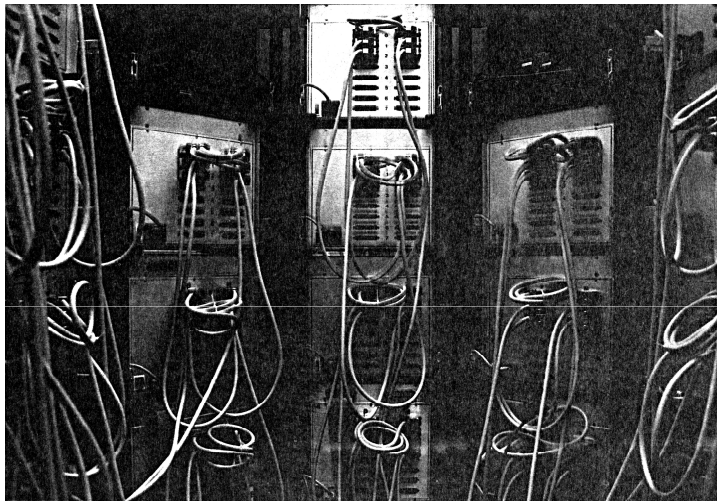
Adaptive Verbindungsnetzwerke für Parallelrechner: vom SAN (System Area Network) zum NoC (Network on Chip)

Erik Maehle
Universität zu Lübeck
Institut für Technische Informatik
maehle@iti.uni-luebeck.de

Informatik-Kolloquium
Universität Erlangen-Nürnberg, 19. Februar 2011

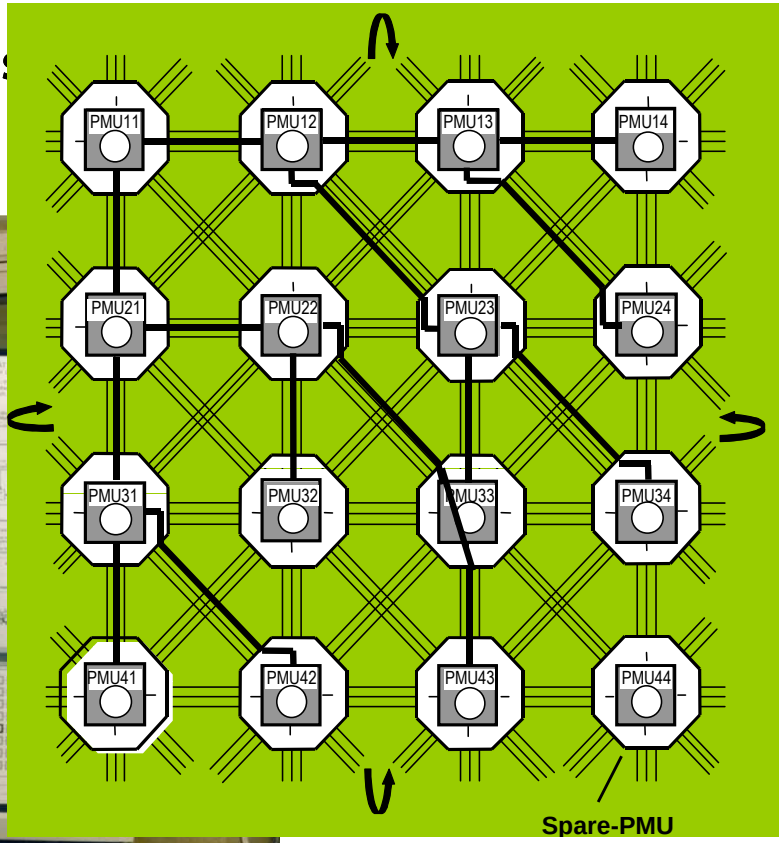
DIRMU (DIstributed Reconfigurable Multiprocessor) University of Erlangen 1983-1995 (IMMD III, Prof. Händler)

25 Processor Modules (8086/87)
Multi-Port Memories



Manually Reconfigurable Topologies
(up to 7 Neighbors)

DAMP (Dynamically Adaptable Multi-Process
University of Paderborn 1990 - 1994



Octogonal Torus
Direct Message Passing
to Remote PMUs

64 Transputer Modules
with Local Crossbars

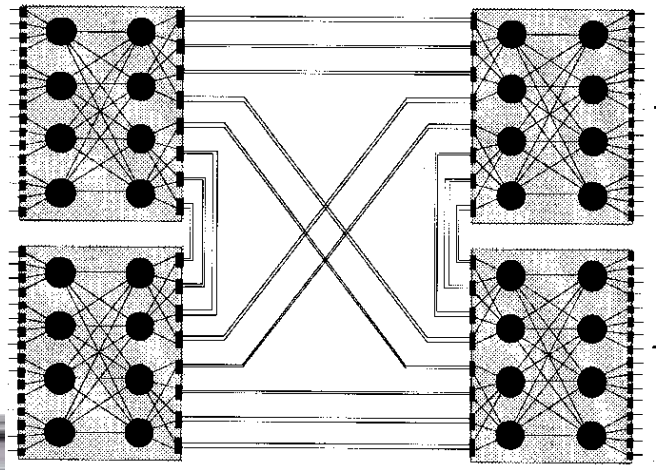
Störtebecker Cluster (Lübeck 1998 - 2005)

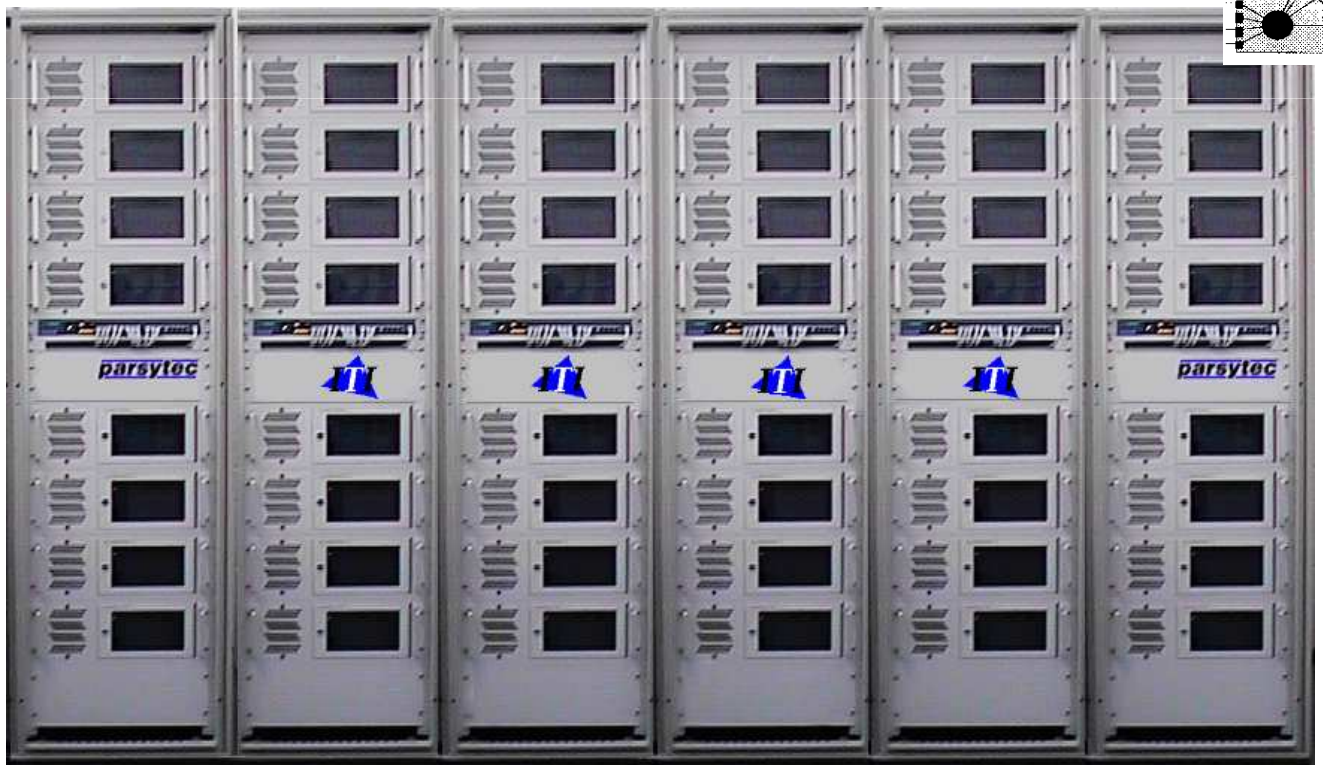
Nodes:

2 * Pentium II @ 333 MHz
128 MByte RAM
4 GByte Harddisk

Cluster:

48 Nodes, 96 Processors
6 GByte RAM
192 GByte Harddisk





SAN (System Area Network): Myrinet I

1.28 GBit/s
Processor on NIC
Switches (16x16)
Multi-Stage Networks

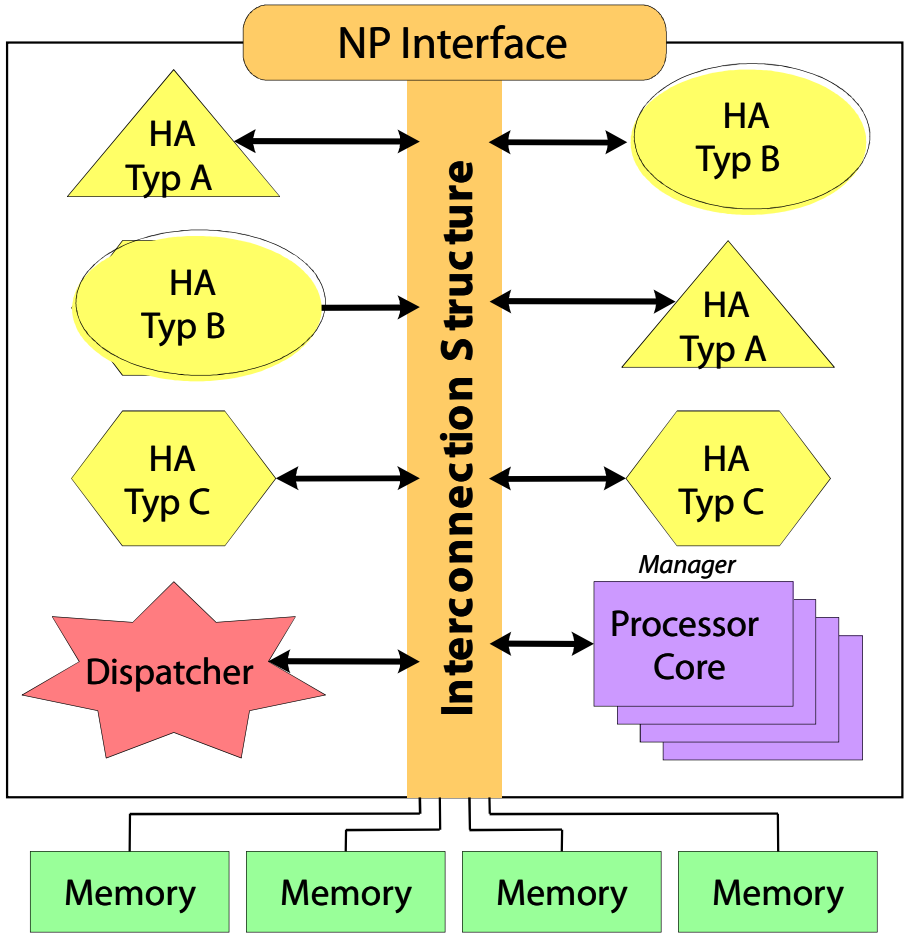
Message Passing with
Wormhole Switching &
Source Path Routing

Fast Ethernet

DynaCORE (Dynamically adaptable COprocessor based on Reconfiguration) University of Lübeck 2003 - 2010

Xilinx Virtex-II Pro (XC2VP30)
FPGA with 30,000 Logic Cells

NoC (Network on Chip) CoNoChi



Outline

- Introduction

- Adaptive Routing in SANs

- Topology-Adaptive NoCs

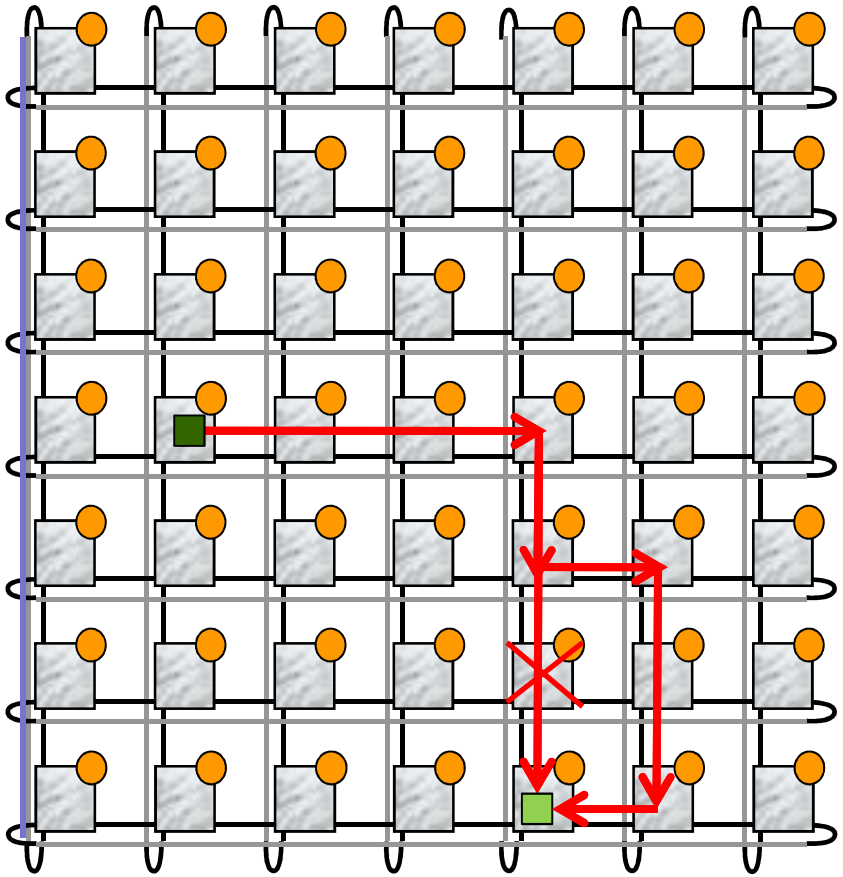
- NoC Adaptation to Selected Data Streams

Adaptive Routing

Requirements for Parallel Computer Interconnection Networks

- Low Latency
- High Throughput
- No Packet Loss
- No Deadlocks, no Lifelocks
- Fault Detection
- Fault Tolerance
- Low Energy Consumption
- Low Complexity
- Low Cost

Adaptive Routing Algorithms



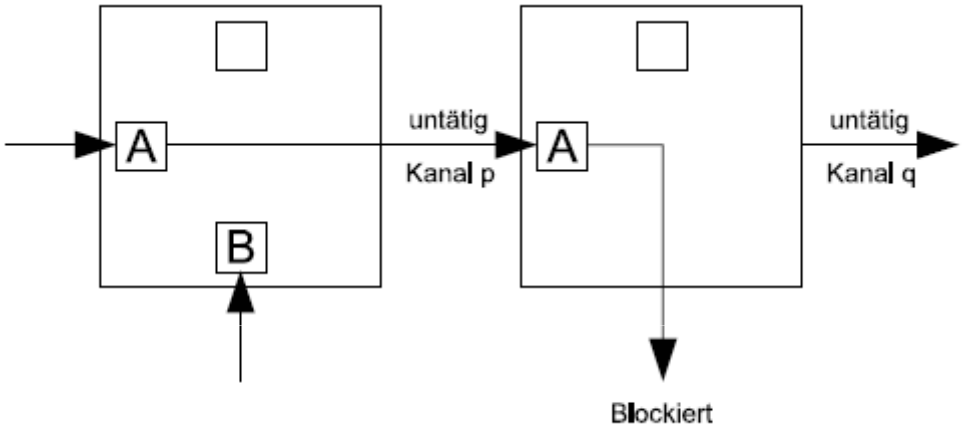
- Connectivity
- Delivery Guarantee
- **Adaptivity**
- Fault Tolerance
- . . .

=> Rule-Based Intelligent Networks
(RuBIN Project)

Supported by DFG
Team: Andreas Döring, Gunther Lustig

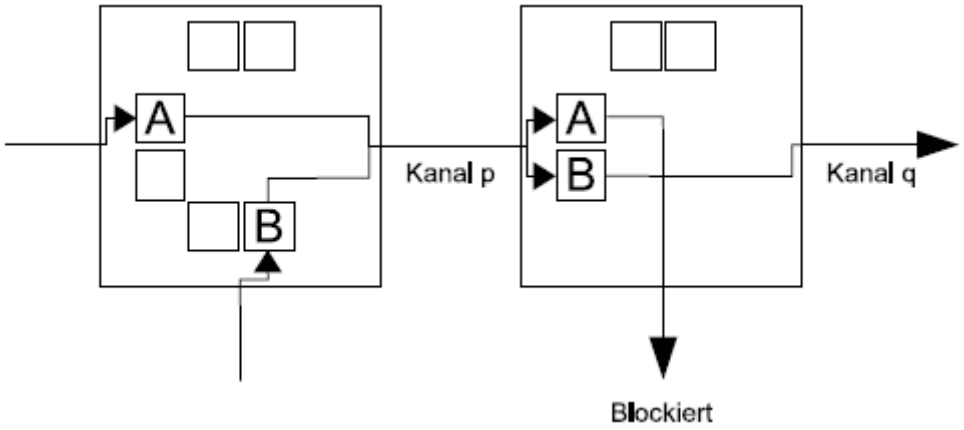
Basic Concepts

Wormhole Switching



Pipelining of Message Flits

Virtual Channels



Multiplexing of Flits on
Physical Channel p

Rule-Based Adaptive Routing

Example: Duato`s Prototcol

Basic Concept

- Use *Adaptive Routing*, if there is a free virtual *Application-Channel* along the minimal path.
- Switch to an *Escape-Channel,* if all these virtual Application-Channels are blocked.
- Use *XY-Routing* on the Escape-Channel (deadlock-free).

```
ON      MessageArrived(InputDir, InChannel, Dest, LPos)
        IF InChannel = AChannel AND
                ∃ OutDir IN MinimalPath(Dest[x], Dest[y],LPos[x],LPos[y]):
                AdaptiveChannelFree[OutDir] = TRUE
        THEN !switch(InputDir, InChannel, OutDir, AChannel);
        IF DEFAULT THEN
                !switch(InputDir, InChannel,
                        routeXY(Dest[x],Dest[y],LPos[x], LPos[y]), EChannel);
END     MessageArrived;
```

- InputDir : input direction of a message
- InChannel: virtual channel, where the message arrived.
- Dest: X- and Y-coordinates of message destination
- Lpos: X- and Y-coordinates of local router position
- !switch: establishes the connection between the virtual input and output channel

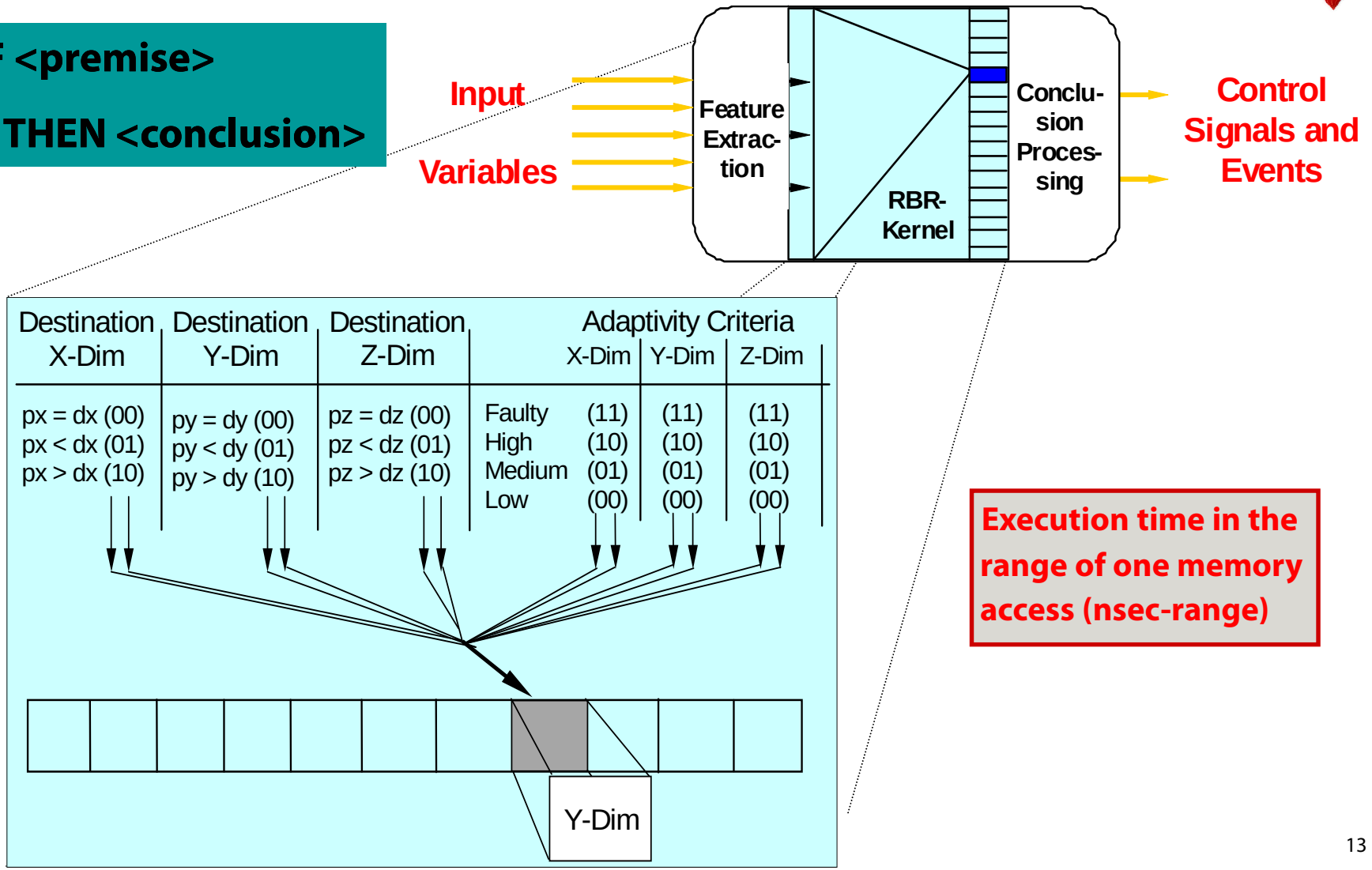

```
SUBBASE routeXY(dx, dy, px, py)
        IF dx > px                  THEN routeXY <- east;
        IF dx < px                  THEN routeXY <- west;
        IF dx = px  AND dy > py     THEN routeXY <- north;
        IF dx = px  AND dy < py     THEN routeXY <- south;
        IF DEFAULT                  THEN routeXY <- home;
END routeXY;
```

```
SUBBASE MinimalPath(dx, dy, px, py)
        IF dx > px  AND dy > py     THEN minPath <- {east, north};
        IF dx > px  AND dy < py     THEN minPath <- {east, south};
        IF dx > px  AND dy = py     THEN minPath <- {east};
        IF dx < px  AND dy > py     THEN minPath <- {west, north};
        IF dx < px  AND dy < py     THEN minPath <- {west, south};
        IF dx < px  AND dy = py     THEN minPath <- {west};
        IF dx = px  AND dy > py     THEN minPath <- {north};
        IF dx = px  AND dy < py     THEN minPath <- {south};
        IF dx = px  AND dy = py     THEN minPath <- {home};
        IF DEFAULT                  THEN routeXY <- home;
END MinimalPath;
```

Implementation of Rule Bases (ARON Method [Brockmann 90])

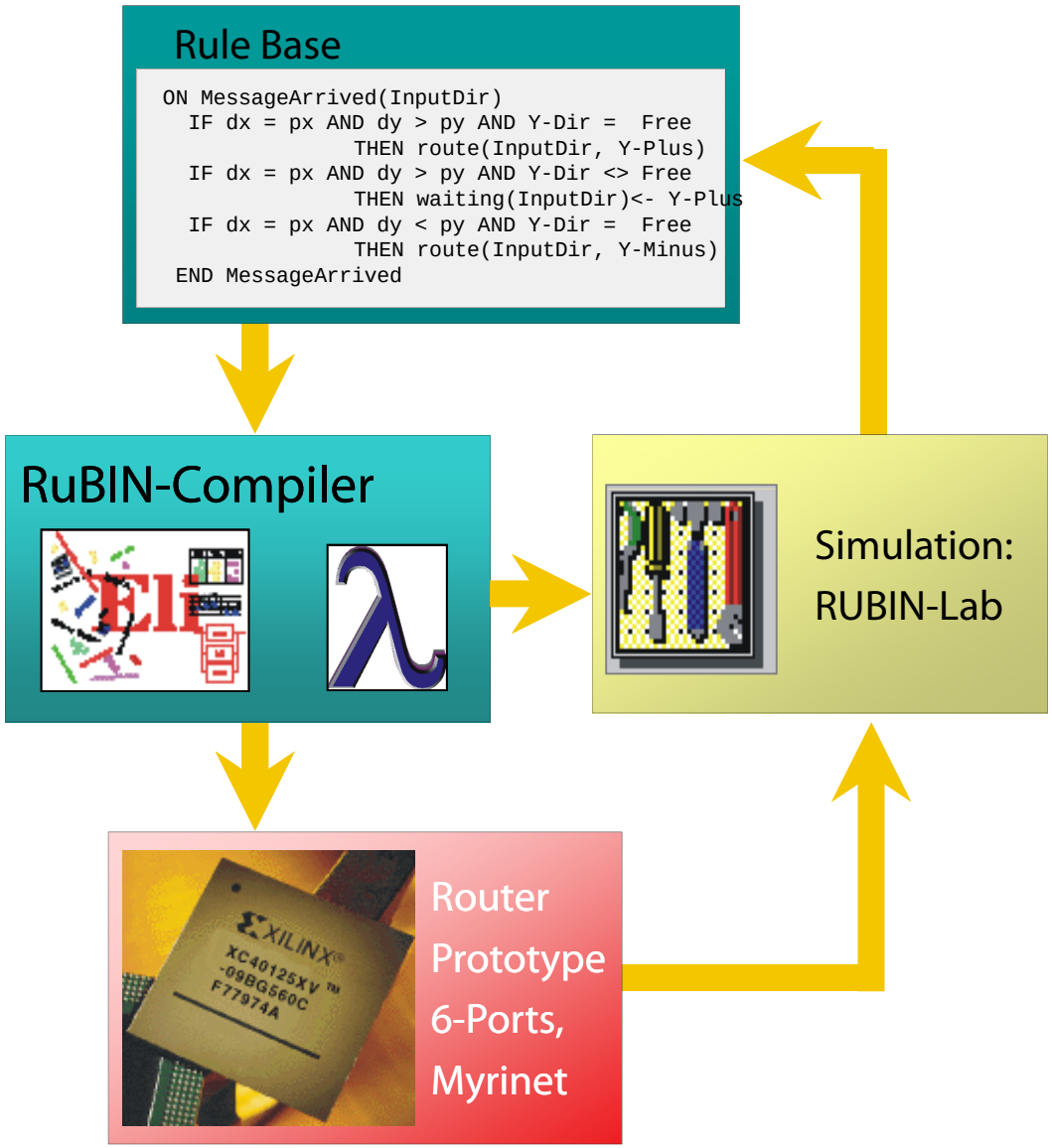
IF <premise>

THEN <conclusion>



Execution time in the range of one memory access (nsec-range)

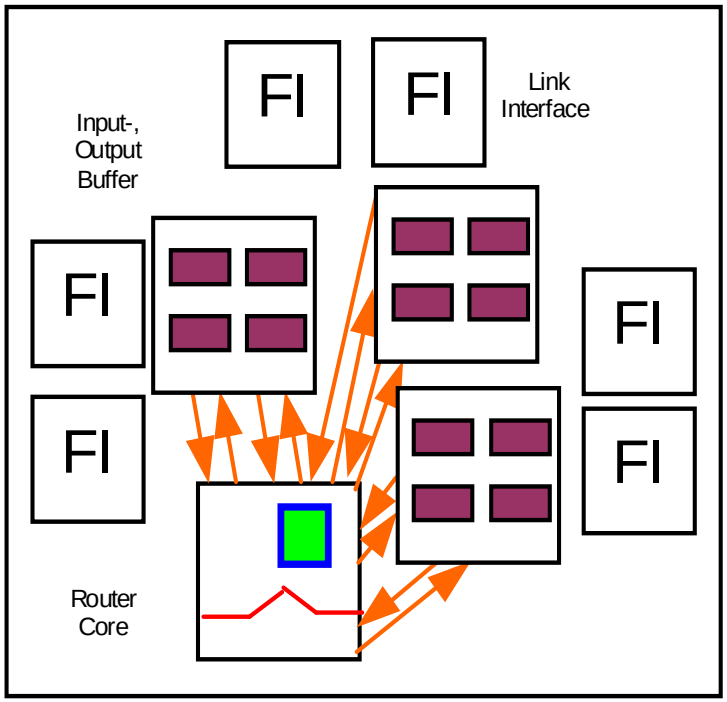
RuBIN Design Flow

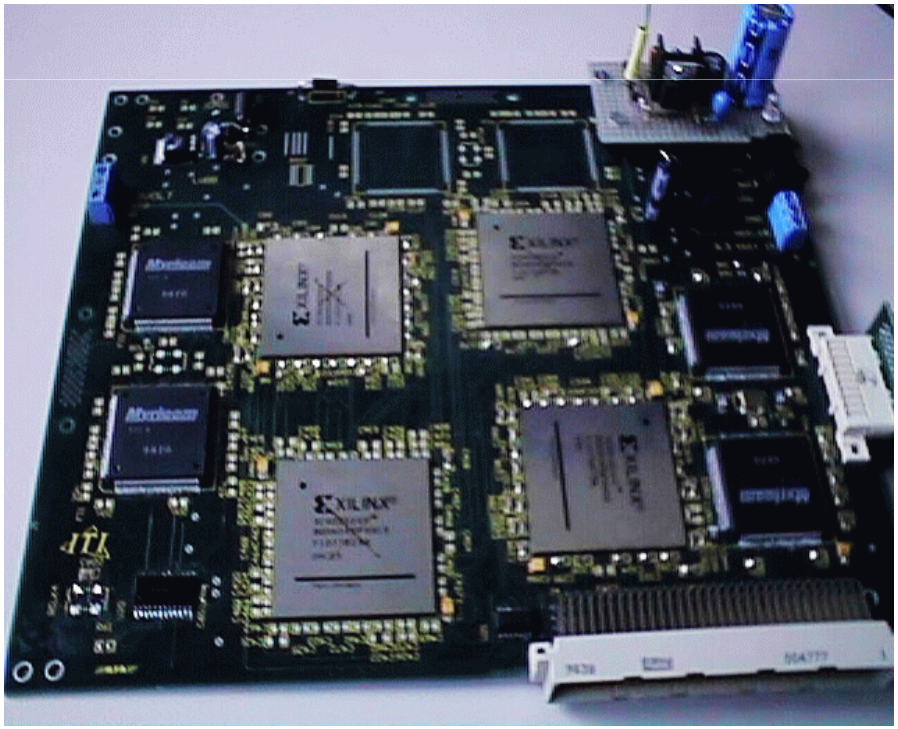


- Formal description of routing algorithms in a symbolic rule-based language
- Compilation into configuration or simulation data

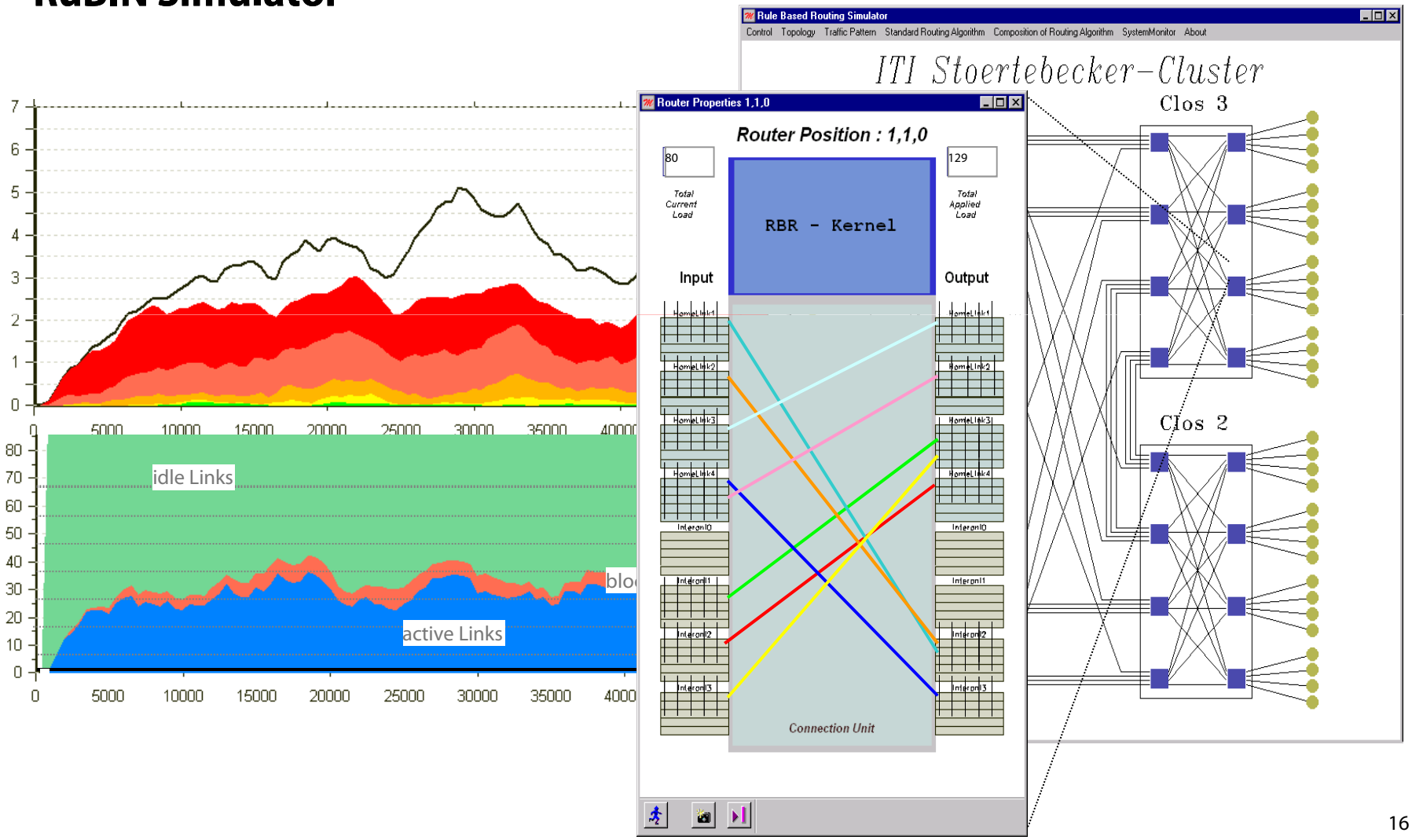
RuBIN Hardware Prototype

Interconnection Technology: Myrinet, 6 ports, 5 virtual channels
Standard components: 4 XILINX-FPGAs (about 1 Mio. gates),
6 FI-chips (Myrinet-Link)
Features: BGA-technology, 10 layer fine line board, 50 MHz system clock

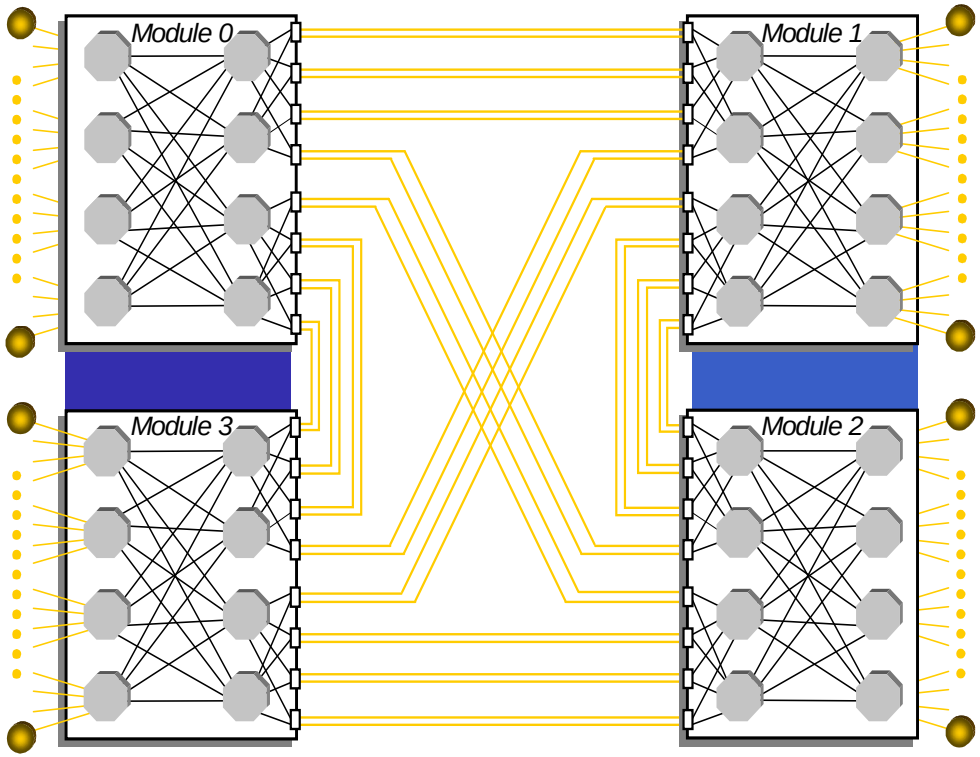




RuBIN Simulator



Störtebeker Network with Adaptive Routing

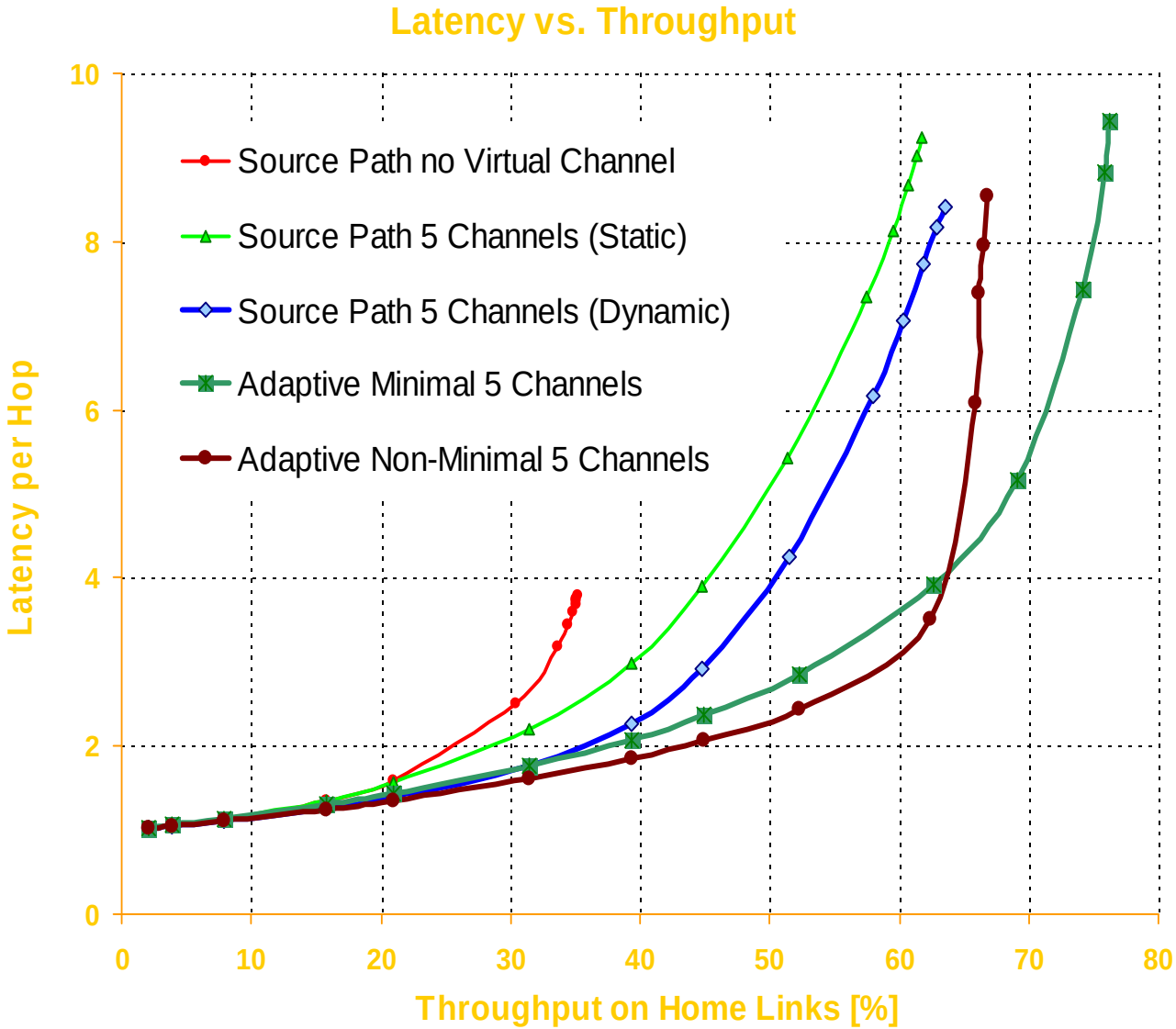


Router Nodes

Computation Nodes

- Deadlock-free routing along minimal paths
- Original version:
  - no virtual channels
  - non-adaptive path selection
- *Benefits of virtual channels?*
- *Adaptive routing along non-minimal paths?*

Simulation Results for Adaptive Routing

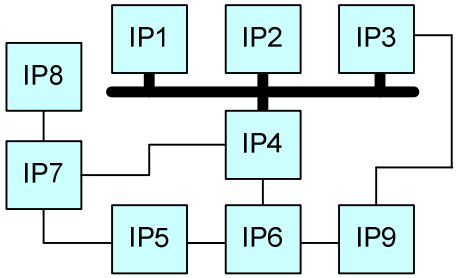


- Uniform message destinations
- Exponentially distributed message length (average 10 Flits)
- Adaptivity criterion: Number of channels in use per link

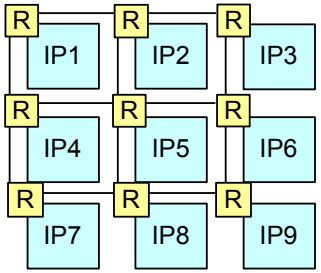
Topology-Adaptive Network-on-Chips (NoCs)

Motivation

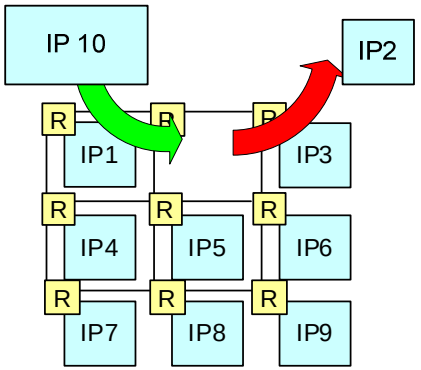
- "Classical" SoC-Design: *The SoC Nightmare*
  - Variety of dedicated interfaces
  - Poor separation between computation and communication.
  - Performance difficult to predict



➡ Usage of NoCs instead of bus-based architectures
  - Modular structure eases integration of different IP cores
  - optimized communication network for one particular SoC design



- SoC-Design with dynamically reconfigurable FPGAs
  - Problem: different sizes and communication requirements of IP-cores

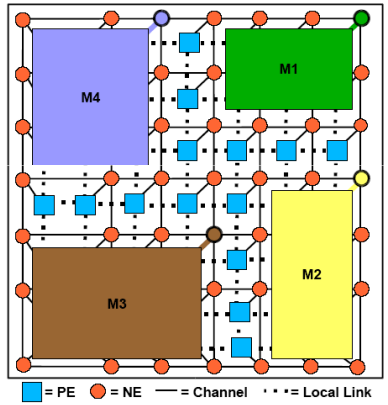


➡ **Adapt the structure of the NoC to the number and location of currently configured hardware modules**
  - Minimal number of switches, minimal latency
  - Eases an arbitrary placement of IP-cores of arbitrary size
  - High device utilization

CoNoChi

Topology Adative NoCs

- Only few work so far
  - Most often: configurable NoCs at **design time**
  - If configurable at **runtime**: restricted to dynamic adaptation of routing strategy or quality of service
- Example: **DyNoC** (Dynamic Network on Chip)
  (University of Erlangen, Germany, Bobda et al.)
  - NoC that can change its **topology** at runtime
  - Array of processing units, surrounded by a high number of switches
  - At runtime, switches can be disabled and their resources can be reused for dynamically inserted hardware modules



Source: University of Erlangen, Germany

CoNoChi: Configurable Network on Chip

Number of switches and their location can be adapted
→ minimal number of switches required

Reduced latency by minimizing the number of hops

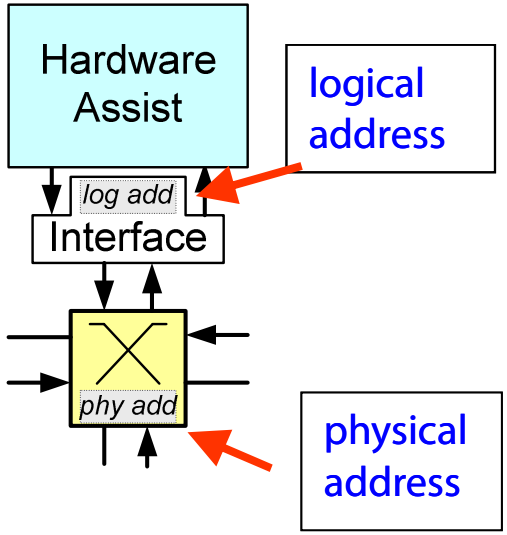
Multi-Layer protocol stack

Target Technology: FPGA

CoNoChi

Physical and logical addresses

- In general: only physical destination addresses
- In addition: logical addresses, evaluated by interface module
  - Physical addresses refer to specific switches at specific locations within the NoC topology
  - Logical addresses: refer to processing entities inside hardware modules



Routing

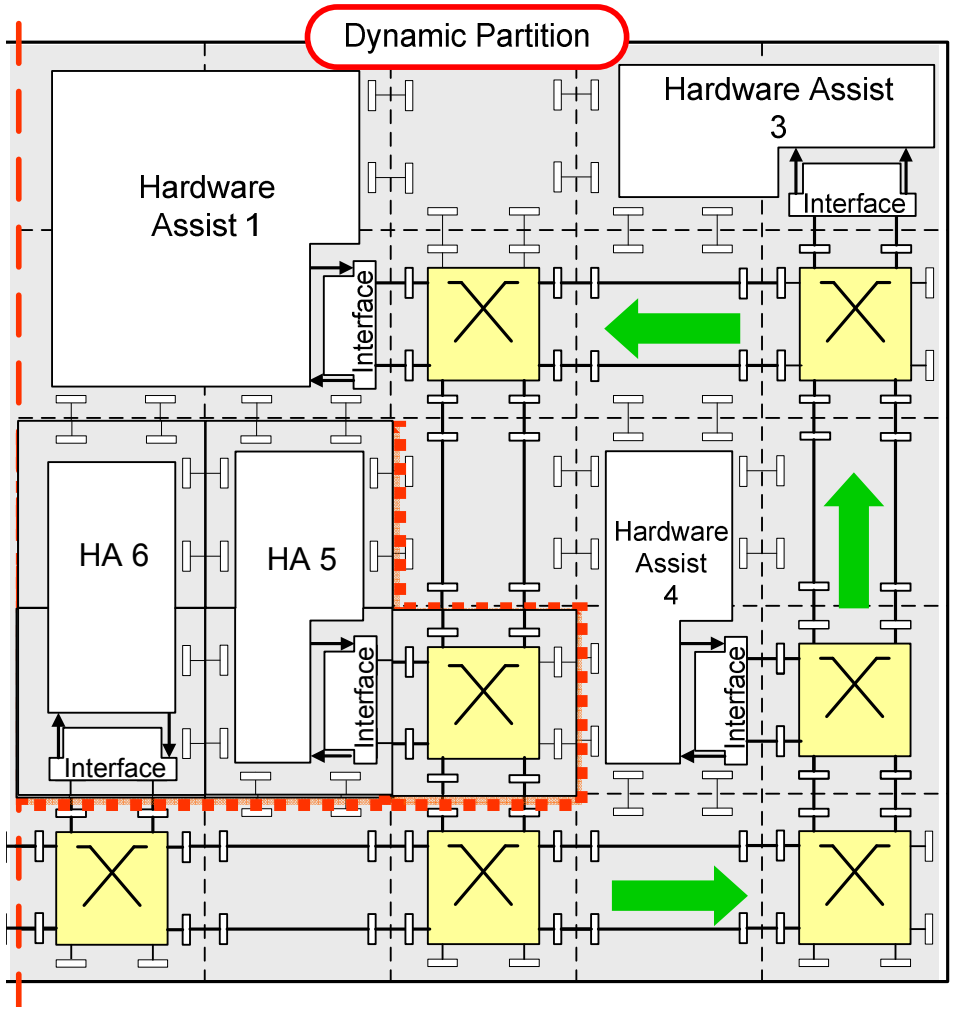
- Solely based on physical addresses
- Dynamic routing tables

CoNoChi

Topology Adaptation

- Network topology can be adapted at runtime
- **Coarse-grained tile** Merging/separation of neighbouring tiles

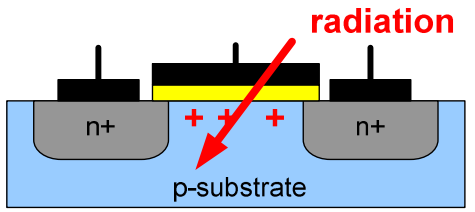
→ Provides space for modules of varying complexity



Fault Tolerance

Fault detection

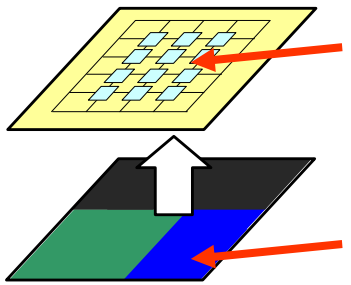
- Alive messages
- Test packets
- *Periodic configuration readback*



Single Event Upset (SEU)

Fault localization and correction

- Stepwise procedure using test packets
- Test against different assumptions
  - SEU in control registers → tile reset
  - SEFI → rewritting configuration
  - Permanent hardware fault → reorganization



change in logic

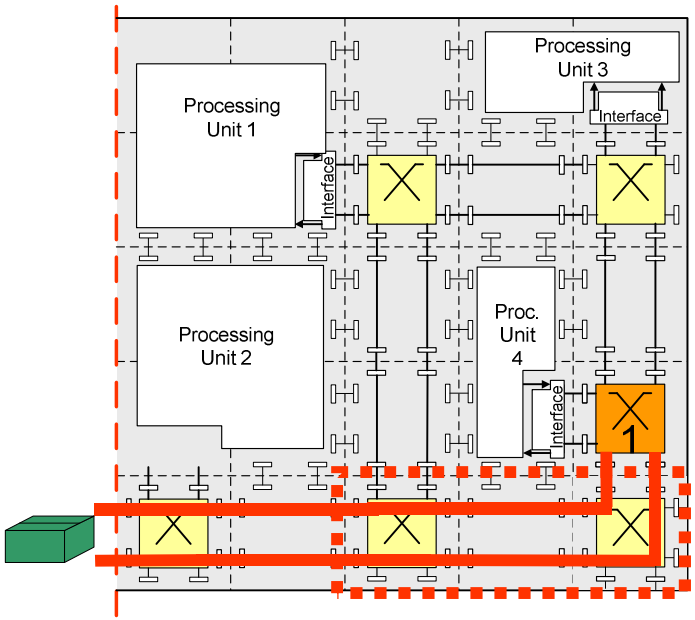
bit flip due to radiation in configuration memory

Single Event Functional Interrupt (SEFI)

Example: no alive message from switch 1

1. Identification of faulty segment

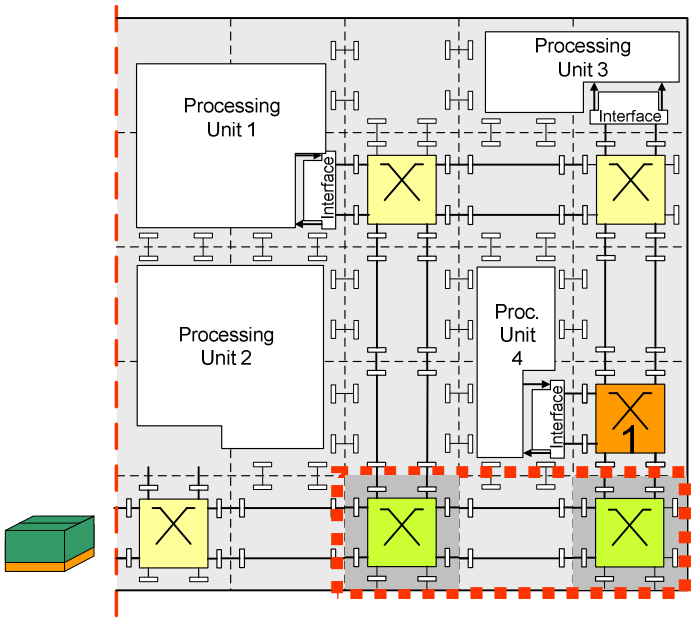
- Identify path under test
  Known by the reconfiguration manager
- Send test packets to all switch along the path under test
- If a test packet does not return correctly,
  faulty segment has been identified



Example: no alive message from switch 1

2. Assumption: SEU in control registers of switches or routing tables

- Reset switches in affected section
- Send new routing tables
- Repeat test



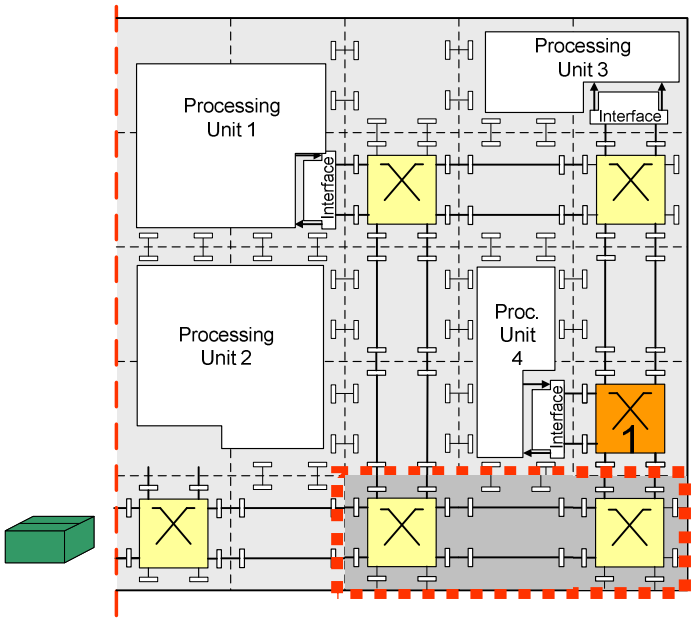
Example: no alive message from switch 1

3. Assumption: SEFI

- Readback configuration data for each tile and compare with reference
- In case of mismatch, reconfigure tile
  If tile contains a switch, send new routing tables
- Repeat test

Permanent hardware fault
→ reorganize system

Procedure takes time, does not cover all fault scenarios, yet is hardware efficient

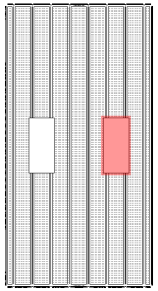


Implementation

CoNoChi

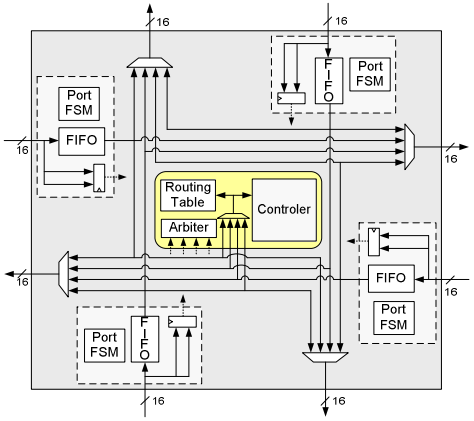
Theoretical homogeneous FPGA (based on Virtex-II Pro or Virtex-4)

- **Global control instance** of CoNoChi: realized in software



PowerPC Hardcores)

- **Switch**
  - Beside the physical destination address, all switches are identical
    - Only one configuration code for all switches is required
    - frame position and physical addresses have to be updated
  - After dynamic insertion, switch has to be initialized
    - Using the feature of reinitialization of Distributed RAM after a configuration



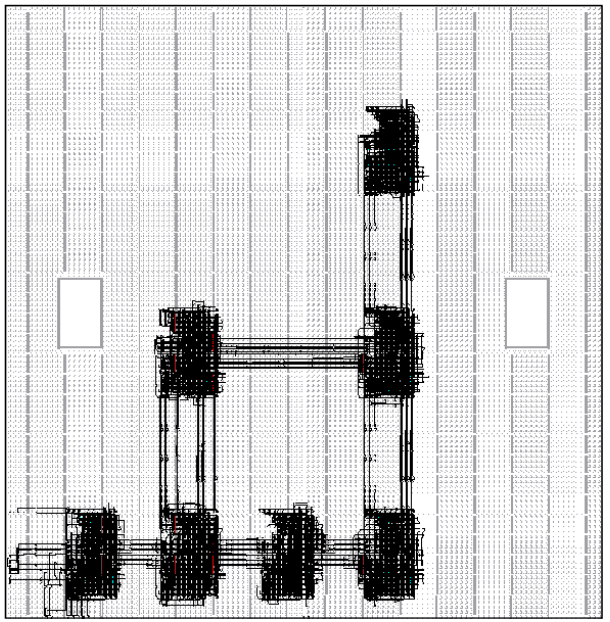
CoNoChi

Implementation

– Synthesis results for placed and routed switches
  - Virtex-II Pro 100, speed grade 6 and 44096 slices
  - In addition, 4 BlockRAMs per switch
  - Latency: 5 clock cycles

	Optimized for			
	Speed		Area	
Bit width	Slices	Freq. [MHz]	Slices	Freq. [MHz]
8	463	90	363	72
16	479	111	397	66
32	493	88	410	73

– Partial bitstream
  - 238 frames
  - 297 Kbyte
  - Reconfiguration time**: 6.1 ms**
    In the meantime all other preparations for updating the NoC can be done, e. g. computation of routing tables

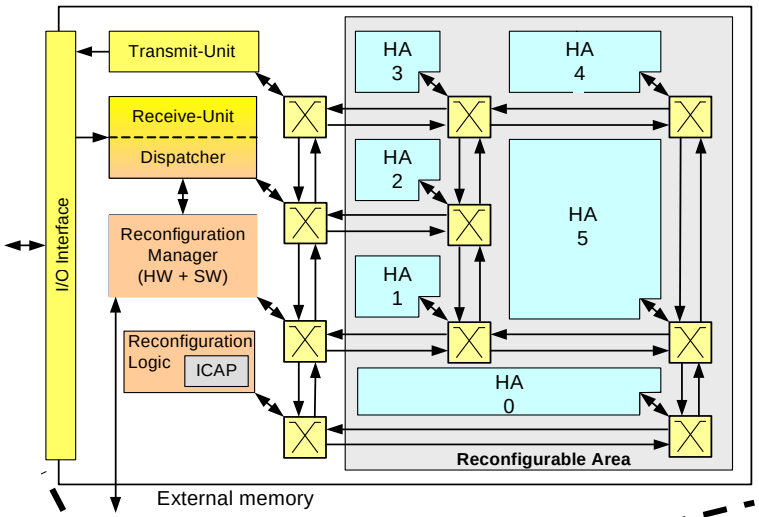


Switches in XC2VP100

Application Example: Network Co-Processor

CoNoChi

DynaCORE: Dynamically adaptable COprocessor based on REconfiguration



- FPGAs
  - Xilinx Virtex-II Pro
  - Virtex-4
- Network Processors
  - Intel IXP 2400
  - FlexPath (TU Munich)

Processed packets

Packets for payload processing

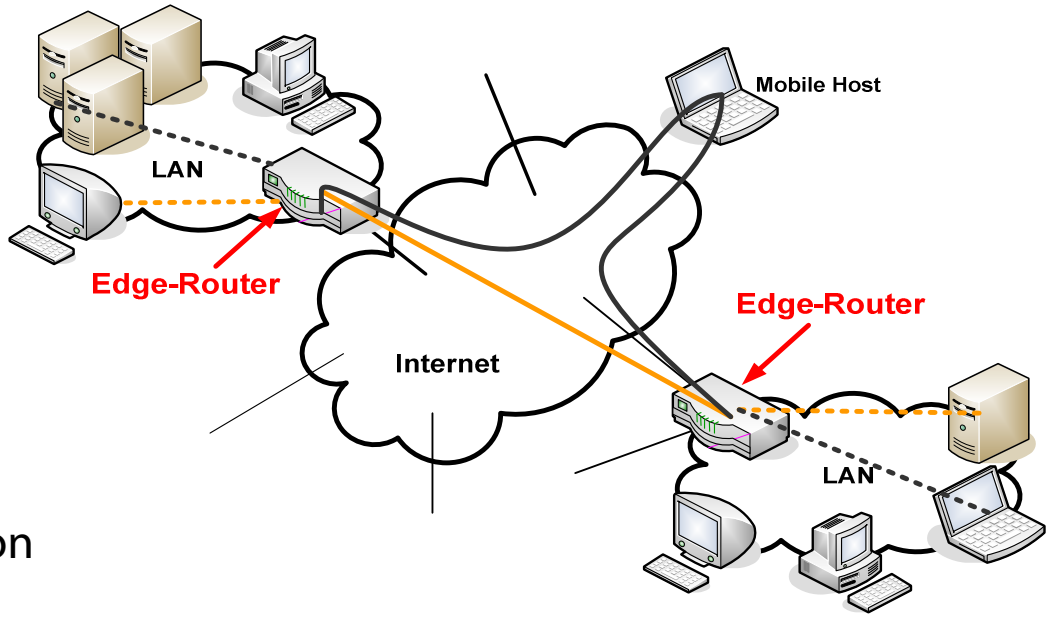
Local Area Network

Backbone

CoNoChi

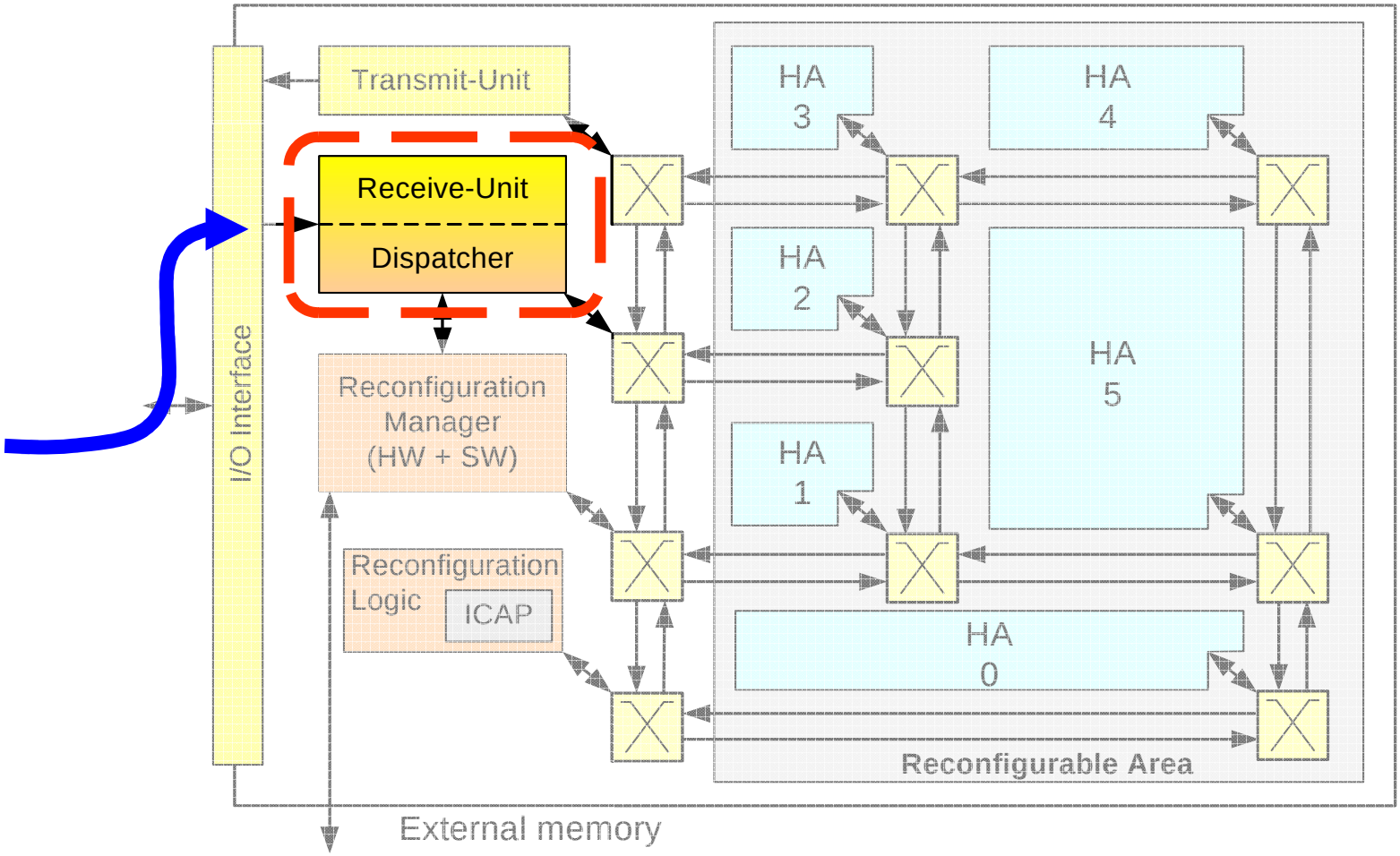
DynaCORE Application Area

- Edge-Routers
  - Device that routes data packets between one or more local area networks (LANs) and a backbone network
  - Virtual Private Networks
- High data rates
  - Today's link speed:
    - 10 Gb/s (OC-192)
    - 40 Gb/s (OC-768)
  - 200-3000 instr per packet
- Packet Processing Tasks
  - En-/Decryption
    - AES
    - DES / 3DES
  - Compression
  - Network Intrusion Detection

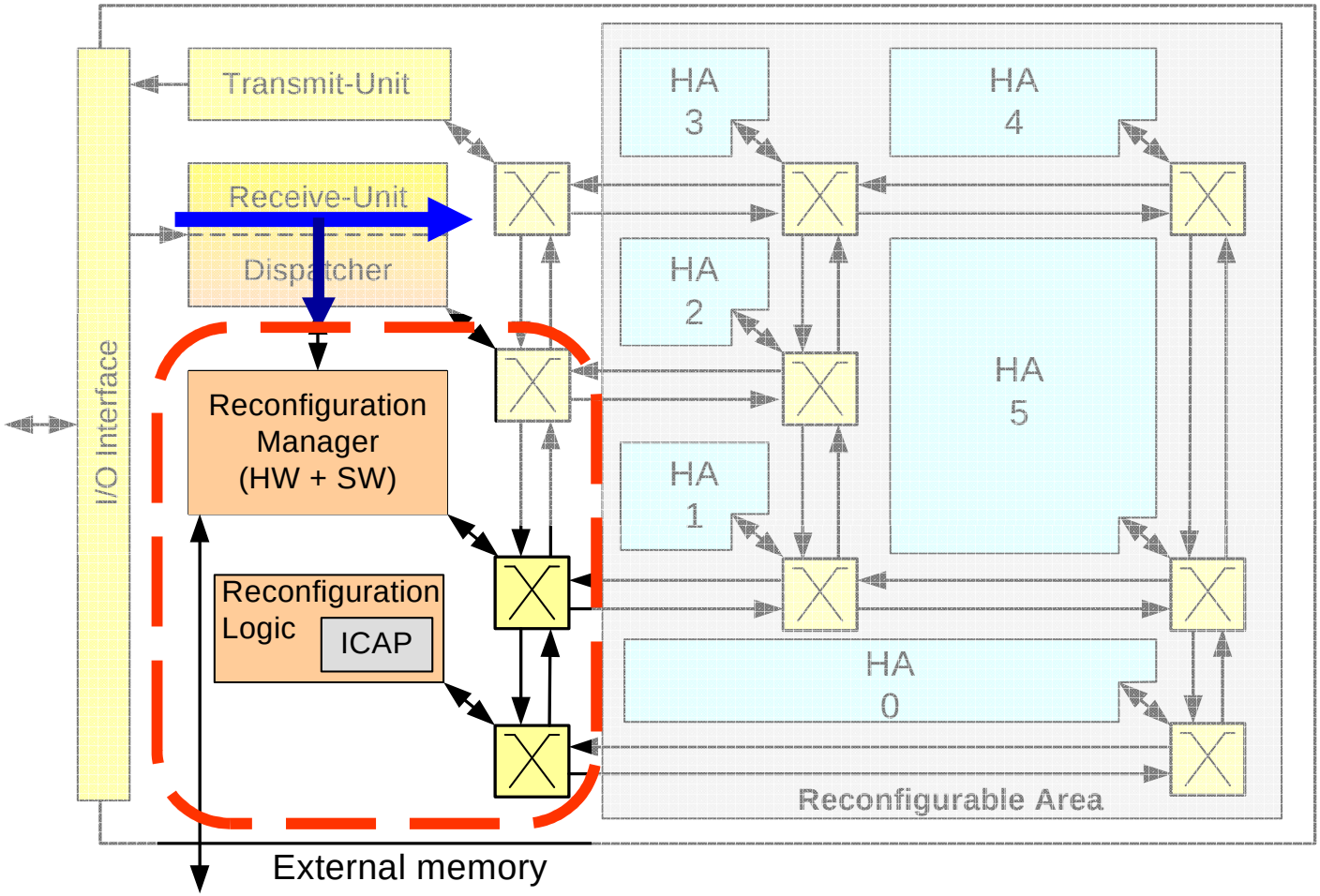


CoNoChi

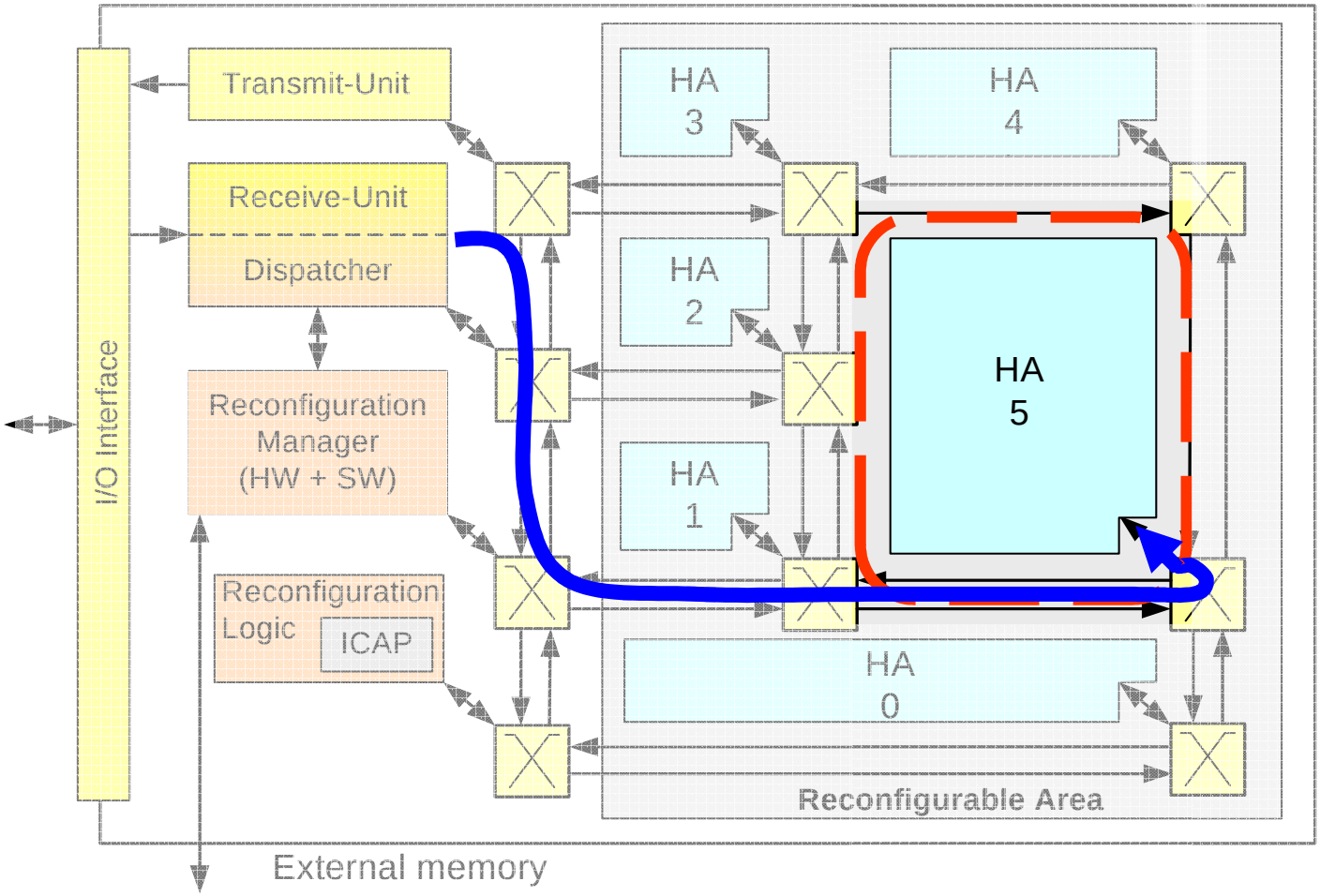
- **System Overview**



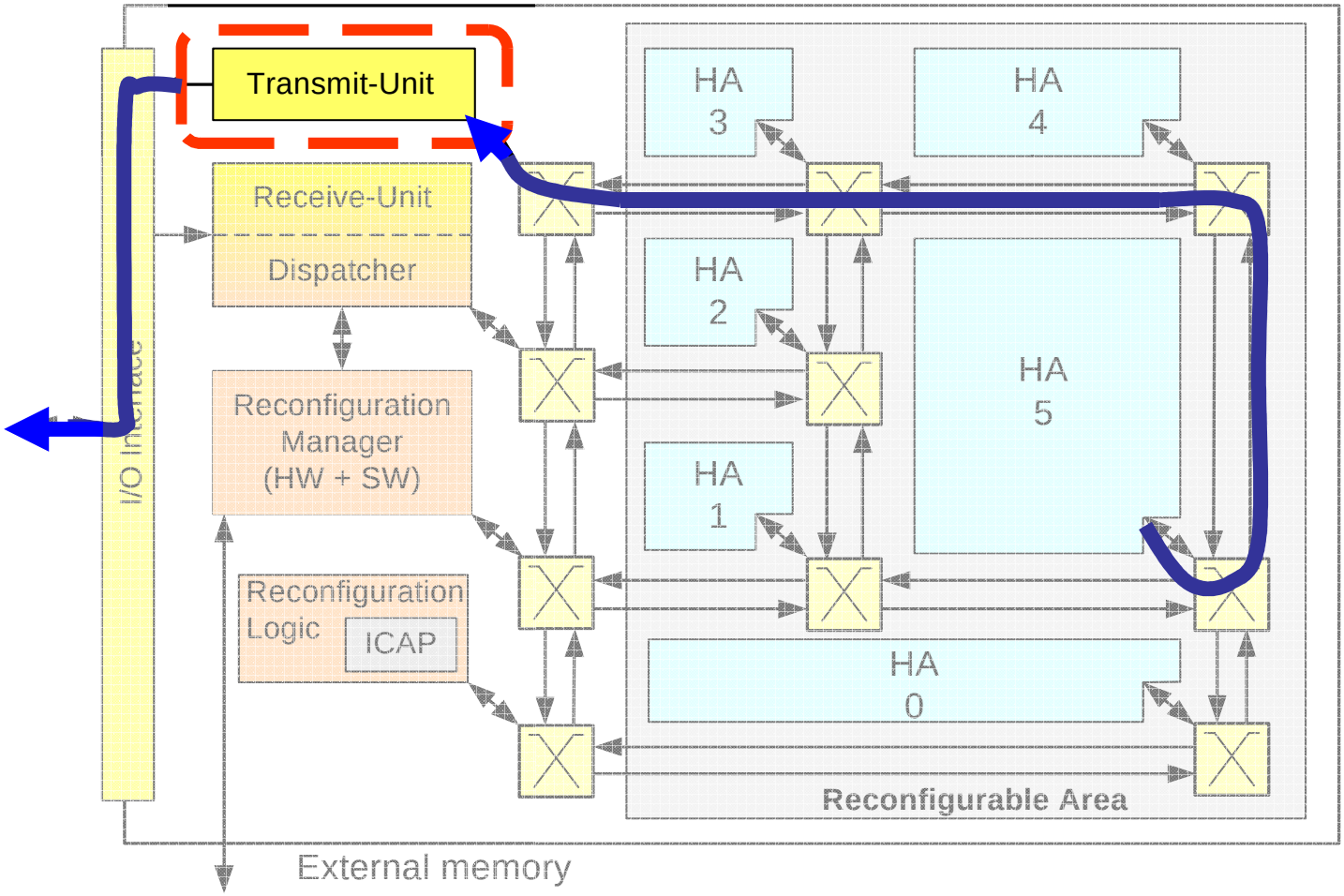
CoNoChi


Transmit-Unit
Receive-Unit
Dispatcher
I/O Interface
Reconfiguration Manager (HW + SW)
Reconfiguration Logic
ICAP
HA 3
HA 4
HA 2
HA 5
HA 1
HA 0
Reconfigurable Area
External memory

CoNoChi

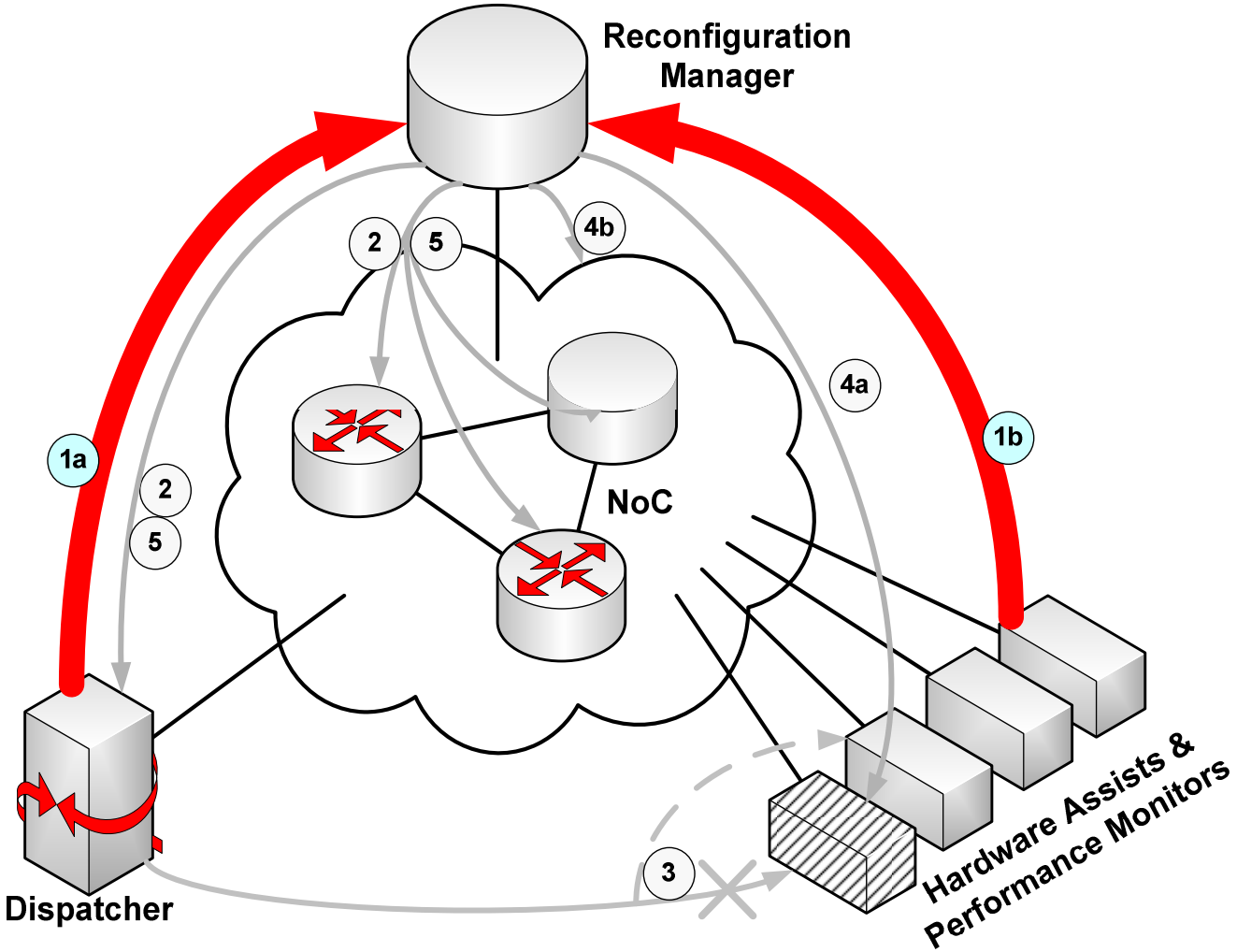

Transmit-Unit
Receive-Unit
Dispatcher
Reconfiguration Manager (HW + SW)
Reconfiguration Logic
ICAP
I/O Interface
External memory
HA 3
HA 4
HA 2
HA 5
HA 1
HA 0
Reconfigurable Area

CoNoChi


Transmit-Unit
Receive-Unit
Dispatcher
Reconfiguration Manager (HW + SW)
Reconfiguration Logic
ICAP
I/O Interface
HA 3
HA 4
HA 2
HA 5
HA 1
HA 0
Reconfigurable Area
External memory

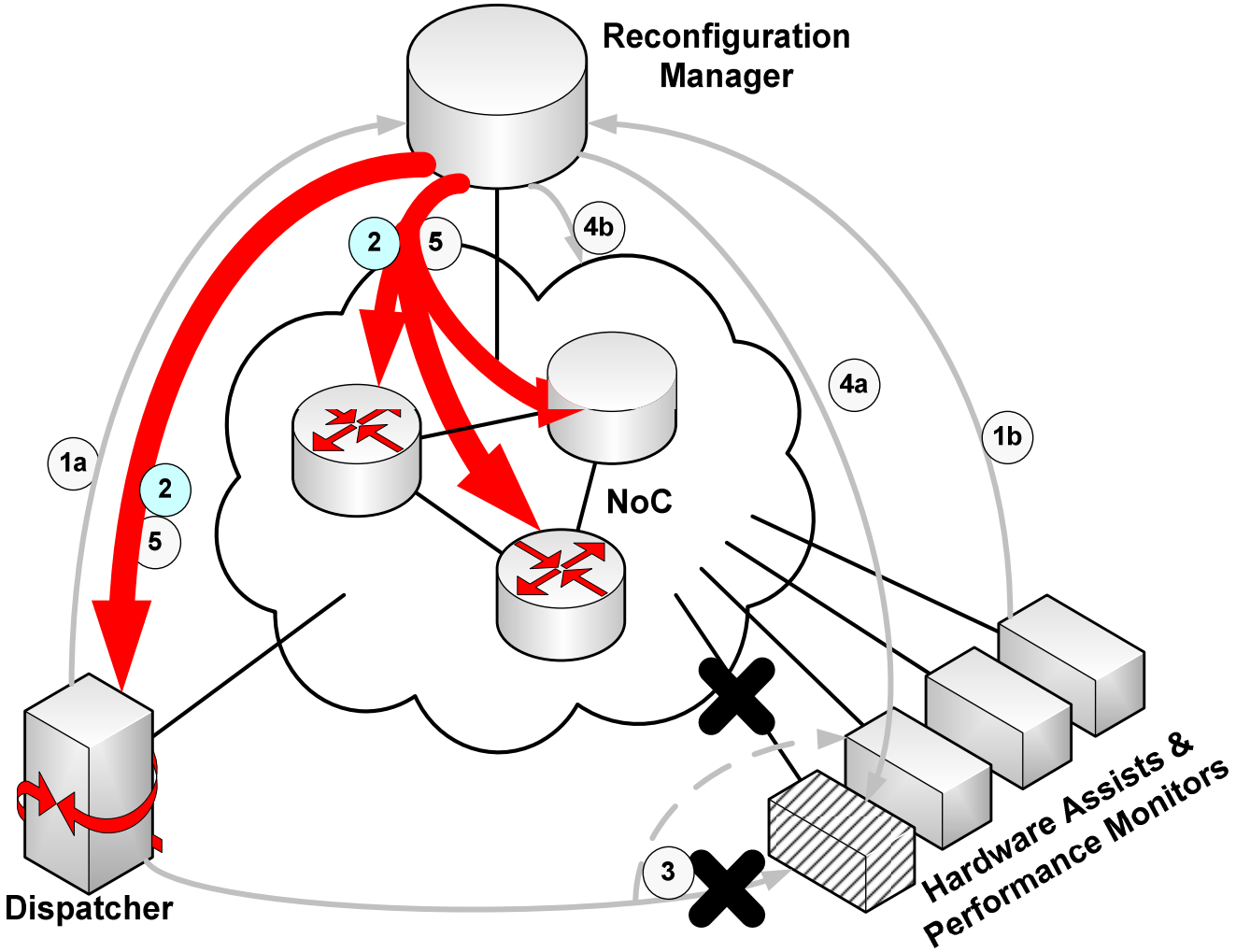
- Exchange of hardware assist

CoNoChi



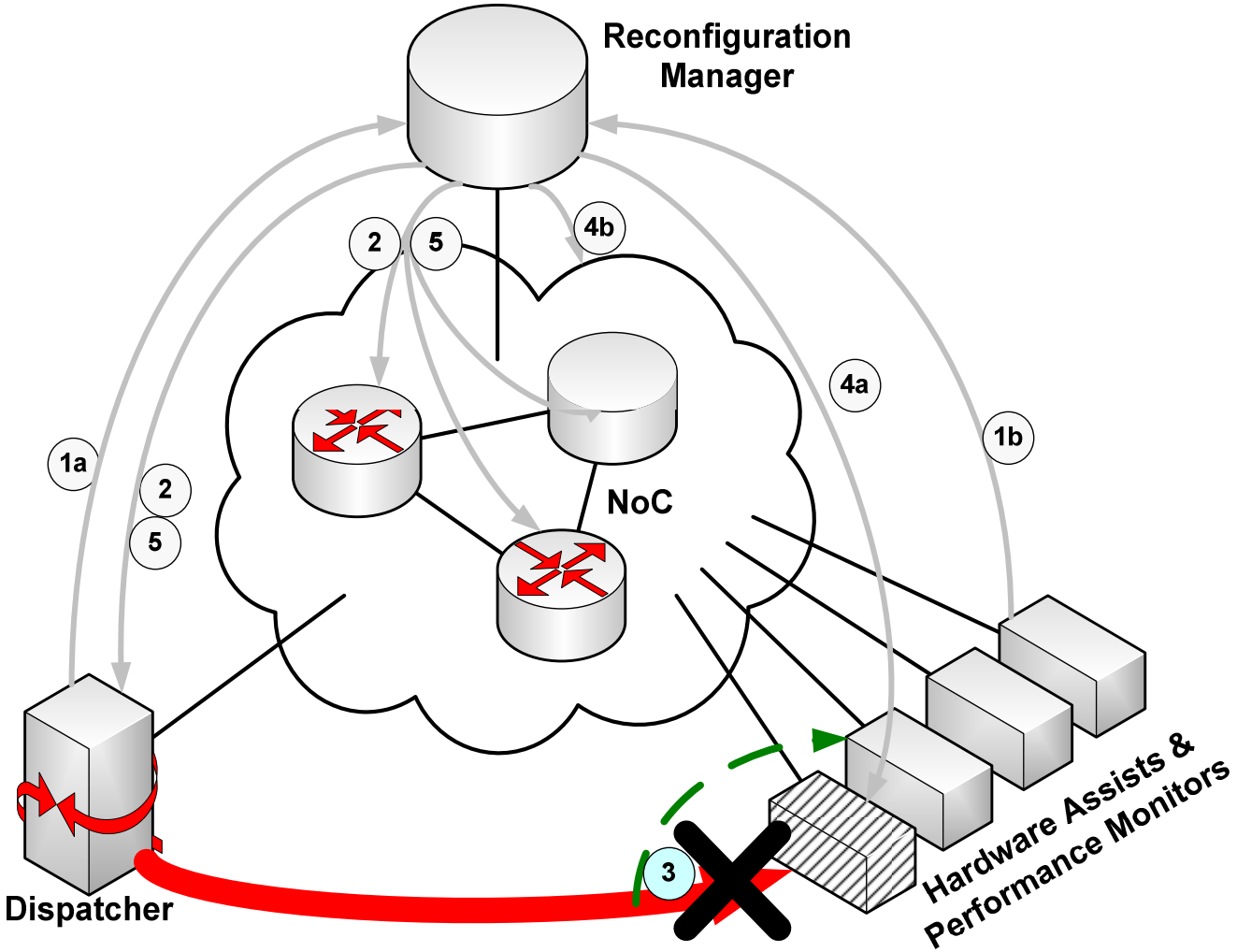
- Exchange of hardware assist

CoNoChi



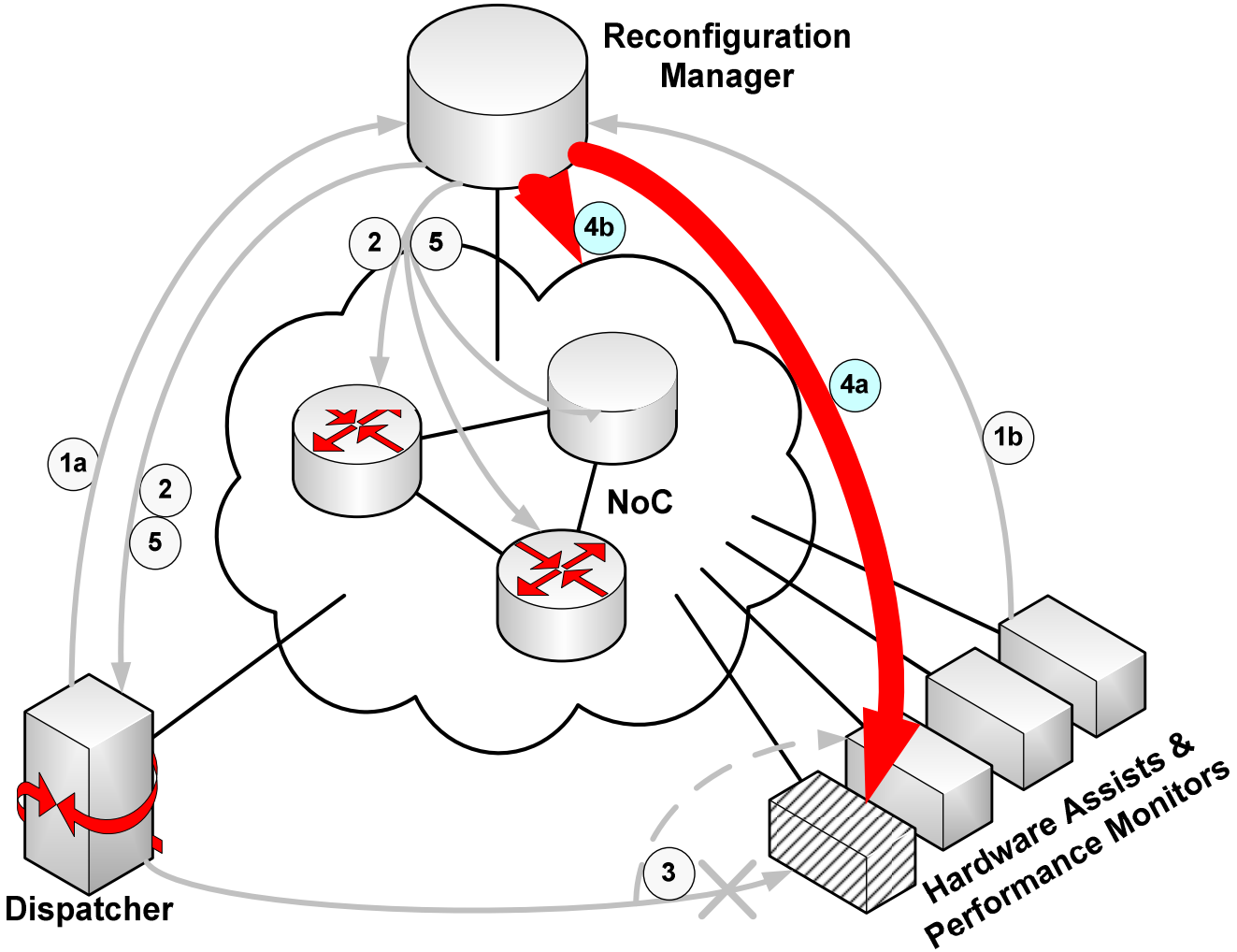
- Exchange of hardware assist

CoNoChi



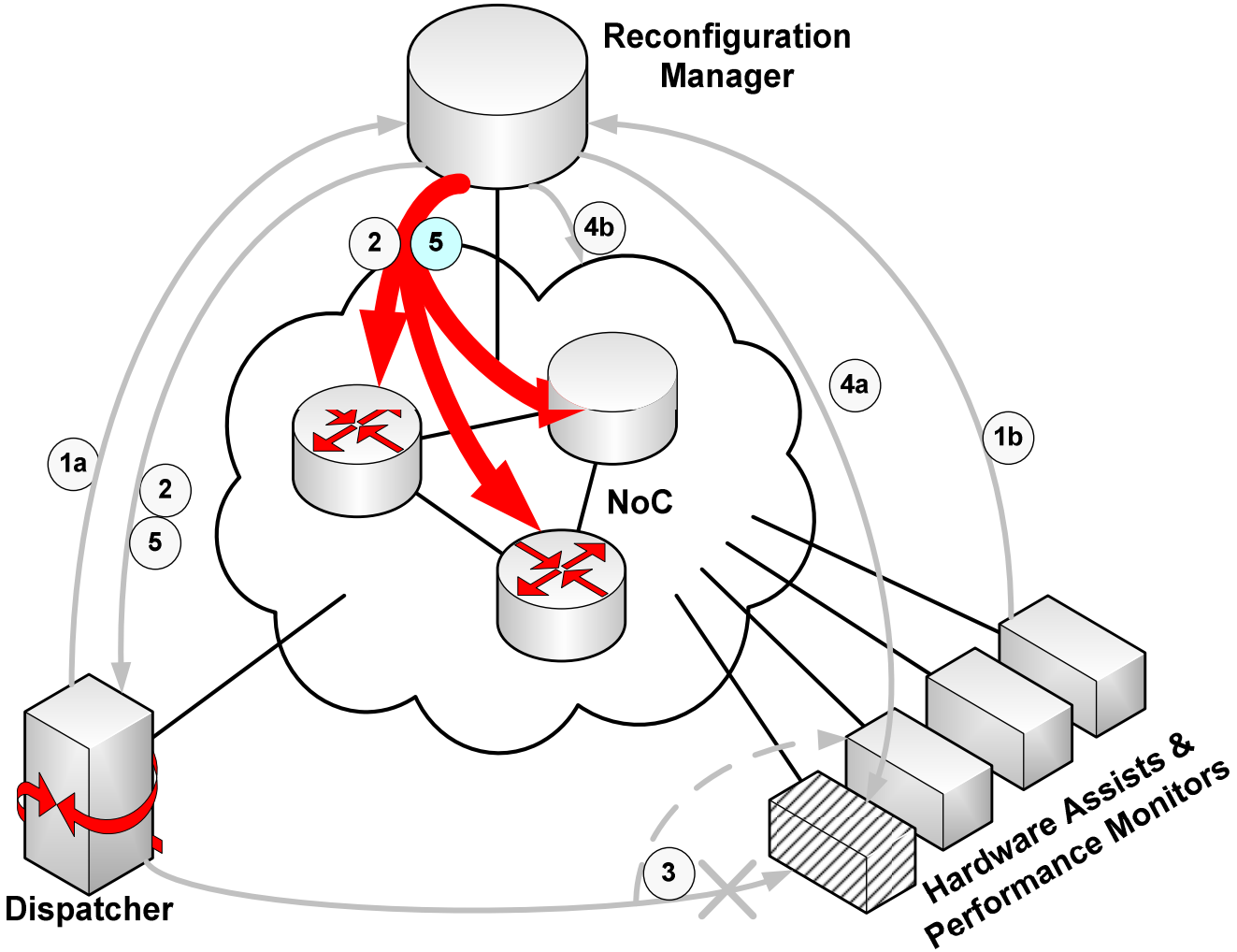
- Exchange of hardware assist

CoNoChi



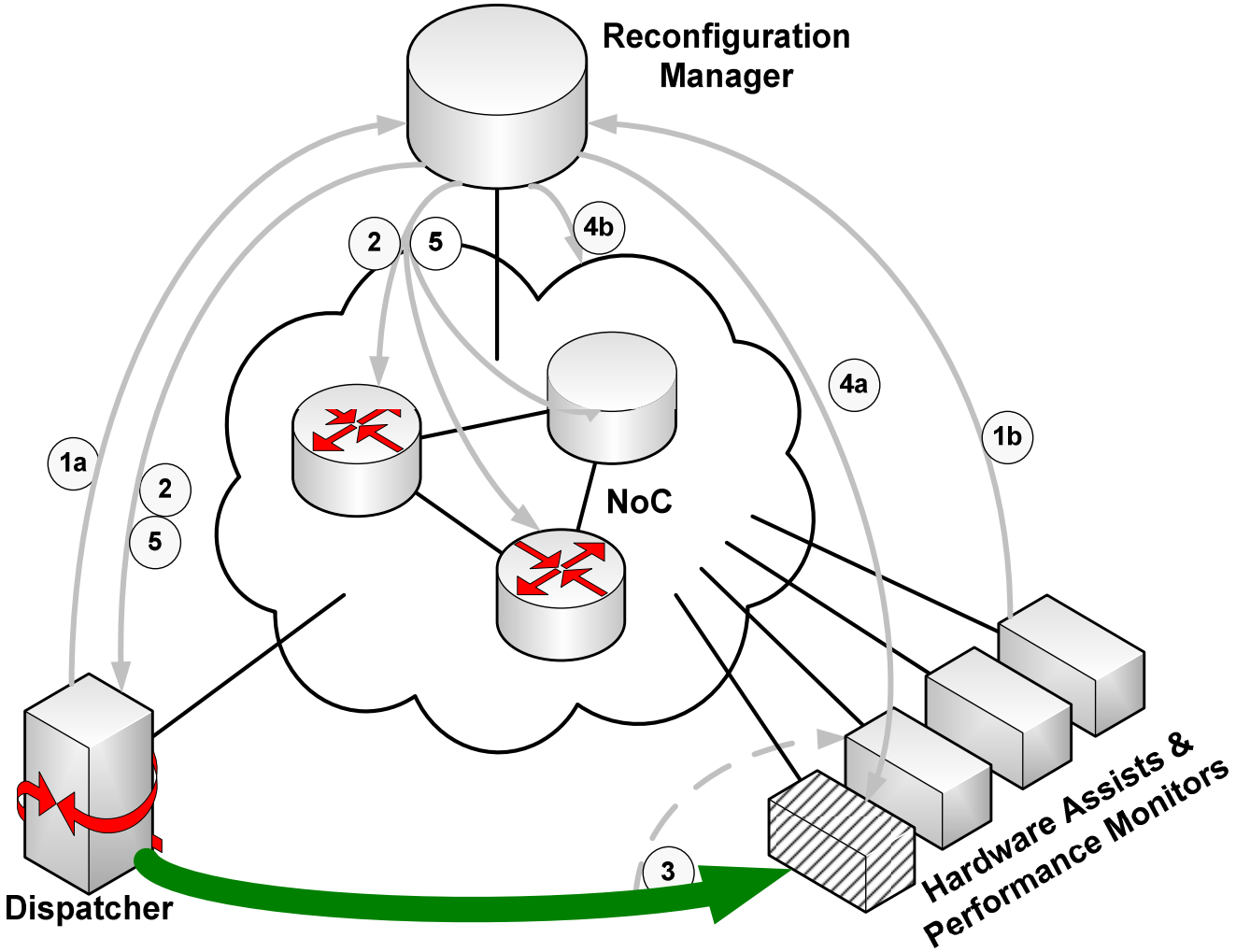
- Exchange of hardware assist

CoNoChi



- Exchange of hardware assist

CoNoChi



Summary

CoNoChi: NoC architecture for dynamically reconfigurable hardware platforms

- Topology of network can be adapted during runtime
- Enables the usage of IP cores of different sizes
- Minimum number of hops and switches, low hardware overhead
- Layered protocol
  Support of physical and logical addresses

Acknowledgement:

This work was funded in part by the German Research Foundation with the priority program 1148 DFG Schwerpunktprogramm 1148

Team: Carsten Albrecht, Roman Koch,
Jürgen Foag, Thilo Pionteck

NoC Adaptation to Selected Data Streams

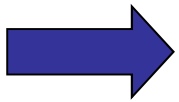
Motivation

Drawback of NoCs: end-to-end latency

- Depending on the number of hops
- Packets have to compete for router resources

Diverse approaches for reducing the latency of individual routers

- Speculative/parallel execution of router pipeline stages
- Pre-computing routing decisions
- Adaptive routing schemes



large hardware overhead

Not all data streams need to be prioritized in the same way!

Application scenarios:

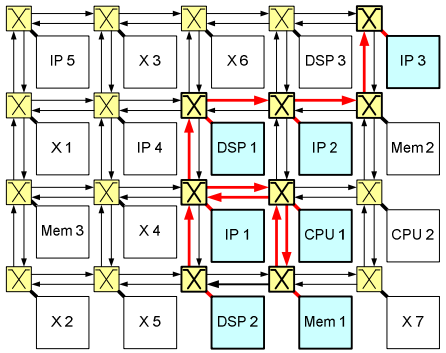
- Shared-memory chip multiprocessor system (CMP) [9]
  - Latency-critical messages: protocol requests, acknowledgment
  - Only 18% of the overall traffic are latency critical
- In general: devices with diverse use cases, e.g. mobile phones
    - Phone calls
    - Video encoding/decoding

Review focuses on NoCs for:

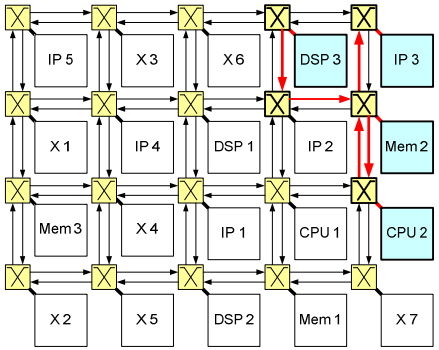
- Semi-static data streams:
- Runtime reconfigurable system
- Regular NoC architectures

[9] Z. Li, J. Wu, L. Shang, R. Dick, Y. Sun: „Latency Criticality Aware On-Chip Communication", DATE 2009

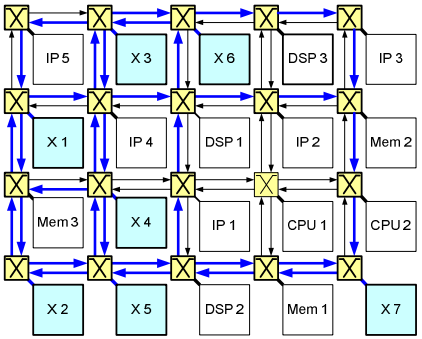
Application example



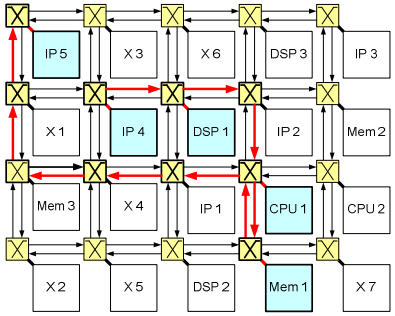
H264 encoding



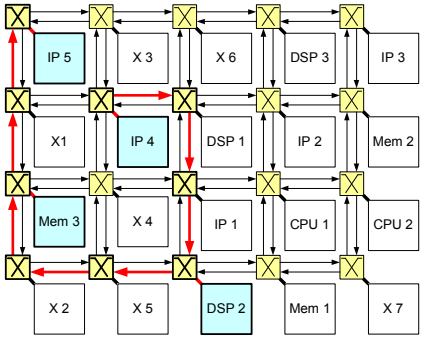
MP3 encoding



Temporary data streams

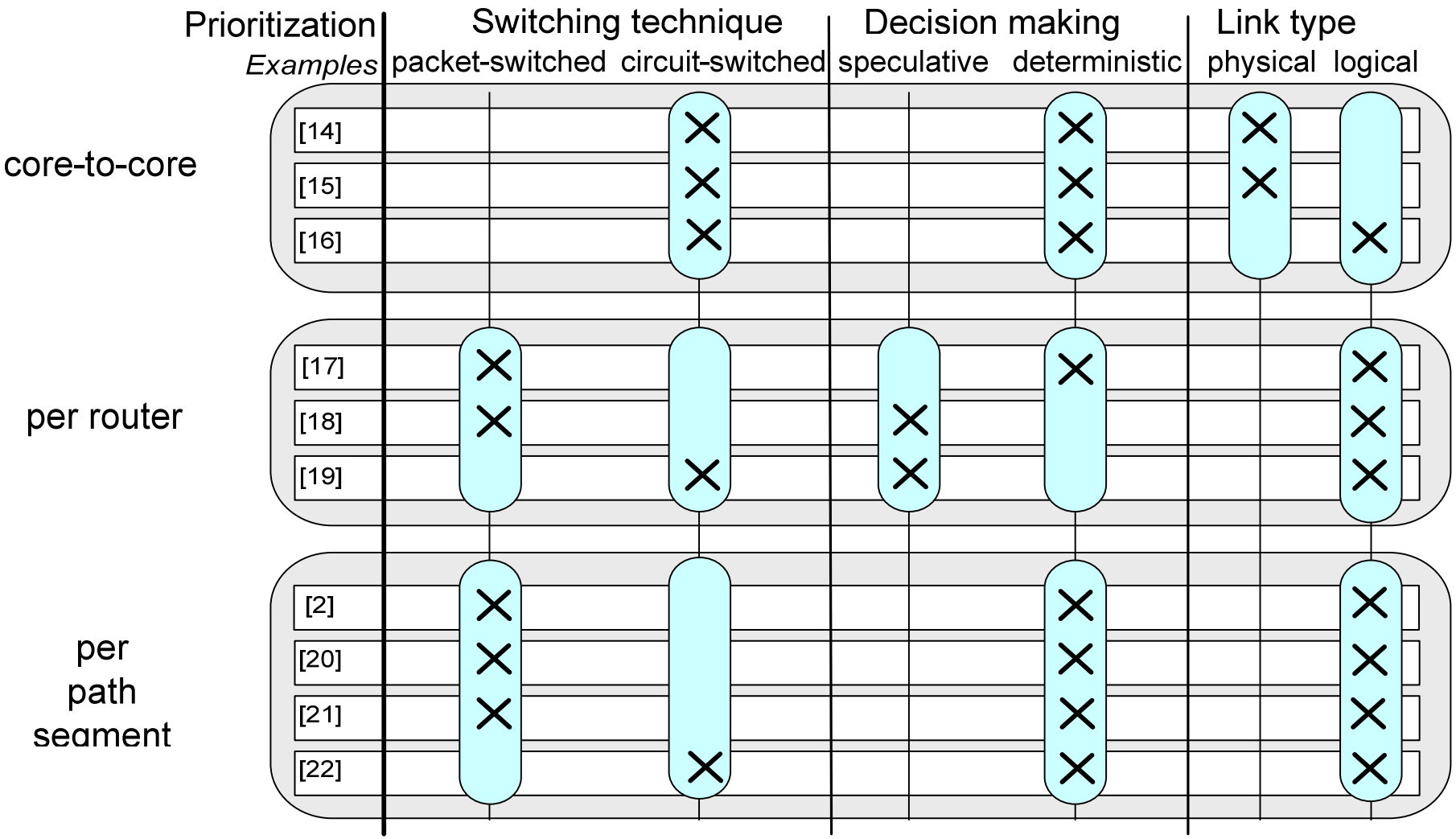


H264 decoding

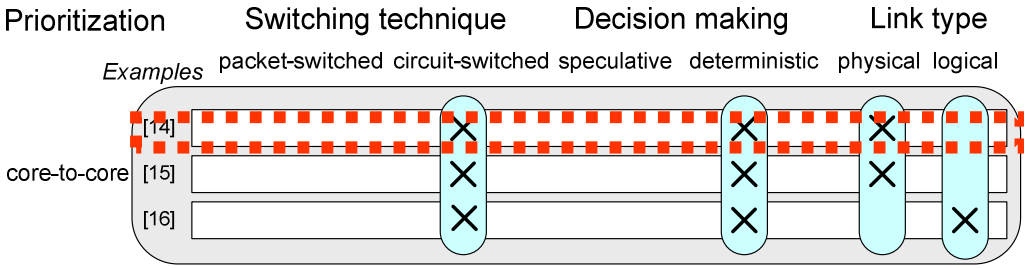


MP3 decoding

Design Space

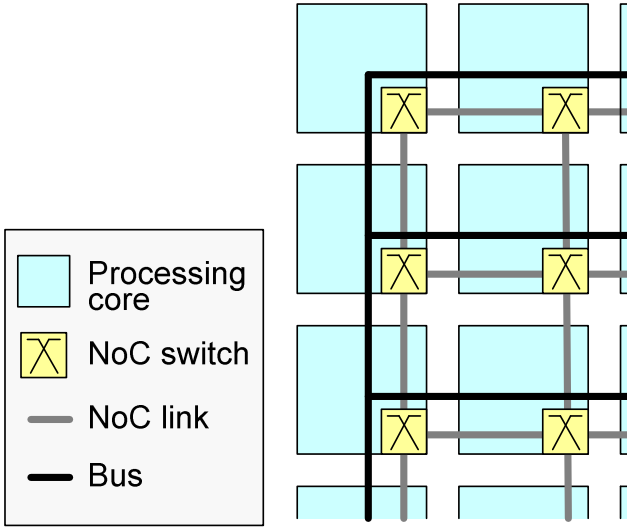


Core to core prioritization

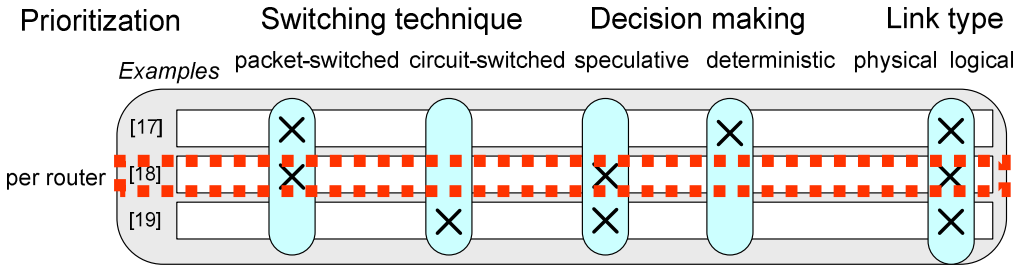


Bus Enhanced NoC (BeNoC)

- Packet switched NoC + low latency bus
- Bus
  - Global latency critical control signals
  - Broadcast
  - Multicast
  - NoC management
- NoC
  - High throughput data
- 300% speedup for application example
- Bus requires 0.1% of total area



[14] R. Manevich, I. Walter, I. Cidon, A. Kolodny: „Best of Both Worlds: A Bus Enhanced NoC (BeNoC)“ NoCs 2009



Per router prioritization

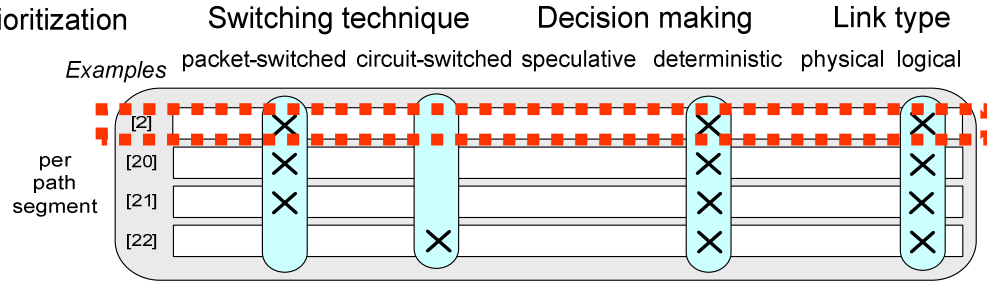
Speculative Forwarding

Prediction Router

- Output link is predicted and switch arbitration is speculative done for each idle input
  - Single cycle router transfer if prediction hits
  - Wrongly forwarded flits are masked at the output → eliminates dead flits
- Several prediction schemes
- Latency reduction between 30.7% and 48.2%

[18] H. Matstani, M. Koibuchi, H. Amano, T. Yoshinaga: „Prediction router: Yet another low latency on-chip router architecture“, HPCA 2009

Prioritization per path segment

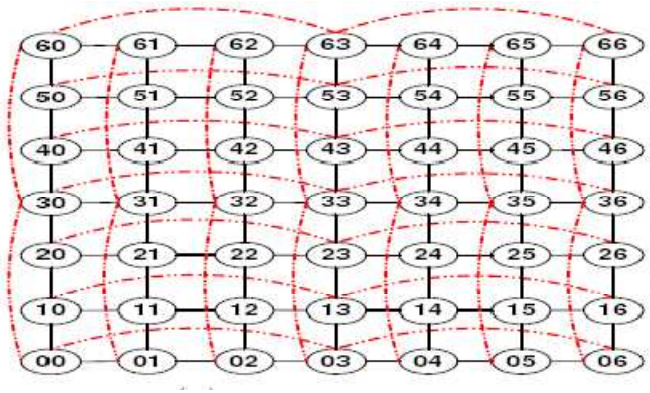


In general:

- Dedicated virtual channels for prioritized traffic
- Router pipeline stages are bypassed

Express Virtual Channels

- Virtual channels are partitioned between
  - Normal VC
  - Express VC (dynamic)
    - Connect routers along a dimension
    - Length of bypass is configurable
      - Requires extra network
- Latency reduction of 84%
- Power reduction of 38%



Source: [2]

[2] A. Kumar, L.-S. Peh, P. Kundu, N. K. Jha: „Express Virtual Channels: Towards the Ideal Interconnection Fabric“, Computer Architecture Symposium, 2007

Discussion

Core-to-core prioritization:

- highest area increase
  - Caused by additional physical communication structure
    - Bus
    - Long-range links
    - Logic for setting up virtual topology
- Near optimal latency
- Bypassing routers leads to power savings
- Increase system throughput
- Limited flexibility
  - Only short messages
  - Link insertion at design time

Discussion

Prioritizing on per hop basis

- Shows best adaptability
- Reduced optimization potential
  - At least some router pipeline stages have to be passed
  - Increased hardware effort due to complex router pipeline structures

Speculative forwarding

- Power inefficient
- Dead flits
- Latency critical traffic can even be delayed

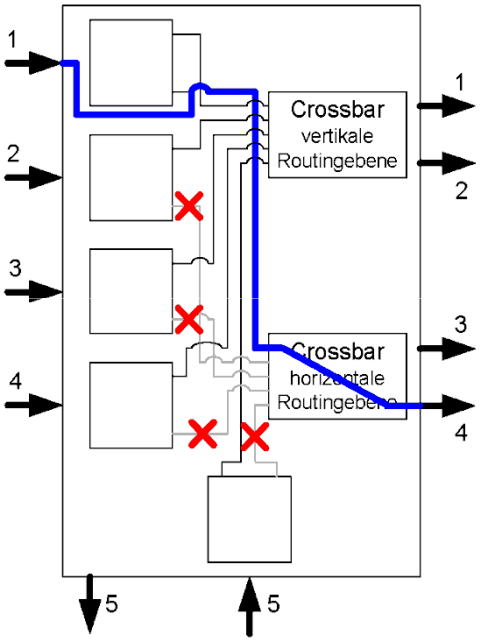
Summary

- Categorizations scheme for NoCs with prioritization of selected data streams
- Presentation of most relevant NoC designs in the selected categories
- Discussion of pros and cons of presented NoCs for runtime adaptive systems

More detailed discussion in the paper:

Pionteck, T.; Osterloh, C.; Albrecht, C.: *Latency Reduction of Selected Data Streams in Network-on-Chips for Adaptive Manycore Systems*. 28th IEEE Norchip Conference, 1-6, Tampere/Finnland 2010

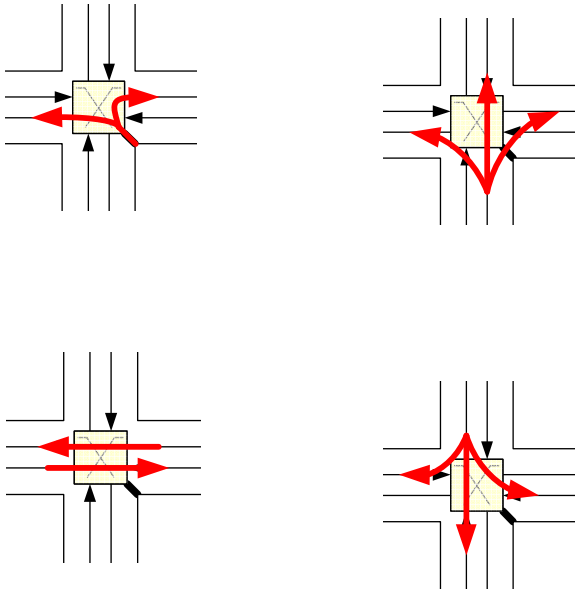
Semi-Static Data Streams



Switch architecture

- Special crossbar design
- Bypasses for semi static data streams

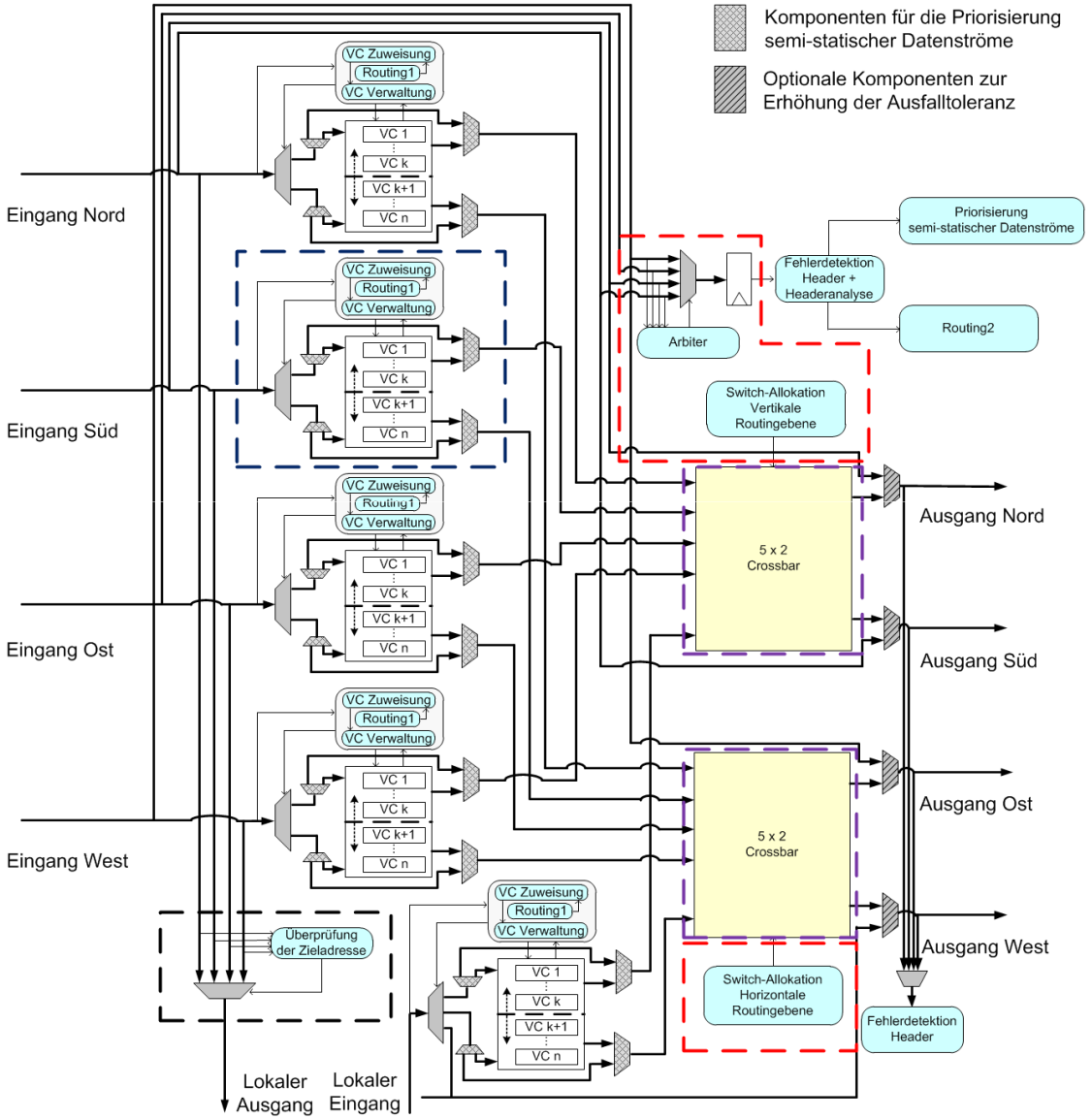
Fault-Tolerance



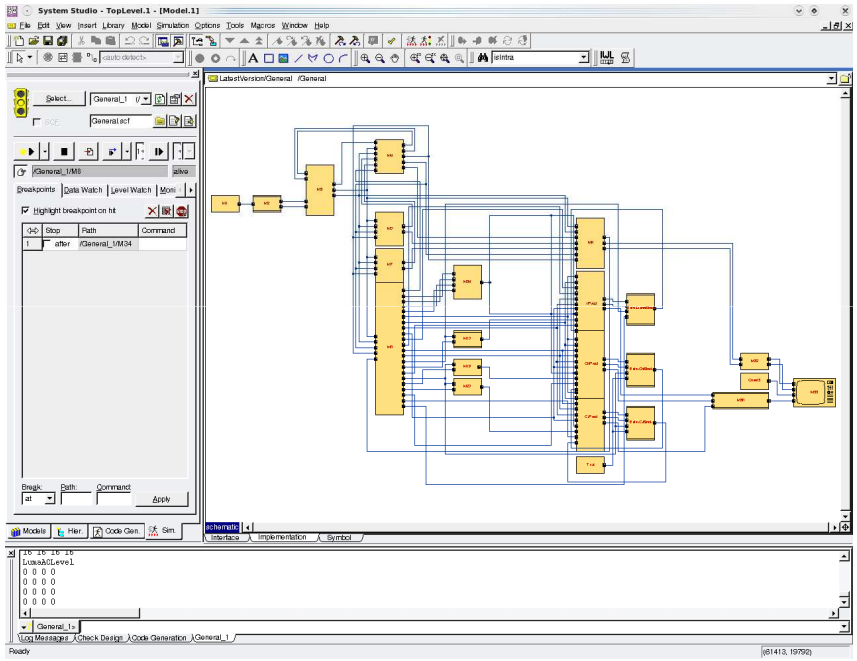
Routing options in case of blocked horizontal crossbar

First Sample Router Design [Pionteck/Samann 2010]:

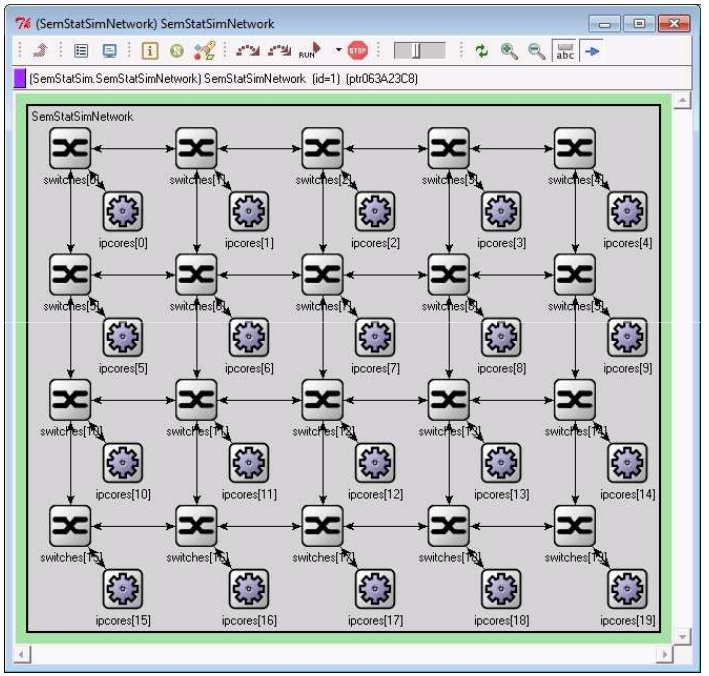
- Wormhole Routing
- Virtual-Channels
- 2 -Stage XY-Routing
- ON/OFF Flow Control
- Early Ejection
- Semi-Static Data Streams
- Fault Tolerance



Simulation environment(s)



Synopsys System Studio
Realistic test data



OMNeT++
NoC simulation

Conclusions and Outlook

Interconnection networks are acrucial part of a parallel computer which widely determine its main features like performance and fault-tolerance.

SANs

Today well established and standardized (e. g. Infiniband)

Advanced routing schemes like adaptive routing supported

NoCs

Still subject to intensive research

Restrictions like 2D-layout, energy consumption, chip area

⇒ Successful concepts from SANs not always applicable!

Simple fast switches better than complex sophisticated ones?

=> New original concepts required for the Many-Core Area!