A Statically Scheduled Time-Division-Multiplexed Network-on-Chip for Real-Time Systems

Martin Schoeberl, Florian Brandner,
Jens Sparsø, Evangelia Kasapaki

Technical University of Denamrk

Real-Time Systems

- Safety critical systems
  - E.g. avionic
- Results need to be delivered within a deadline
- Worst case execution time (WCET) needs to be statically analyzed
- Real-time systems go CMP
- How to provide timing guarantees?

Real-Time CMP

- NoC for real-time systems
  - Core to core communication
  - Core to shared memory communication
- Include NoC in WCET analysis
- Statically scheduled arbitration
- Time-division multiplexing

Outline

- What is T-CREST?
- A real-time network-on-chip
- Design of the S4NOC
- Bounds on minimal schedule periods
- Evaluation in an FPGA
- Discussion and conclusion

T-CREST

- EC funded FP7 STREP project
  - Time-predictable Multi-Core Architecture for Embedded Systems
- Construct time-predictable architectures:
  - Processor
  - Network-on-chip
  - Memory
  - Compiler
  - WCET analysis

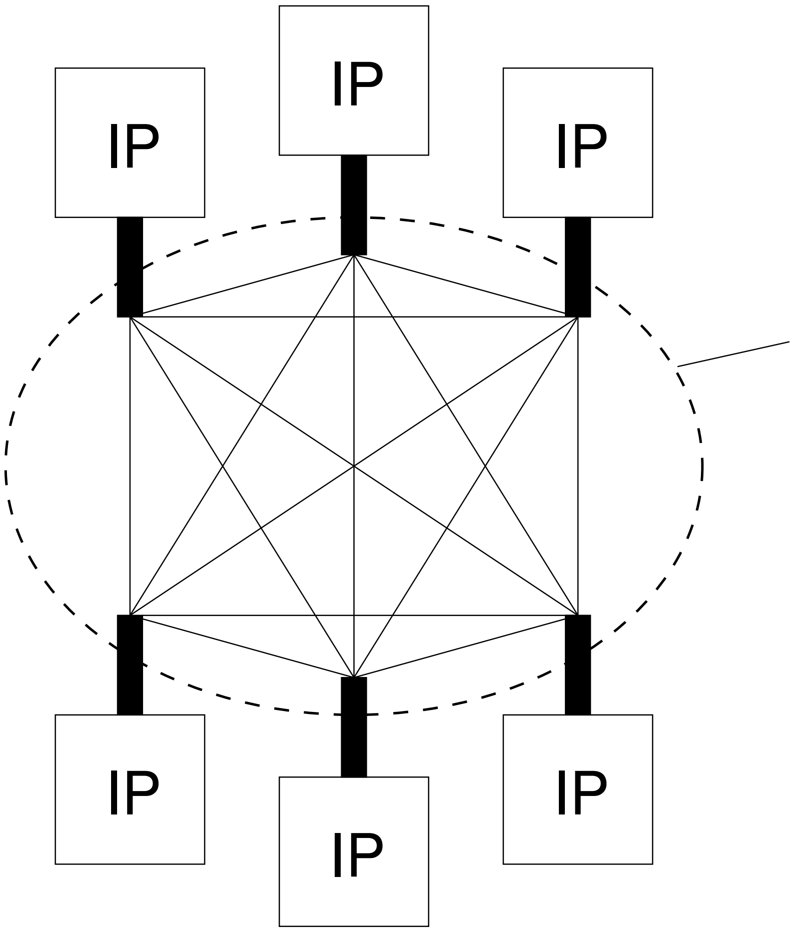
T-CREST

- 4 Universities, 4 industry partners
- 3 years runtime, started 9/2011
- Provide a complete platform
  - Hardware in an FPGA
  - Supporting compiler and analysis tool
- Resulting designs in open source – BSD
  - Cooperation welcome

NoC for Chip-Multiprocessing

- Homogenous CMP
- Regular network to connect cores
  - Mesh, bidirectional torus
- Serves two communication purposes
  - Message passing between cores
  - Access to shared memory
- This talk is about the message passing NoC

NoC



Network–on–chip
- TDM–based
- Virtual circuits; all–to–all
- Topologies: 2D–mesh, torous, tree

S4NoC and T-CREST

- S4NOC is a first step to explore ideas
- Real T-CREST NoC will be
  - Asynchronous
  - Configurable TDM schedule
  - Might contain 2 (or more) NoCs
  - Fancier network adapter
  - …we will see during the next 2 years…
- Communication and memory hierarchy is where the action is in a CMP

Real-Time Guarantees

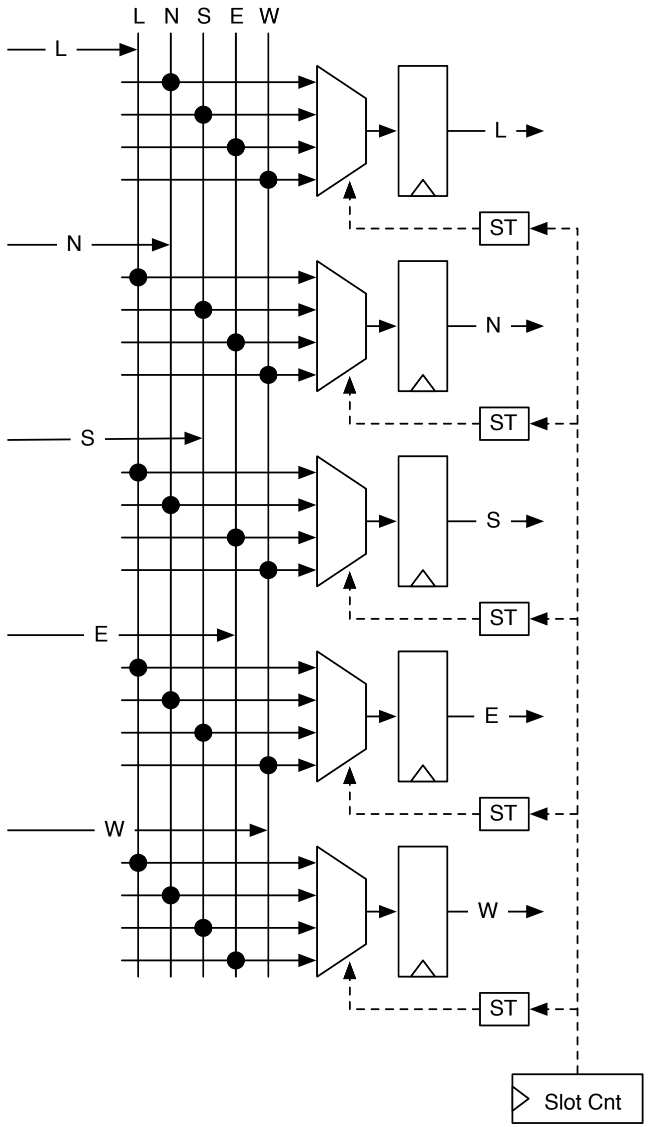
- NoC is a shared communication medium
- Needs arbitration
  - Time-division-multiplexing is predictable
- Message latency/bandwidth depends on
  - Schedule
  - Topology
  - Number of nodes

First Design Decisions

- All to all communication
- Single word messages
- Routing information in the
  - Router
  - Network adapter
- Single cycle per hop
  - No buffering in the router
- No flow-control at NoC level
  - Done at higher level

The Router

- Just multiplexer and register
- Static schedule
  - Conflict free
  - No way to buffer
  - No flow control
- Low resource consumption



TDM Schedule

- Static schedule
  - Generated off-line
  - ‘Before chip production’
- All to all communication
- Has a period
- Single word scheduling simplifies schedule generation
  - No ‘pipeline’ effects to consider

Period Bounds

- A TDM round includes all communication needs
- That round is the TDM period
- Period determines maximum latency
- Minimize schedule period
  - We found optimal solutions
    - Up to 5x5
  - Heuristics for larger NoCs
    - Nice solution for regular structures

Period Bounds

- IO Bound (n-1)
- Capacity bound (# links)
- Bisection bound (half to half comm.)

Size	**Mesh**	**Torus**	**Bi-torus**
3x3	8	9	8
4x4	16	24	15
5x5	32	50	24
6x6		90	35
7x7			48
8x8			64
9x9			92

Router Implementation

- Build a many core NoC in a medium sized FPGA
  - Router is small
  - Use a tiny processor – Leros
- Router is simple
  - Double clock the NoC
- First experiment without a real application

Size and Frequency

- Leros processor
  - ~220 LCs, ~125 MHz
- Router/NoC
  - 50-160 LCs, 230—330 MHz
- 9x9 fitted into the Altera DE2-70!
- However, no real network adapter

- A simple RISC pipeline ca. 2000 LCs

A Simple Network Adapter

- Router/NoC is minimal
  - What is a minimal NA?
- Single rx and tx register
  - But one pair for each channel
- Rx register full flag, tx register empty flag
  - Like a serial port on a PC

NA First Numbers

- 4x4 bi-torus system
- Network adapter:
  - 1 on-chip memory block
  - ~ 230 LCs (18 for schedule table)
- Router
  - 98 LCs (19 for schedule table)
- Fmax: 90 MHz Leros, 170 MHz NoC

Schedule Tables

- Fixed schedules
  - Generated VHDL code
  - Implemented in LUTs

Cores	NA Table	Router Table	Schedule Length
16	18 LCs	19 LCs	20
25	26 LCs	22 LCs	28
36	52 LCs	37 LCs	43
49	73 LCs	50 LCs	59

Discussion

- TDM wastes bandwidth
- All to all schedule wastes even more!
  - Does it matter?
- There is plenty of bandwidth on-chip
  - Wires are cheap
  - 1024 wide busses in an FPGA possible
- Bandwidth relative to cost matters

Discussion

- Fixed/static schedules are cheap
  - The table is just ‘ROM’
  - No hardware needed to the load schedule
  - Instant on – no HW needed to support bootstraping of the system
- Not enough bandwidth?
  - Wider links
  - Additional NoCs
  - Cluster your cores

Summary

- Many-core CMP systems need a NoC
- For RTS we need time-predictable communication
  - TDM based arbitration
- First experiments with static TDM NoCs
  - Cheap HW
- TDM router is simple – NA is where the action is