Networks on Chip: A New Contract between Hardware and Software

Axel Jantsch
Royal Institute of Technology, Stockholm

September 2003

Overview

Current Trends

Makimoto's Pendulum

Nostrum Architecture and Protocol Stack

NoC Design Flow

NoC Programming Model

Summary

Current Trends

- Deep sub-micron effects make it harder to maintain global synchrony.
- Synthesis and compiler technology cannot bridge more than one abstraction level.
- Single chip applications become more heterogeneous.

Globally Asynchronous Locally Synchronous Designs

- The clock signal takes many cycles across the chip in a deep sub-micron gigahertz design.
- The clock tree consumes already up to 50% of the total power with increasing trend.
- Process variations on the chip lead to different clock signal delays.

Globally Asynchronous Locally Synchronous Designs

- The clock signal takes many cycles across the chip in a deep sub-micron gigahertz design.
- The clock tree consumes already up to 50% of the total power with increasing trend.
- Process variations on the chip lead to different clock signal delays.

Trend for current and next technology generations:

- Synchronous regions of size 50 - 2000 k-Gates;
- Synchronous clusters (computing element + memory) communicate asynchronously over communication network.

Synthesis and Compiler Technology

- Synthesis technology can bridge only one shallow abstraction level.
- Compilers for C, Pascal, Fortran, etc. are very successful.
- Compilers for more abstract languages (Prolog, ML, Haskell, etc.) do not produce as efficient code.
- RTL synthesis (direct structural mapping with logic optimisation) is very successful.
- High level synthesis (resource allocation, binding, scheduling) does not produce as efficient designs.

Synthesis and Compiler Technology

- Synthesis technology can bridge only one shallow abstraction level.
- Compilers for C, Pascal, Fortran, etc. are very successful.
- Compilers for more abstract languages (Prolog, ML, Haskell, etc.) do not produce as efficient code.
- RTL synthesis (direct structural mapping with logic optimisation) is very successful.
- High level synthesis (resource allocation, binding, scheduling) does not produce as efficient designs.

Current trend:

- Designer responsibility:
  - ⋆ Functional model of behaviour of synchronous cluster;
  - ⋆ Precise specification of communication semantics;
  - ⋆ Assignment of tasks to resources;
- Synthesis tools:
  - ⋆ Traditional compilation/synthesis of a synchronous cluster;
  - ⋆ Direct refinement and mapping of communication onto provided communication services;

Heterogeneous Applications

Most applications grow by accumulating diverse functionality.

- Mobile phone ⇒ mobile office
  (phone + email + Internet + word processing + calendar + language translation + ...)
- Car GPS ⇒ navigation assistant (GPS + maps + tourist guide + traffic information system + ...)

Heterogeneous Applications

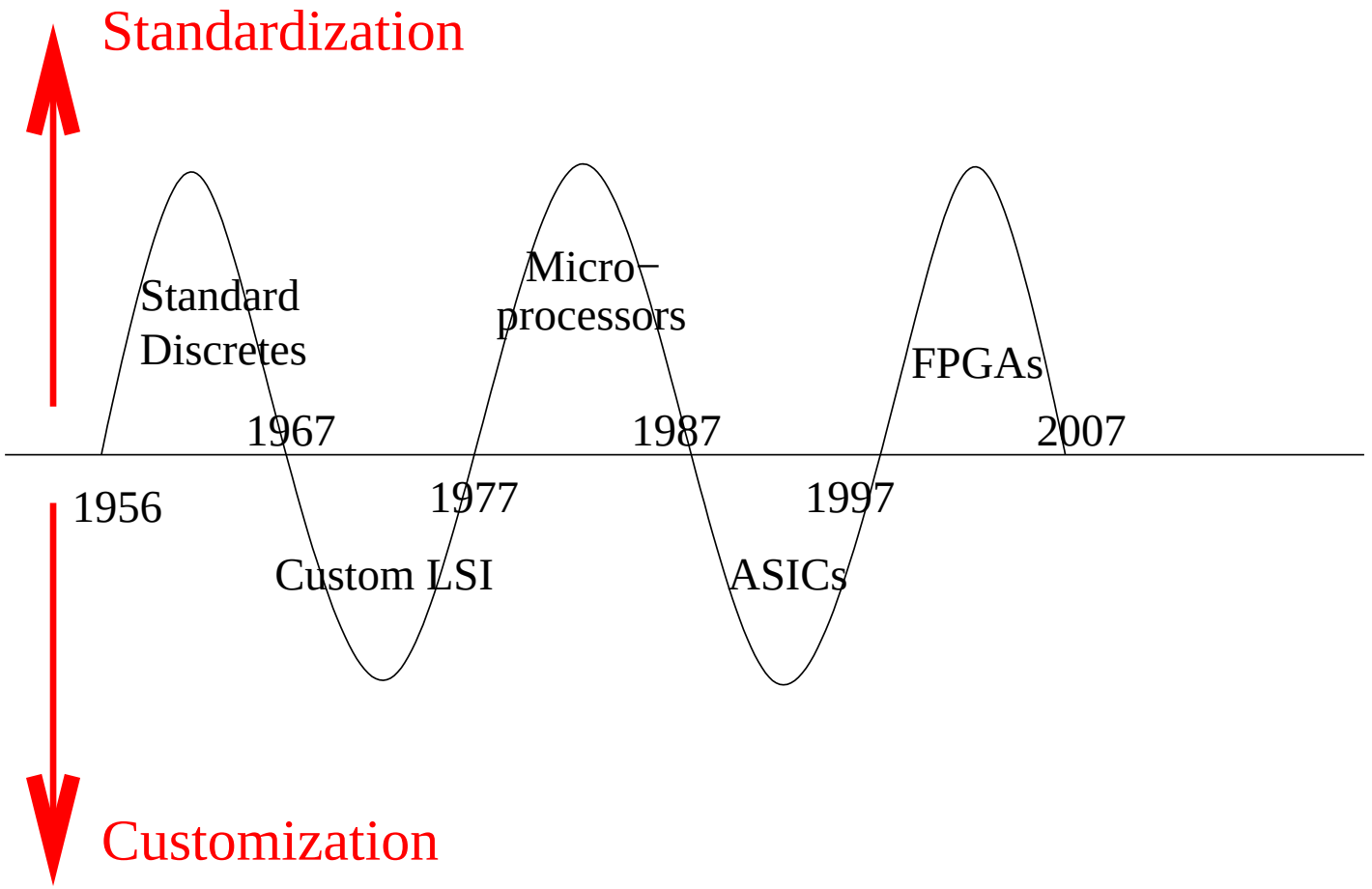
Most applications grow by accumulating diverse functionality.

- Mobile phone ⇒ mobile office
  (phone + email + Internet + word processing + calendar + language translation + ...)
- Car GPS ⇒ navigation assistant (GPS + maps + tourist guide + traffic information system + ...)

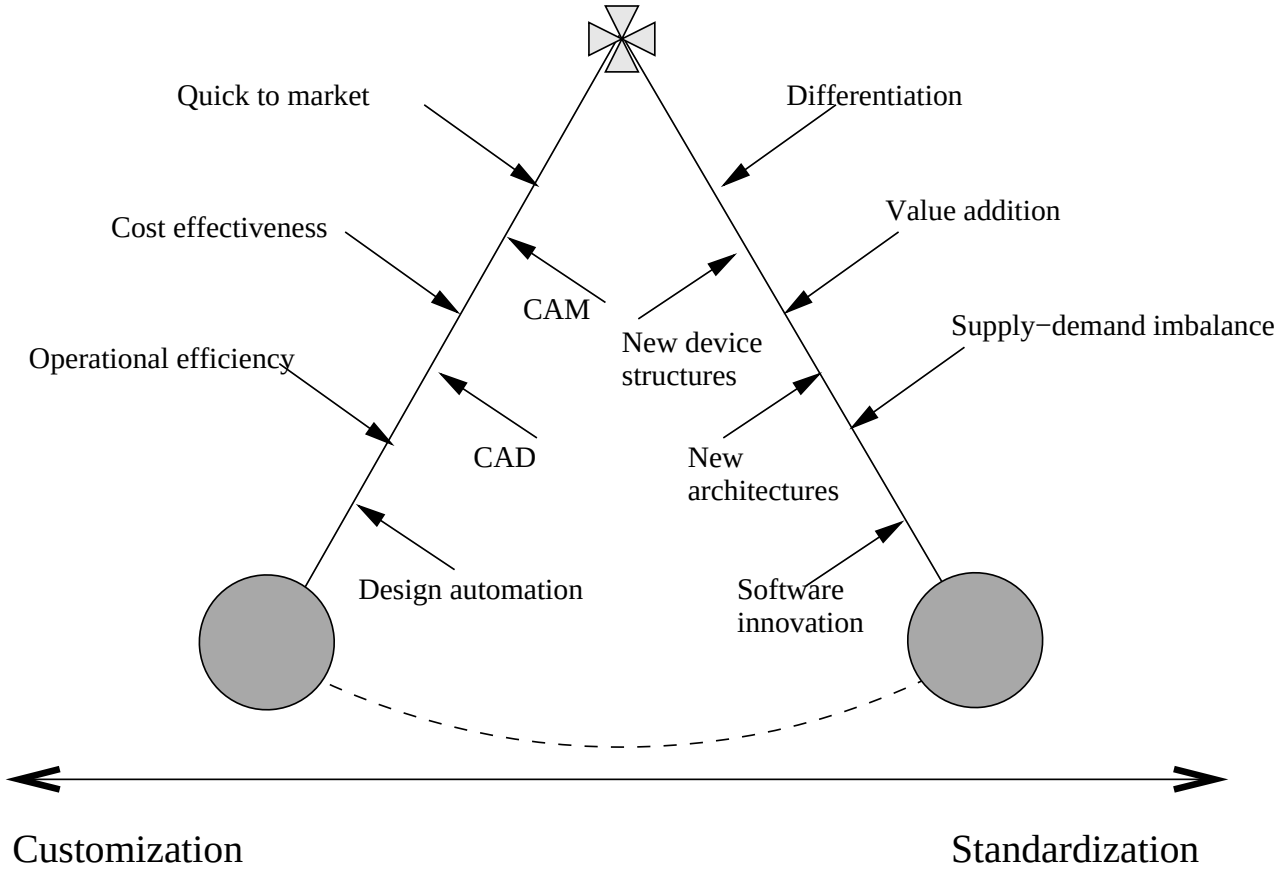
Current trend:

- Ever more complex IPs (components, functions, ...) are reused.
- A function with its implementation (IP) is reused in new applications.
- Search for well defined interfaces to connect IPs (VCI, Amba, OCP, ...).

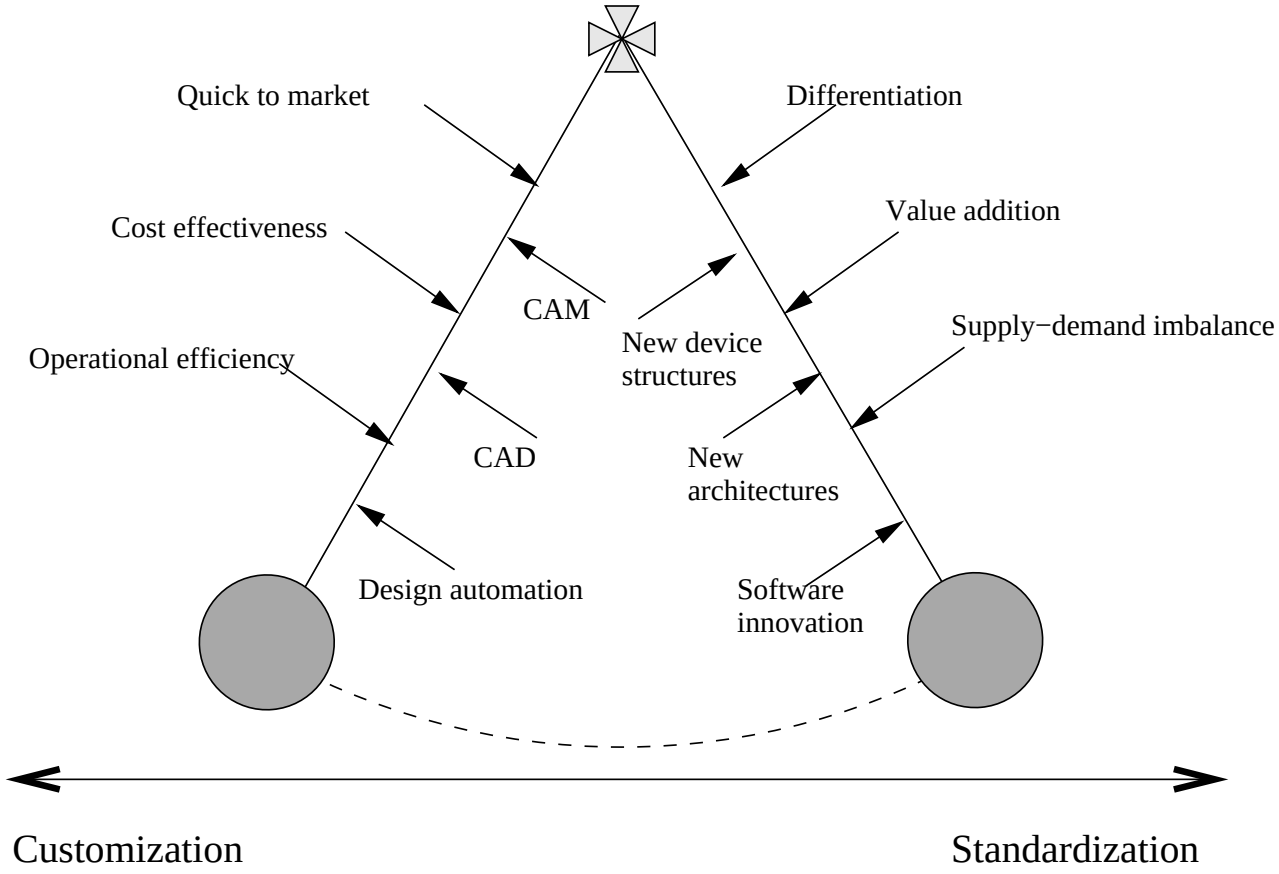
Makimoto Waves



Makimoto's Pendulum



Makimoto's Pendulum



Current forces

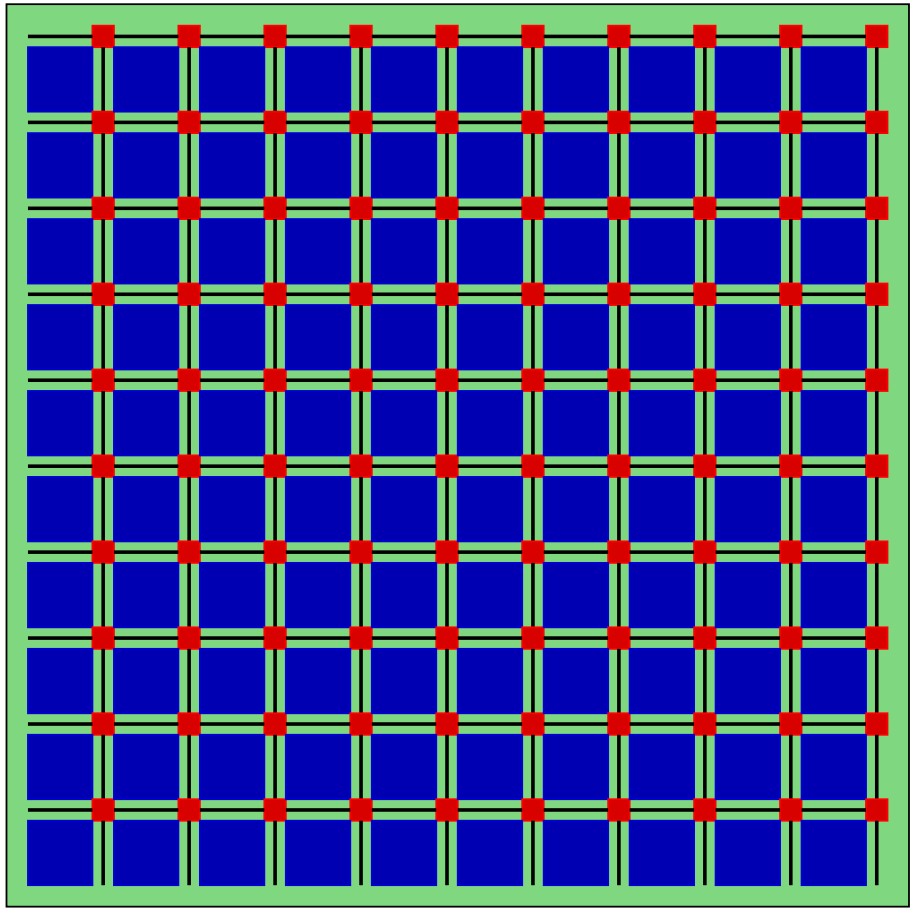
- Manufacturing cost efficiency (fabs, design costs, masks, testing),
- Demand for increased operational efficiency and maintenance efficiency,
- New technologies (deep sub-micron, memory, packaging)
- New architectures (FPGAs, ASIPs, platforms, NoCs)

push towards standardised platforms.

Summary of Observations and Trends

- Technology development
  ⇒**Locally synchronous clusters communicating asynchronously;**
- Application development
  ⇒**Integration of independently developed functions and features into new systems;**
- Desire to control costs
  ⇒**Standardization of manufacturing, operation and usage;**
- New technology, devices, architectures
  ⇒New possibilites and problems
  ⇒**Standardization of design flows and platforms;**

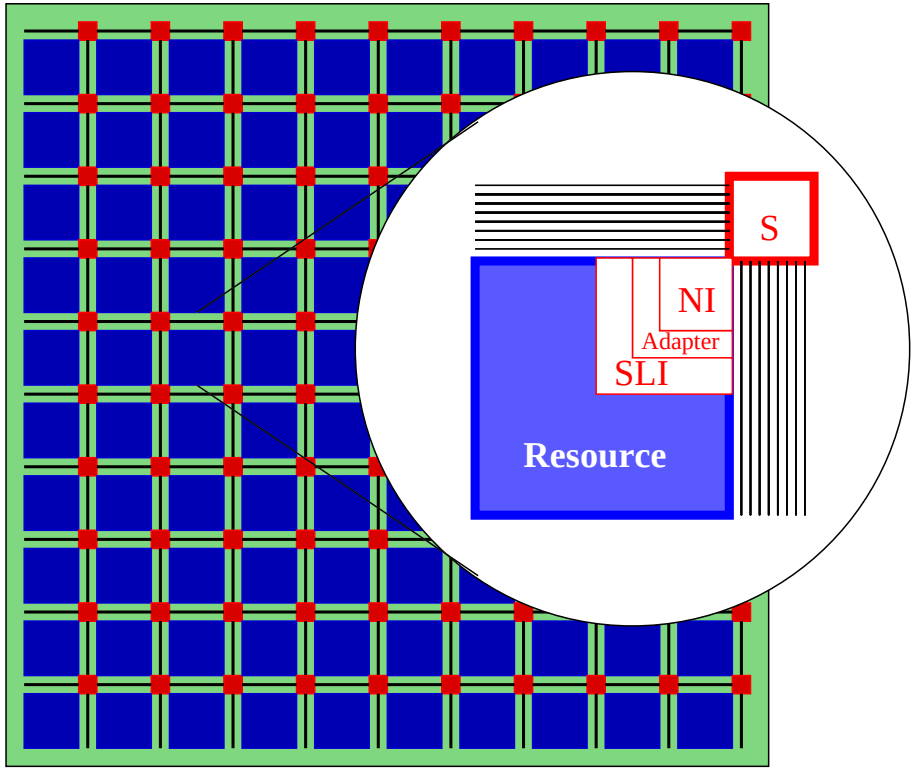
NoC Example: Nostrum



Characteristics:

- Regular 2-dimensional $k \times k$ mesh topology
- Resource-to-switch ratio: 1
- A switch is connected to 4 switches and 1 resource
- A resource is connected to 1 switch
- Average distance: $2/3k$
- Bisection: $2kw$

The Node in a Mesh



NI: Network Interface:

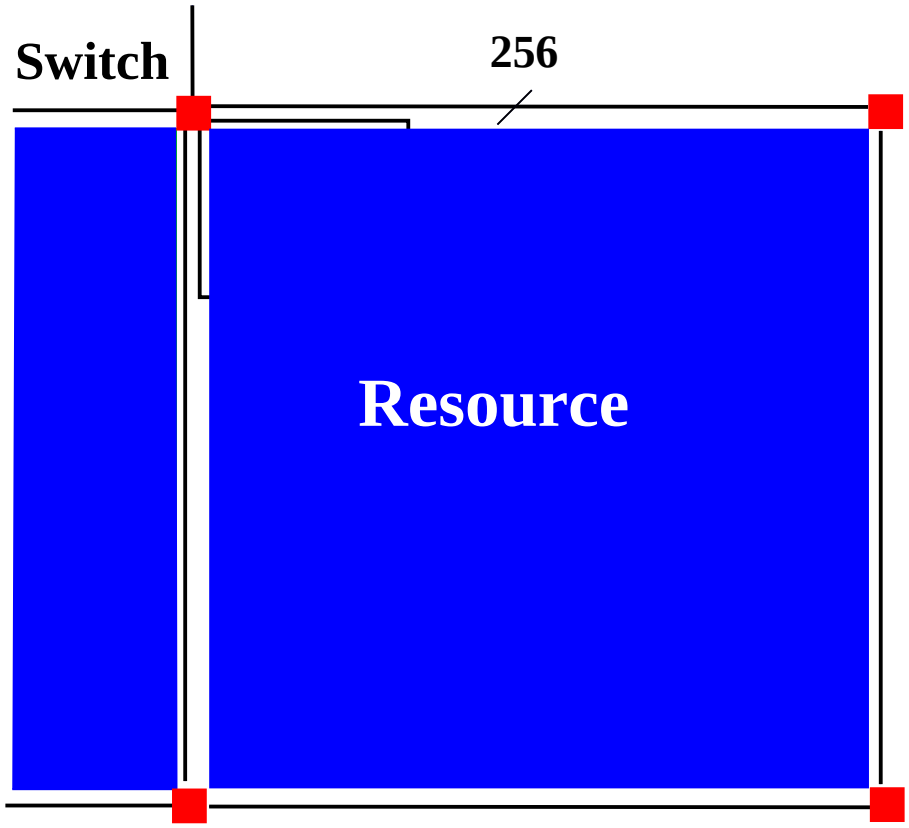
- Compulsory
- HW
- Implements the network layer protocol

Adapter: Resource specific interface circuit;

SLI: Session Layer Interface:

- Optional
- Hardware and/or software
- Implements the session layer protocol

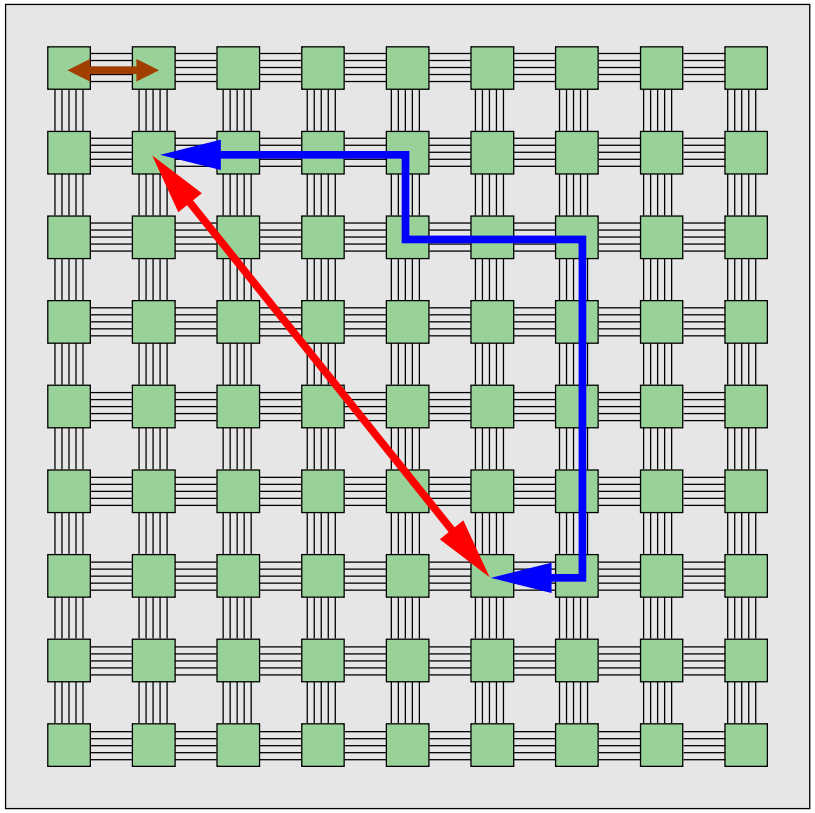
Node Geometry



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

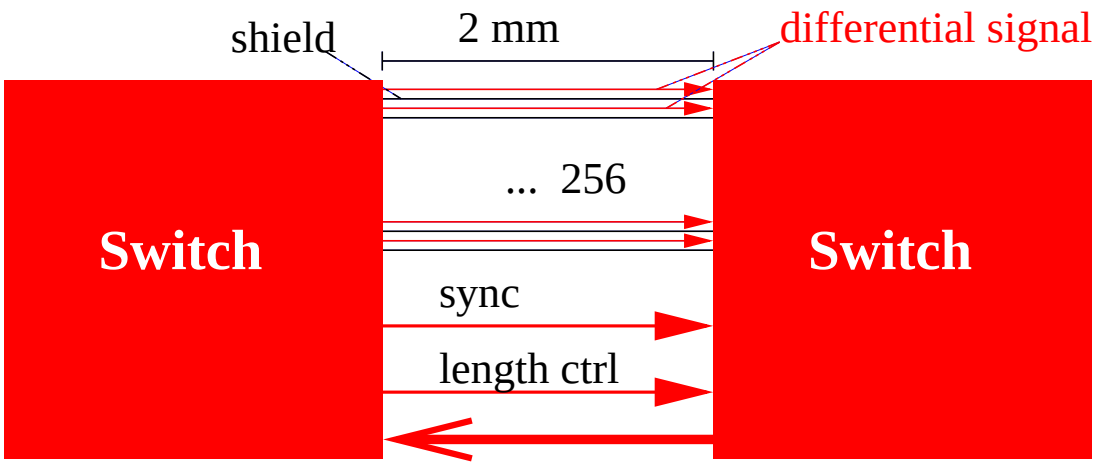
Communication is Key



Communication Layers:

- Physical layer: switch-to-switch and switch-to-resource
- Data link layer: switch-to-switch and switch-to-resource
- Network layer: resource-to-resource
- Session layer: process-to-process
- Application layer: application-to-application

Physical Layer



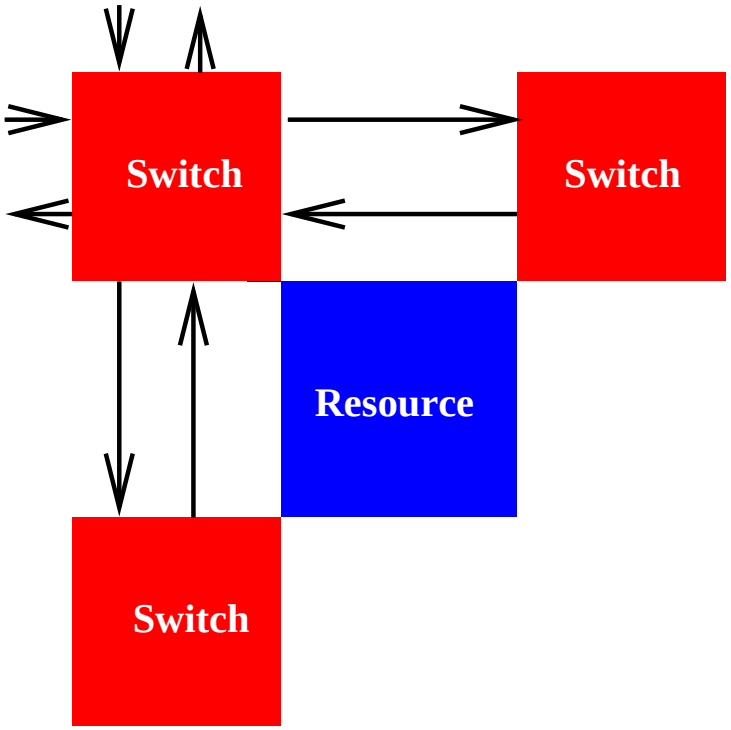
Parameters:

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

Nostrum status:

- 128 data lines in each direction
- No pipelining
- On/off control for power saving

Data Link Layer



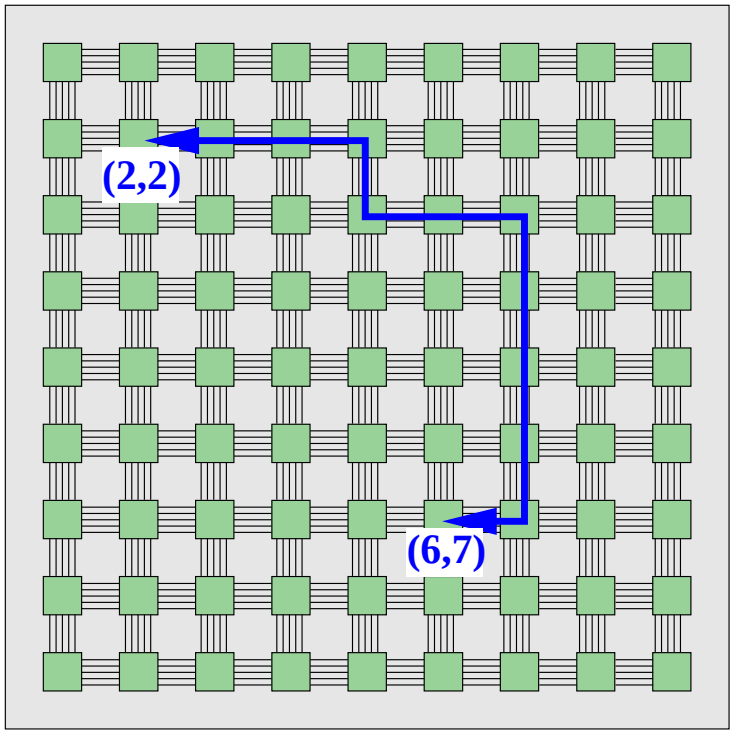
Parameters:

- Line frequency versus switch frequency
- Buffering
- Error correction
- Power optimization encoding

Nostrum status:

- Physical packet = data link packet
- Physical clock = data link clock
- Single packet input buffer
- Error correction
- On/off activity control

Network Layer



Parameters:

- Link layer cell size vs. network layer packet size
- Network address scheme
- Routing algorithm
- Priority classes
- Error correction

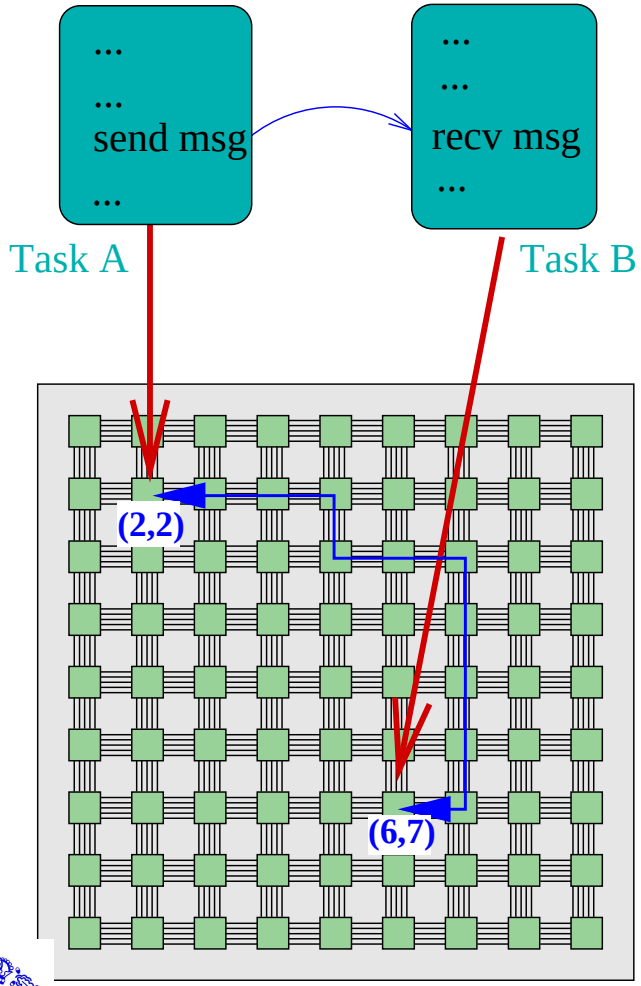
Nostrum status:

- Link layer packet = network layer packet
- Relative x-y addresses
- Deflective routing with no buffers and no routing tables
- Virtual circuits with guaranteed bandwidth and delays
- No error protection

Network Layer Services

- Packet switched best effort service
  - ⋆ Packets are guaranteed to arrive
  - ⋆ Packet payload may be protected (4 levels of protection)
  - ⋆ Load dependable delay in the network
  - ⋆ Load dependable delay at the network access point
- Virtual circuit service
  - ⋆ Guaranteed bandwidth
  - ⋆ Guaranteed maximum delay
  - ⋆ Multicast circuits
  - ⋆ Based on packet switching service

Session Layer



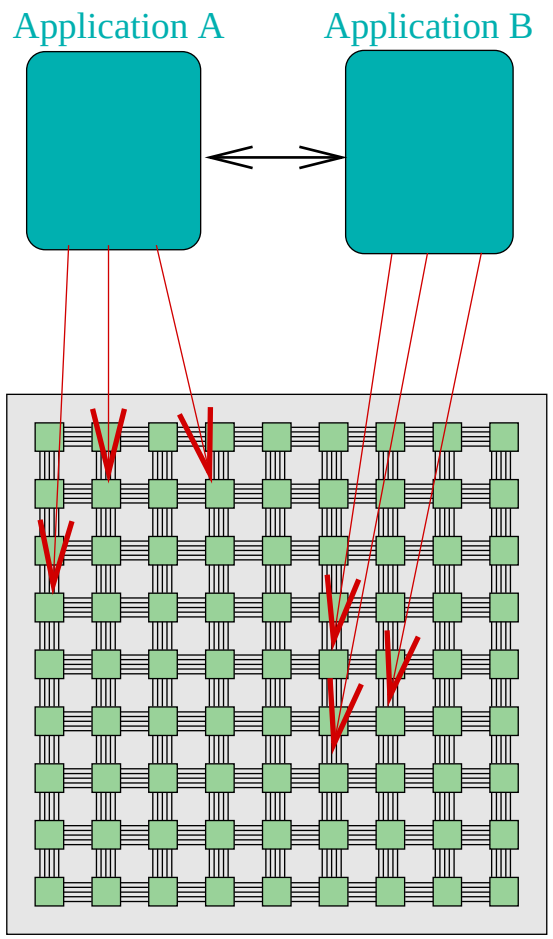
Parameters:

- Task level communication primitivs
- Message passing
- Shared memory based communication
- Synchronization
- Error correction

Nostrum status:

- Set of communication primitivs defined
- Both message passing and shared memory
- User controlled synchronization
- Optional end-to-end data protection

Application Layers



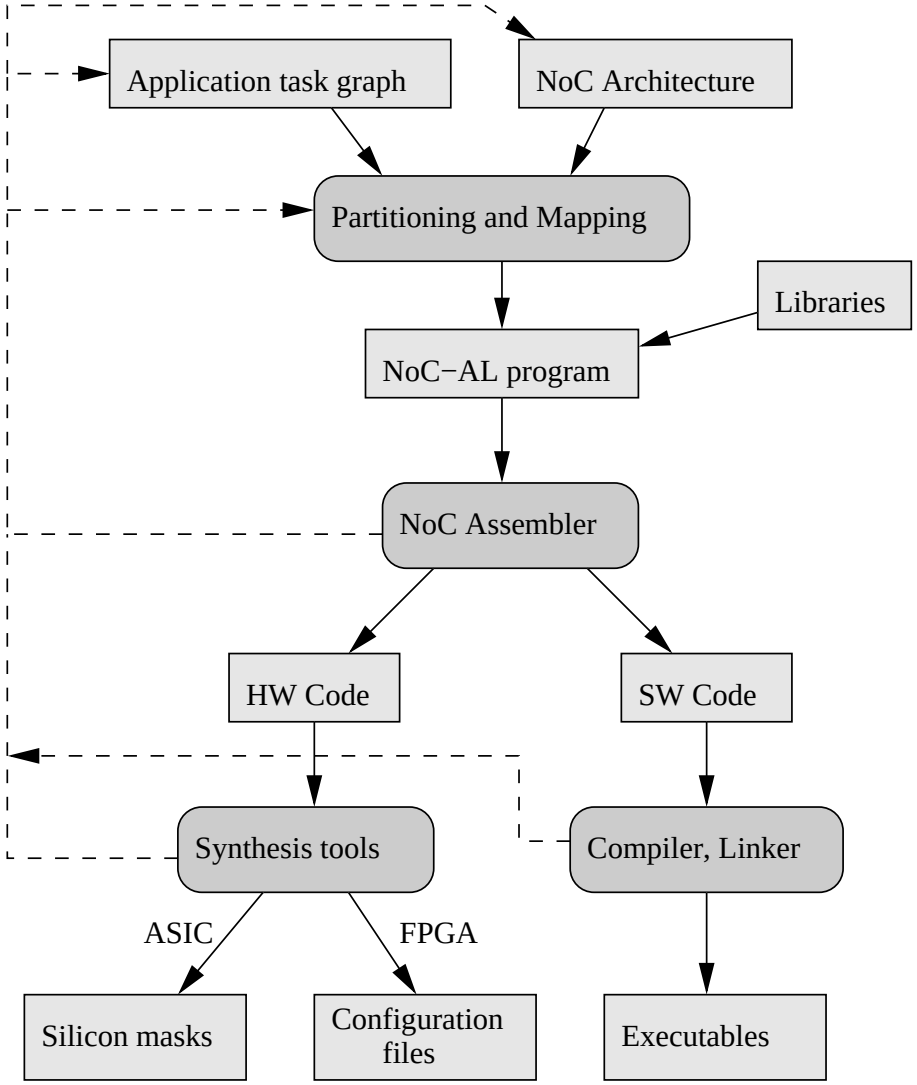
Application specific communication services;
E.g. the NoC operating system could use:

- Task/resource database access protocol
- Task migration protocol

What a NoC based Platform Offers

- A set of resource slots;
- Well defined set of communication services with performance characteristics;
- Resource management services;
- A design methodology for
  - ⋆ Network configuration and allocation;
  - ⋆ Resource implementation;
  - ⋆ Communication refinement.

NoC Based Design Flow



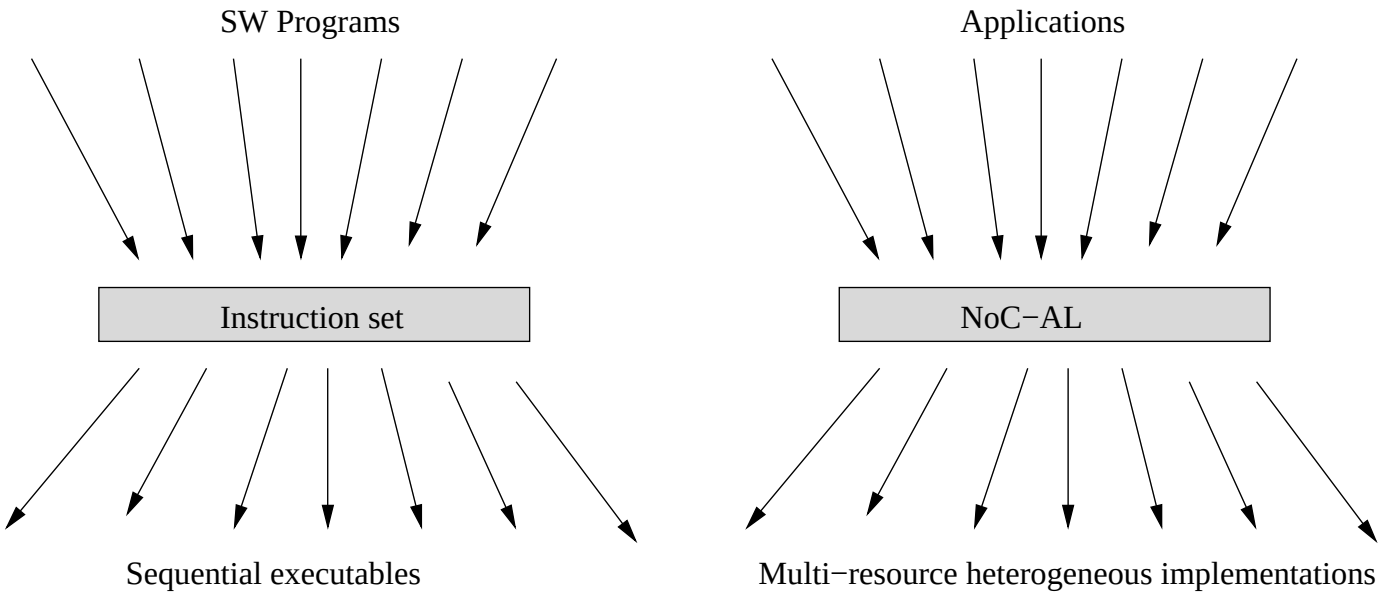
NoC Assembler Language Program Example

```
NoC Architecture
{ Topology: mesh 1 x 2
  Resource List: Row1: R1=SHARC DSP,
                       R2=ARM CPU}
NoC Application
{ R1: {#include <NoC-AL-SHARC.h>
       #include <f1.h>
       #include <f2.h>
       double in1,in2,out,x,y;
       int ch1=0, ch2=0, ach2=0;
    Process P11
    { while (1)
      { x=f1(in1,in2);
        while (ch1<=0) {ch1=channel(P11,P21);} //Open channel ch1 until success
        while (send(ch1,x)!=1) {continue;}}    //Wait for send to ch1 success
      while (close(ch1)!=1) {continue;}}       //Close channel ch1
    Process P12
    { while (1)
      { while (ach2<=0) {ach2=accept(&ch2);}   //Wait for channel ch2 accepted
        while (receive(ch2,y)!=1) {continue;}  //Wait for receive from ch2 success
        out=f2(y);}}}
```

NoC Assembler Language Program Example - cont'd

```
R2: {#include <NoC-AL-ARM.h>
     #include <f3.h>
     double a,b;
     int ch1=0, ch2=0, ach1=0;
  Process P21
  { while (1)
    { while (ach1<=0) {ach1=accept(&ch1);} //Wait for channel ch1 accepted
    while (receive(ch1,a)!=1) {continue;}  //Wait for receive from ch1 success
    b=f3(a);
    while (ch2<=0) {ch2=channel(P21,P12);} //Open channel ch2 until success
    while (send(ch2,b)!=1) {continue;}}    //Wait for send to ch2 success
    while (close(ch2)!=1) {continue;}}}}   //Close channel ch2
```

The NoC-AL Contract



- Defines NoC configuration;
- Defines functionality of processes;
- Defines functionality of inter-process communication;
- Defines performance of inter-process communication;
- Defines mapping of processes to resources;

Summary

- Technology development
  ⇒**Locally synchronous clusters communicating asynchronously;**
- Application development
  ⇒**Integration of independently developed functions and features into new systems;**
- Desire to control costs
  ⇒**Standardization of manufacturing, operation and usage;**
- New technology, devices, architectures
  ⇒New possibilites and problems
  ⇒**Standardization of design flows and platforms;**

Summary

- Technology development
  ⇒**Locally synchronous clusters communicating asynchronously;**
- Application development
  ⇒**Integration of independently developed functions and features into new systems;**
- Desire to control costs
  ⇒**Standardization of manufacturing, operation and usage;**
- New technology, devices, architectures
  ⇒New possibilites and problems
  ⇒**Standardization of design flows and platforms;**

- A NoC implementation is a set of asynchronously communicating clusters.
- An application is modeled as a set of communicating processes.
- Communication is at the center of a NoC contract;
- The NoC contract is a natural extension of the IS/HDL contract;
- The extensions include task-resource mapping, communication functionality and communication performance;
- The NoC contract separates platform development from application development;
- It allows standardization of platforms and design flows.