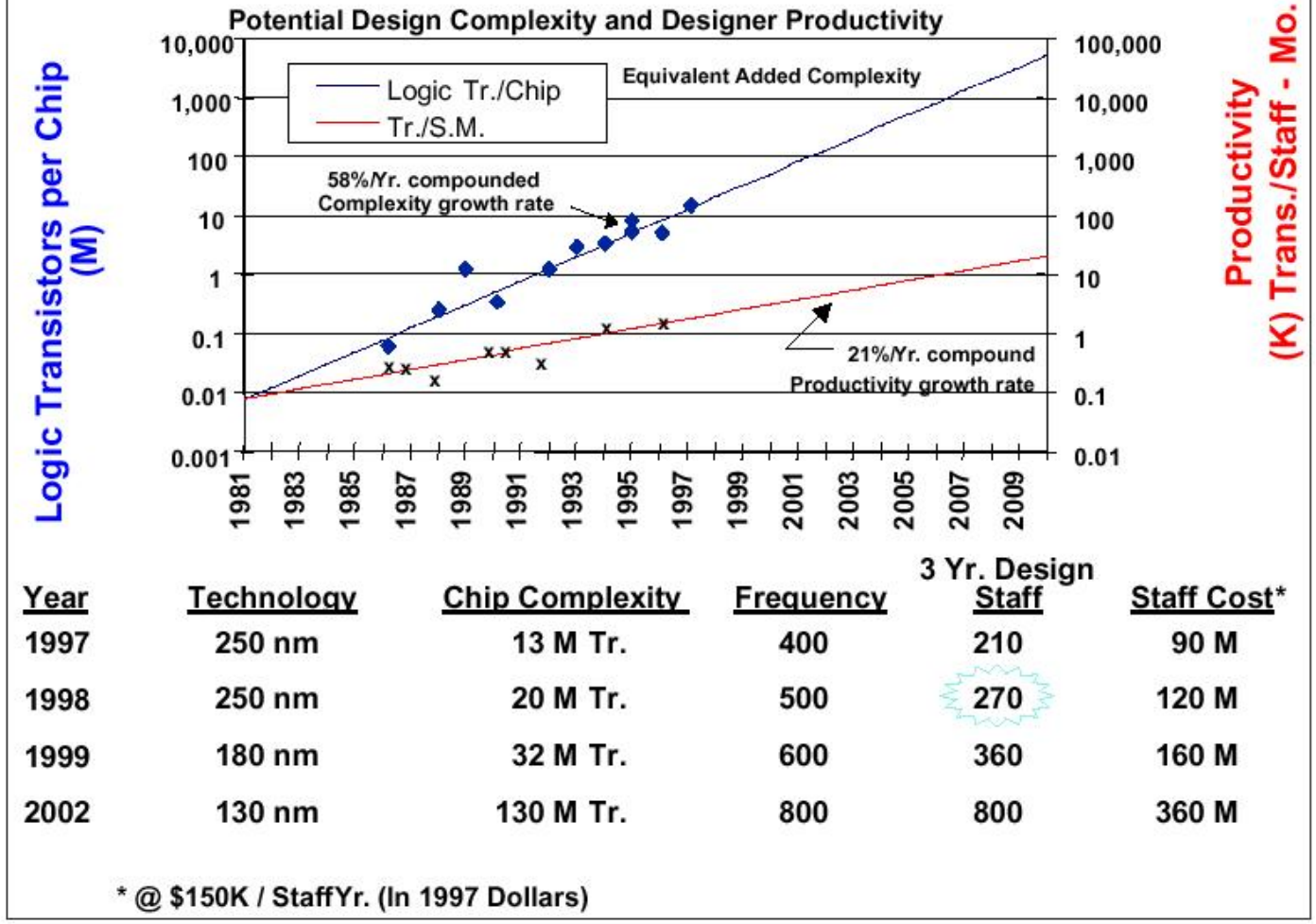
Summary - NoC Architecture

- The NoC Architecture defines
  - ⋆ the communication infrastructure
  - ⋆ the resource-to-network interface
  - ⋆ the network services
- Strict layering of communication protocols and services allows
  - ⋆ the separation of the network backbone from the resources
  - ⋆ combination of different features and functionality at different levels
  - ⋆ the customization of a generic platform into an efficient product

NoC Design Methodology

- Design Productivity Gap
- Arbitrary Composability
- Reuse
- Predictability

Design Productivity Gap

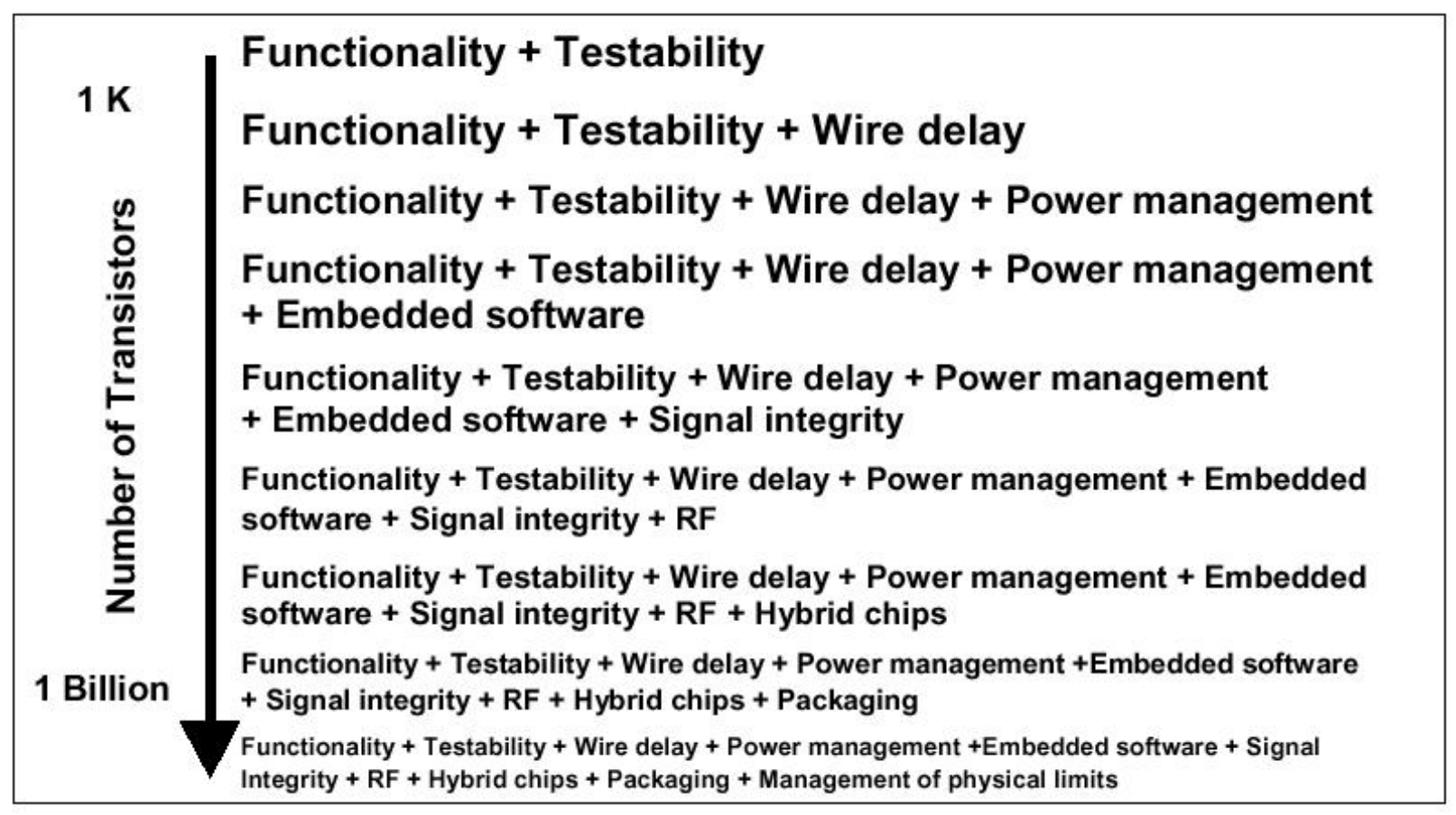


Year	Technology	Chip Complexity	Frequency	3 Yr. Design Staff	Staff Cost*
1997	250 nm	13 M Tr.	400	210	90 M
1998	250 nm	20 M Tr.	500	270	120 M
1999	180 nm	32 M Tr.	600	360	160 M
2002	130 nm	130 M Tr.	800	800	360 M

* @ $150K / StaffYr. (In 1997 Dollars)

International Technology Roadmap for Semiconductors 1999

Superexponential Increasing Design Complexity



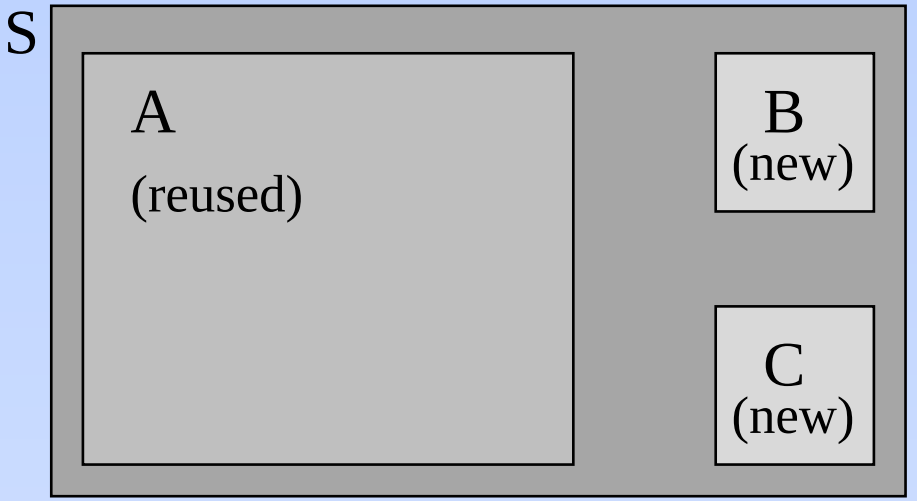
International Technology Roadmap for Semiconductors 1999

Arbitrary Composability

Given a set of components C and combinators O.
Let A_1 be a component assemblage.
(C, O) is arbitrary composable if

$$A_1 + B \Rightarrow A_2$$

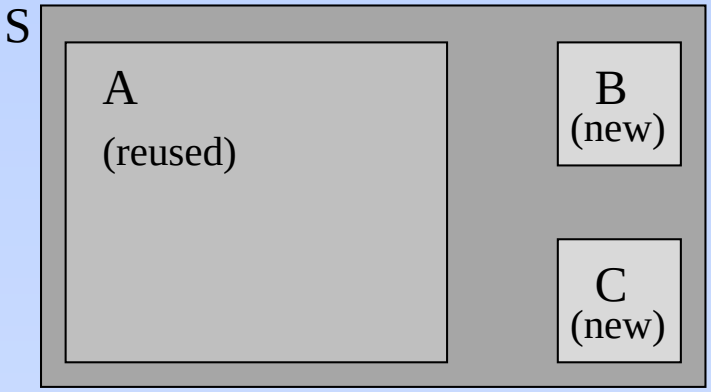
can be done for any $B \in C, + \in O$ without changing the relevant behaviour of A_1.



Linear Effort Property

Given a set of components C and combinators O.
Let $A_1, \ldots, A_n$ be component assemblages.

A design process using C and O to build a system has the linear effort property if $A_1, \ldots, A_n$ can be integrated into a system S with an effort dependent on n but not on the size of the assemblages: $\texttt{Ieffort}(n)$.
Total design effort for S is



$$\texttt{Deffort}(S) = \texttt{Deffort}(A_1) + \cdots + \texttt{Deffort}(A_n) + \texttt{Ieffort}(n)$$

Reuse

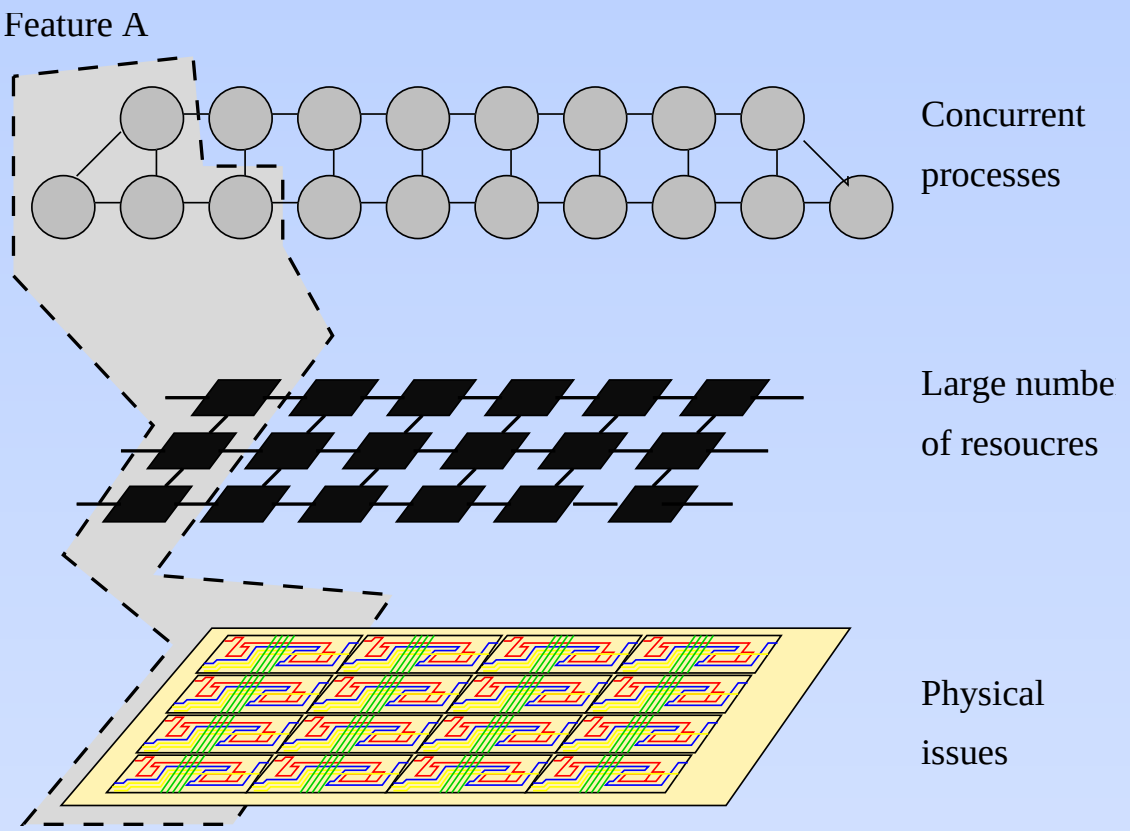
- Components and resources
- Communication infrastructure
- Application parts and feature reuse
- Design, simulation and prototype environment
- Verification effort

Predictability

- Communication performance
- Electrical properties
- Design and verification time

A NoC Based Design Process

- Configuring the platform
- Selecting resources
- Reuse of features
- Evaluation and integration



Design Methodology - Summary

- A NoC Platform greatly restricts the design space
- It trades in optimality for design efficiency and predictability
- The arbitrary composability and the linear effort properties provide a scalable platform
- Reuse
- Predictability

Further information: www.imit.kth.se/info/FOFU/NOC/