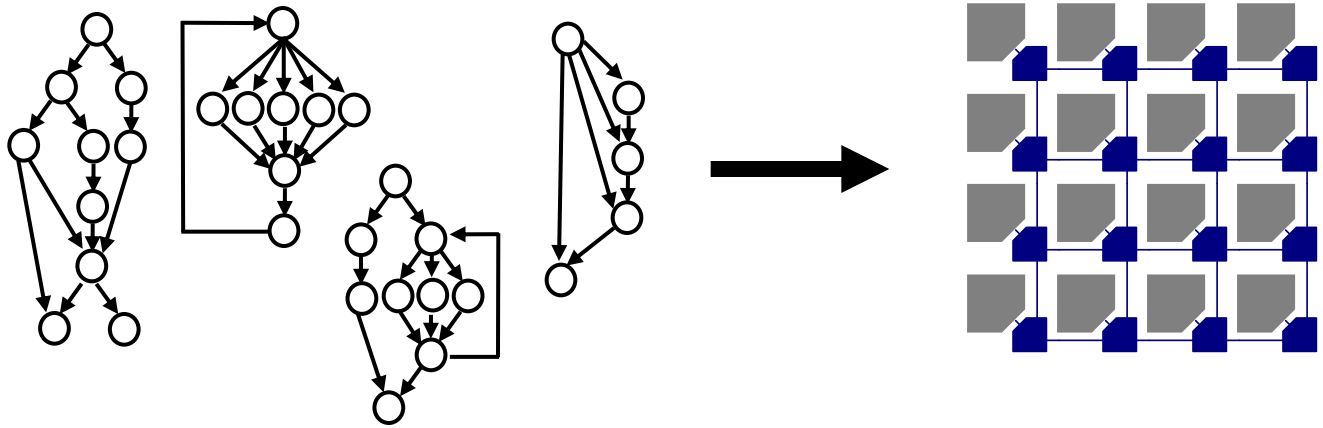
A Triple Hybrid Interconnect for Many-Cores: Reconfigurable Mesh, NoC and Barrier

Heiner Giefers, Marco Platzner
Computer Engineering Group
University of Paderborn

{hgiefers, platzner}@upb.de

Motivation

- Parallel applications on many-core architectures show different communication requirements



- Today's interconnect of choice is the packet switched NoCs
  - Scalable, allows for resource sharing
  - Broadcast and multicast are typically slow or expensive

- Can we improve performance by using a hybrid interconnect?
  - Select interconnect depending on communication pattern

Outline

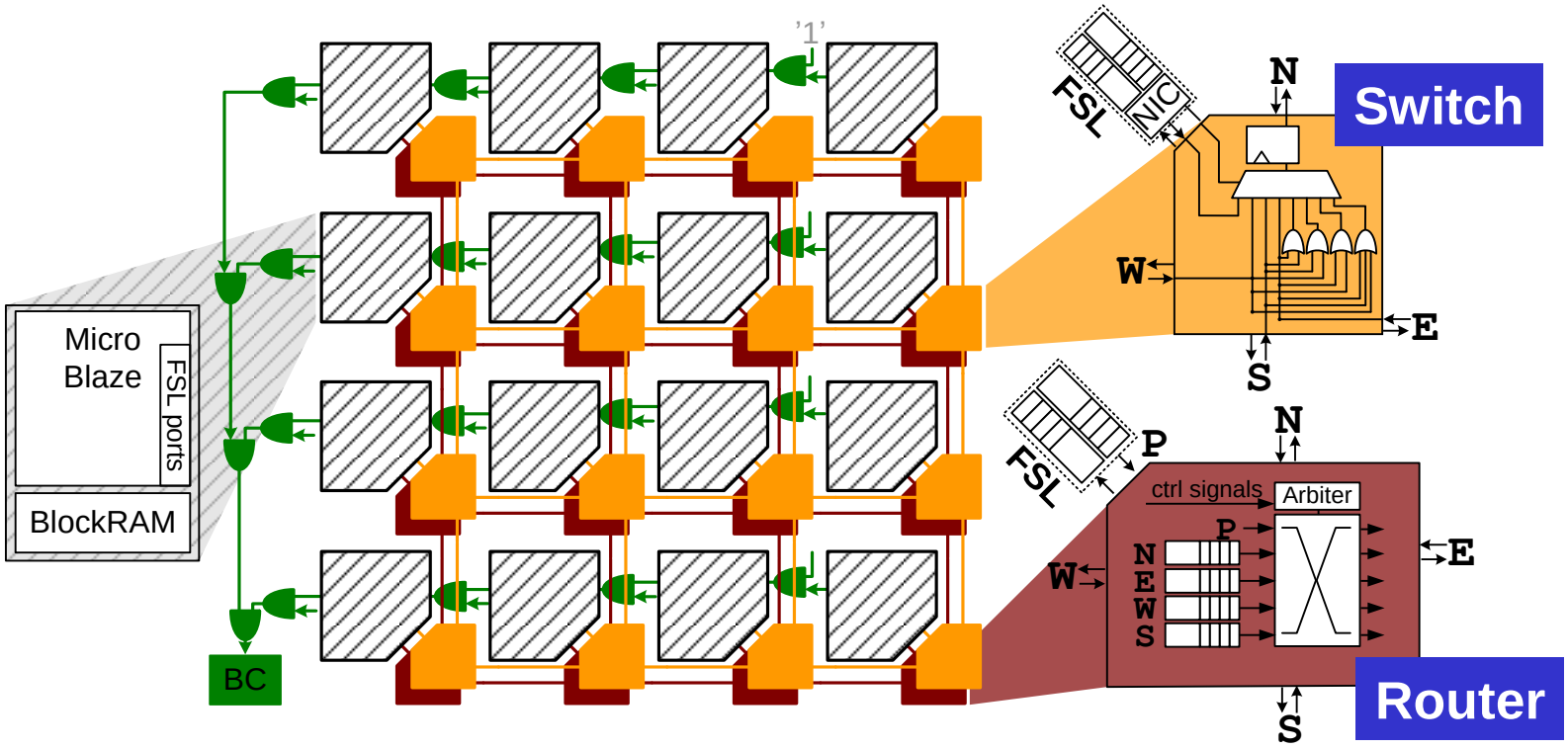
I. Many-core architecture

- Packet switched NoC
- Reconfigurable mesh network
- Barrier synchronization network
- Resource utilization results
- Communication API

II. Experimental results

- Operand communication
- Application case study

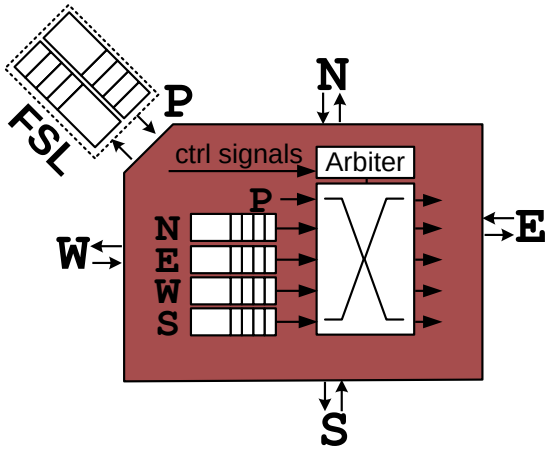
Triple Hybrid Interconnect for Many-Cores

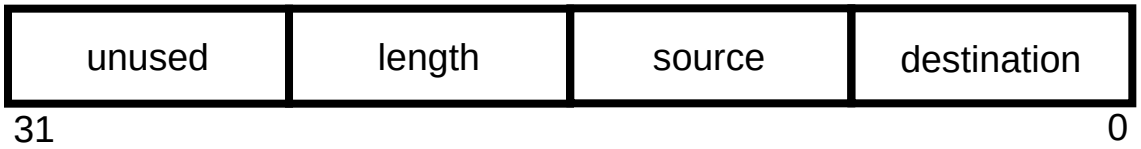


- Packet switched NoC
- Reconfigurable mesh network
- Barrier synchronization network

Packet Switched NoC

- Network interface: fast simplex link
- Buffers: 16 entry FIFOs
- Routing: 2D dimension order routing (XY routing)



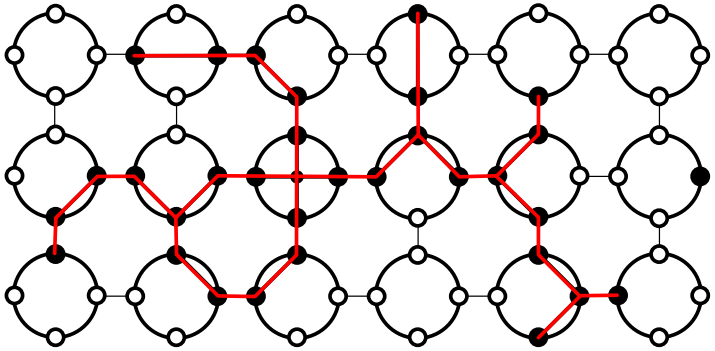


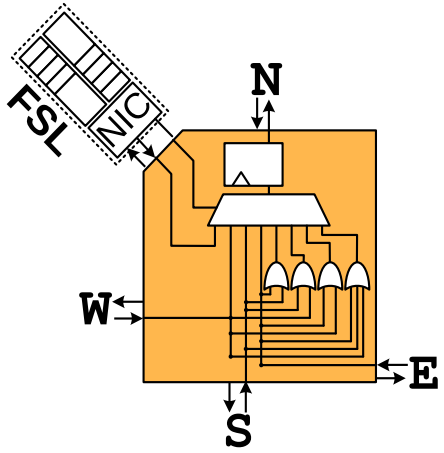
- Arbitration: static priorities, blocking
- Flow control: wormhole routing, handshake signals

→ 2 cycle flit latency per hop

Reconfigurable Mesh Network

- Based on the reconfigurable mesh model



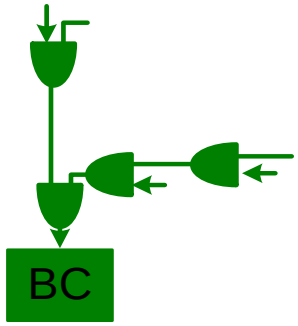


- Cores can autonomously reconfigure their local *switch pattern*
  - Build up global topologies
  - NIC interprets a control-labeled word as switch configuration
- Native support for broadcast and multicast
- Investigated reconfigurable mesh FPGA implementations earlier
  - Picoblaze processing elements, e.g., [FPL'07] and [FPL'09]
  - Microblaze processing elements [ERSA'10]

→ Single cycle latency per hop

Barrier Synchronization Network

- Beneficial for parallel algorithms that require global synchronization



- Bit-level AND-reduce tree
  - Implemented in FPGA's fast carry chains
  - Single-cycle evaluation

- Use the FSL capabilities to stall processors while waiting for a barrier
  - One *fslput* to signal a '1' onto the AND-reduce tree
  - A subsequent *fslget* stalls the Microblaze until all nodes have reached the barrier
  - Only use FSL ports, no FIFO

→ 3-cycle latency for global barrier

Resource Utilization Results (XC5VLX110T)

Module	LUT	FF	DSP	BRAM	f_{max}
6x5 tiles	66175 95%	35352 51%	60[1] 93%	120 81%	102 MHz
Router (8-bit)	385	286	0	0	247 MHz
Router (32-bit)	679	398	0	0	240 MHz
Switch (8-bit)	133	36	0	0	448 MHz
Switch (32-bit)	446	132	0	0	448 MHz
Barrier Core	6	3	0	0	519 MHz

[1]Only 64 DSPs available. One µB utilizes 3 DSPs for build-in HW-multiplier

- Tiled architecture
  - µB + 8-bit router + 32-bit switch + barrier core + 16kByte BRAM
  - Operate prototypes up to 30 tiles
- Switch is almost 2x as fast as a router

Communication API

NoC	Reconfigurable Mesh	Barrier Net	Description
	swconf(P)		Reconfigure local switch element
send(T,D)	*broadcast(D)*		Send single value
*sendMsg(T,*D,L)*	*broadcastMsg(*D,L)*		Send message of *L* values
receive()	*read()*		Receive a single value
*receiveMsg(*D,*S)*	*read(*D,L)*		Receive a message of *L* values
		sync	Set synchronization point

- Use standard microblaze-gcc toolchain
- *D*: 32-bit data (integer or float)
- *S/T*: 8-bit source/target address
- *P*: 4-bit reconfigurable mesh switch pattern
- *L*: 8-bit length of message in 32-bit words

Outline

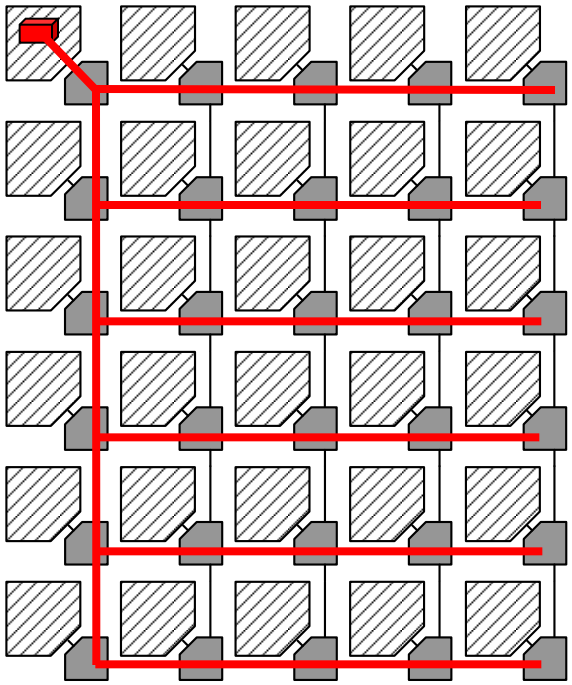
I. Many-core architecture

- Packet switched NoC
- Reconfigurable mesh network
- Barrier synchronization network
- Resource utilization results
- Communication API

II. Experimental results

- Operand communication
- Application case study

Operand Communication



Communication Pattern	Reconf. Mesh	NoC
1:n (broadcast)	74	1118 (15x)
1:1 (node 0 to node 29)	74	108
Multicast on first row	69	297
n:1 (nodes 1..29 to node 0)	733	402

measured cycles

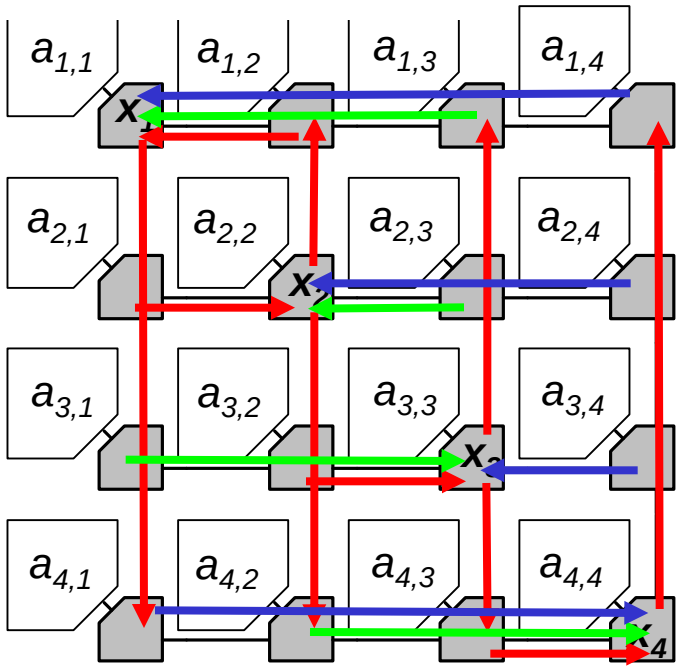
- NoC-broadcast or -multicast is handled by sending separate messages
- Overhead for setting up and consuming messages is relatively high
- NoC is more efficient for n:1 and n:m communication

Case Study: Jacobi Method

- Given a square system of linear equations $Ax = b$
- Iteratively solve $x_i^{(\mu+1)} = \frac{b_i}{a_{i,i}} - \sum_{\substack{k=1 \\ k \neq i}}^{n} \frac{a_{i,k}}{a_{i,i}} x_k^{\mu} \quad (i = 1, 2, \ldots, n)$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix}$$

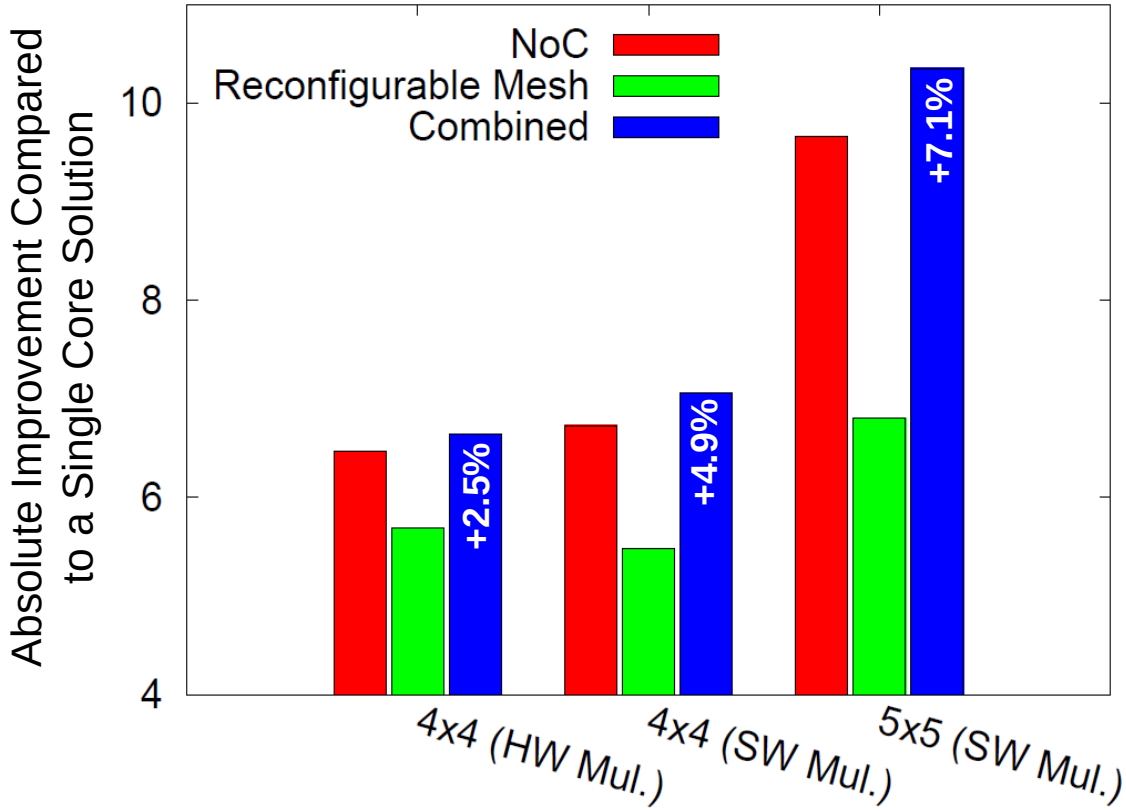
$$\begin{pmatrix} x_1^0 & & & \\ & x_2^0 & & \\ & & x_3^0 & \\ & & & x_4^0 \end{pmatrix} \begin{pmatrix} b_1 & & & \\ & b_2 & & \\ & & b_3 & \\ & & & b_4 \end{pmatrix}$$



1. Distribute x_i^{μ} on columns
2. All nodes compute products $a_{i,k}\, x_i^{\mu}$ in parallel
3. Send product to diagonal node of the same row
4. Diagonal nodes compute $x_i^{\mu+1}$

Case Study: Results

- Jacobi method trades on the NoC capabilities
- However, combined use of reconfigurable mesh, barrier and NoC brings additional performance



Summary and Outlook

- Approach: use different interconnects for different communication patterns in applications
- Introduced a triple hybrid interconnect
- Hybrid interconnect provides opportunities
  - NoC is advantageous for larger messages and dynamic workloads
  - Reconfigurable mesh features fast operand broad-/multicast
  - Barrier network brings fast global synchronization
- Jacobi method case study benefits from a combined use of all networks
- Outlook
  - Improve programming tool-flow: automate interconnect selection
  - Detailed analysis: energy consumption
  - Comparison: Hybrid interconnect vs. more complex NoC