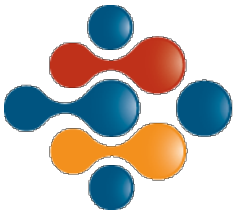


T-CREST

Time-predictable Multi-Core Architecture for Embedded Systems

Martin Schoeberl

Technical University of Denmark

Real-Time Systems

- Systems with timing constraints
  - In (hard) real-time systems
    - Function has to be correct
    - Function has to deliver result in time
- Timing proof with schedulability analysis
  - Execution time of tasks need to be known
  - WCET analysis gives the input

Worst-Case Execution Time

- Measurement of execution time is not safe
  - Execution time is data dependent
  - Did we trigger the worst-case?
- Static WCET Analysis
  - High-level WCET analysis is mature research
  - Considers control flow and flow facts (loop bounds)

Static WCET Analysis Issue

- Low-level analysis is the main issue
  - Modern processors are too complex
  - Lot of (hidden) state information
    - Key for performance
    - Issue for WCET analysis
- WCET analysis is about 10 years behind current processors
- Multiprocessors currently not analyzable

New Architectures Needed

- Design a computer architecture for real-time systems
  - WCET is the main design constraint
  - Average-case performance not (so) interesting
- Use and develop features that are
  - WCET analysis driven
  - Have a low WCET

Time-predictable Computer Architecture

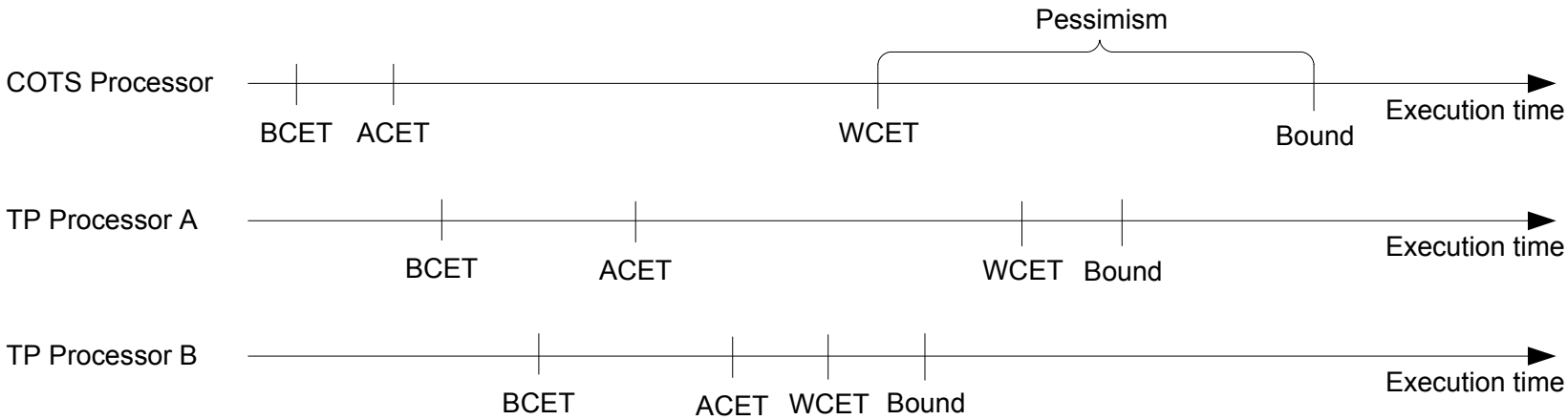
- Common computer architecture wisdom

  Make the common case fast and
  the uncommon case just correct

- Time-predictable computer architecture

  Make the worst case fast and
  the whole system analyzable

Our WCET Target Architecture

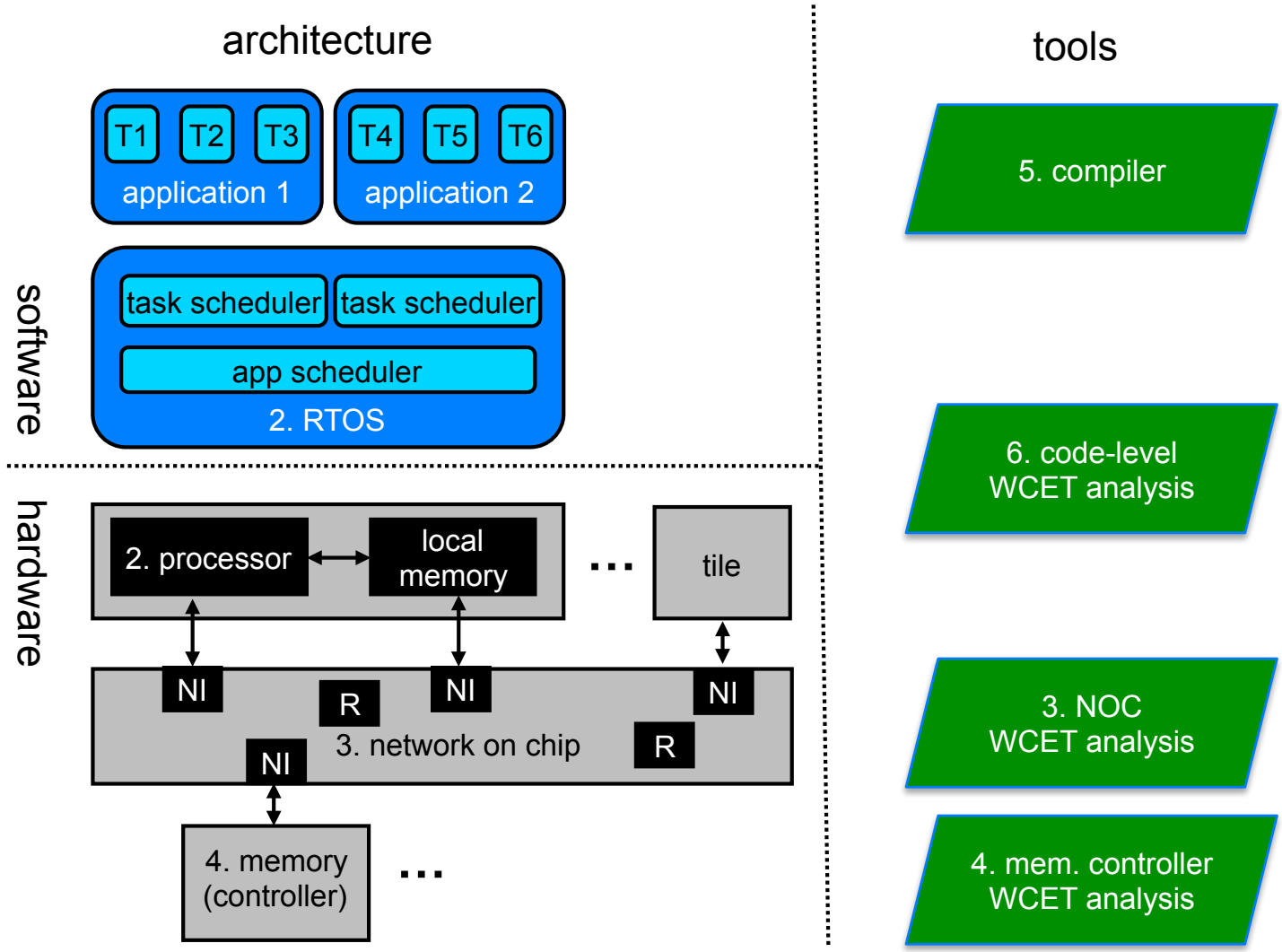


- Trying to catch up with the analysis on the complexity of average case optimized architectures is not an option
- We need a sea change and take a constructive approach. Design processors, memory, and interconnect for real-time systems!

T-CREST Architecture

- Chip-multiprocessor for high performance
  - Target: 64 core in an FPGA
- Time-predictable
  - Processor
  - Network-on-Chip (NoC)
  - Local memory (SPM, $)
  - SDRAM controller
- Integration in WCET analysis

T-CREST Architecture



T-CREST Outcome

- Provide a complete platform
  - Hardware in an FPGA
  - Supporting compiler and analysis tool
- Resulting designs in open source
  - BSD license
  - Cooperation welcome
- Up to compiler
  - No operating system research
  - No MoC research
  - No automatic parallelization research

T-CREST Funding

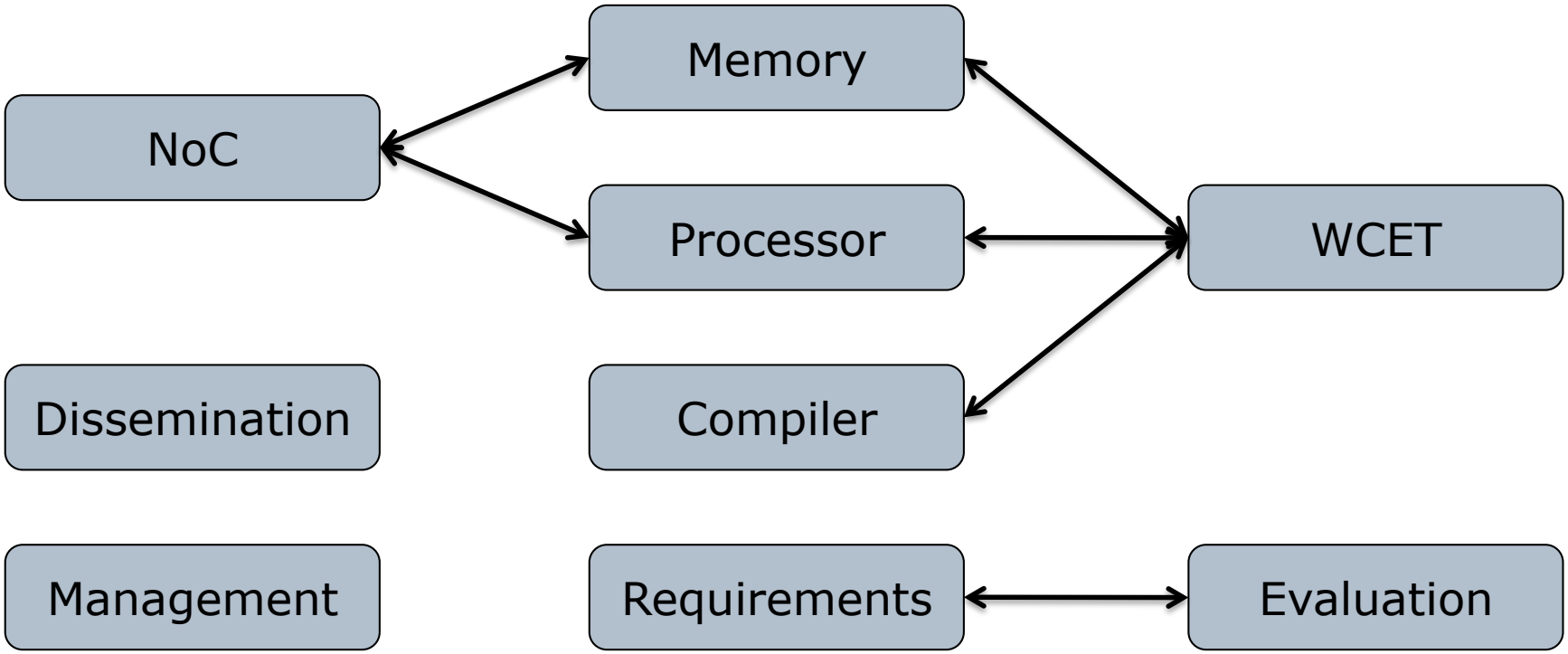
- 3 Year STREP project
  - Started 09/2011
- Funded by the European Commission
- Total EC contribution 2.65 M EUR
- 4 Universities, 4 industry partners
  - Open Group project coordinator
  - DTU technical lead

T-CREST Partners

Work Packages

- WP 1 Requirements Analysis (GMV)
- WP 2 Processor (DTU)
- WP 3 Network on Chip (DTU)
- WP 4 Memory Hierarchy (UoY)
- WP 5 Compiler (TUV)
- WP 6 Code-Level WCET Analysis (AbsInt)
- WP 7 Integration and Evaluation (GMV)
- WP 8 Dissemination (TOG)
- WP 9 Management (TOG)

WP Interaction



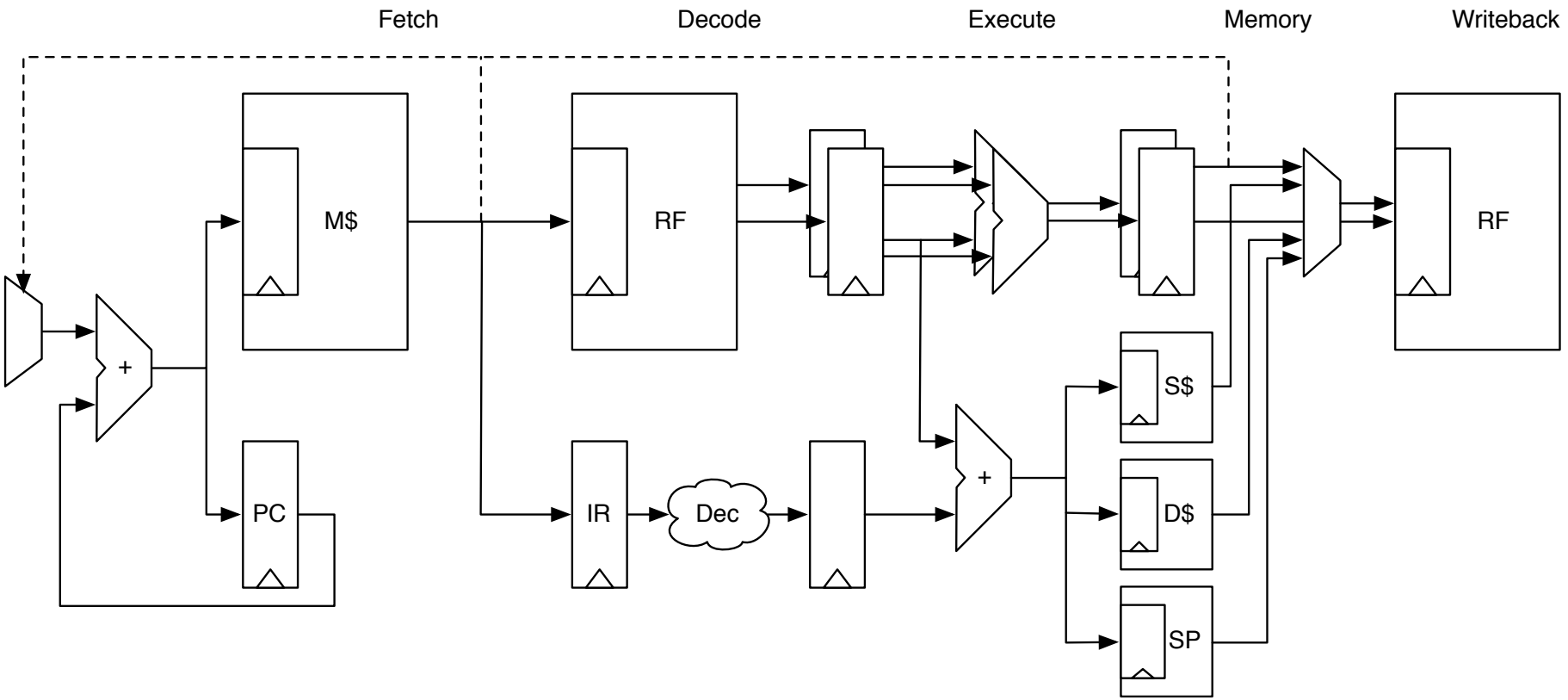
WP2: Processor

- Time-predictable processor
- Called Patmos
- Flexibility to define the instruction set
  - A compiler is adapted for Patmos
- Co-design for low WCET of
  - Patmos
  - Compiler
  - WCET analysis

Patmos

- RISC style microprocessor
- Dual issue
- Full predication – all instructions
- Split caches
- Split load
- Intended as research platform for real-time architecture (e.g. caches, SPM)

Pipeline Overview



Instruction Set

- 3 register ALU instructions
- Immediate ALU instructions
  - Short and long ALU (2. issue slot)
- Load/store with register + offset
  - Word, half word, byte
  - Aligned access
  - Typed ld/st for different data caches
  - Split load

Instruction Set cont.

- Branch with offset and register indirect
- Call instruction to support method cache
- Compare and predicate operation
- Stack cache support instructions
- Dual issue
  - ALU in both pipelines
  - Load/store and branch in 1. pipeline

Simulator

- Faithful emulation of Patmos
  - Reasonable simulation speed
  - Easily adaptable and extensible
  - Detailed feedback and runtime statistics
- Intended as design aid
  - Compiler development
  - Runtime statistics
  - Support execution tracing

Simulator Capabilities

- Pipeline model
  - by-passing, stalling, delay slots
- Stack cache
- Method cache
  - FIFO and LRU replacement
- Data cache
  - Set associative, LRU, write-through, no-allocation
- Main memory
  - Timing of block transfers
- IO Devices: UART, cycle counter

Compiler

- Adaption of LLVM
- Basic code generation works
  - Only single pipeline
- Support for stack cache instructions
  - Reserve, free, ensure
- Method cache support
- Shall optimize for WCET
  - Integration with AbsInt tool aiT

Technology Observation

- Prototype in FPGA
  - Logic cells, registers, on-chip memories
- About 200-400 LCs per on-chip memory block
  - Processor 3000 LCs => 10 memory blocks
  - Memory is the bottleneck
- On-chip memories are synchronous
  - Registers at the inputs
  - Influences the pipeline structure

Patmos Status

- Simulator complete
  - Complete ISA coverage
  - Executes MiBench benchmark suite
- Hardware described in VHDL
  - FPGA implementation
  - On-chip instruction ROM and data memory
  - Test programs in assembler
  - Tiny C programs are ok

Co-simulation

- Compare high-level simulation with VHDL description
  - Pasim and VHDL in ModelSim
- Compare most important processor state every cycle: register file
  - RF content dumped into files
  - Small tool compares
- Collection of assembler test cases
- Nightly run on test server + email

Patmos Next Steps

- Executing MiBench in the HW
- Stack cache implementation and paper
- Dual pipeline
- More on caches, SPMs,…
- Attach the Network-on-Chip
- Integration with memory controller
  - Is there a memory NoC?

WP3: Network-on-Chip

T-CREST

- A basic processor pipeline is predictable
  - Not too many research challenges
- Communication and memory hierarchy is where the action is in a CMP
- Multicore needs a time-predictable on-chip communication

Real-Time CMP

- NoC for real-time systems
- Include NoC latency in WCET analysis
- Statically scheduled arbitration
- Time-division multiplexing

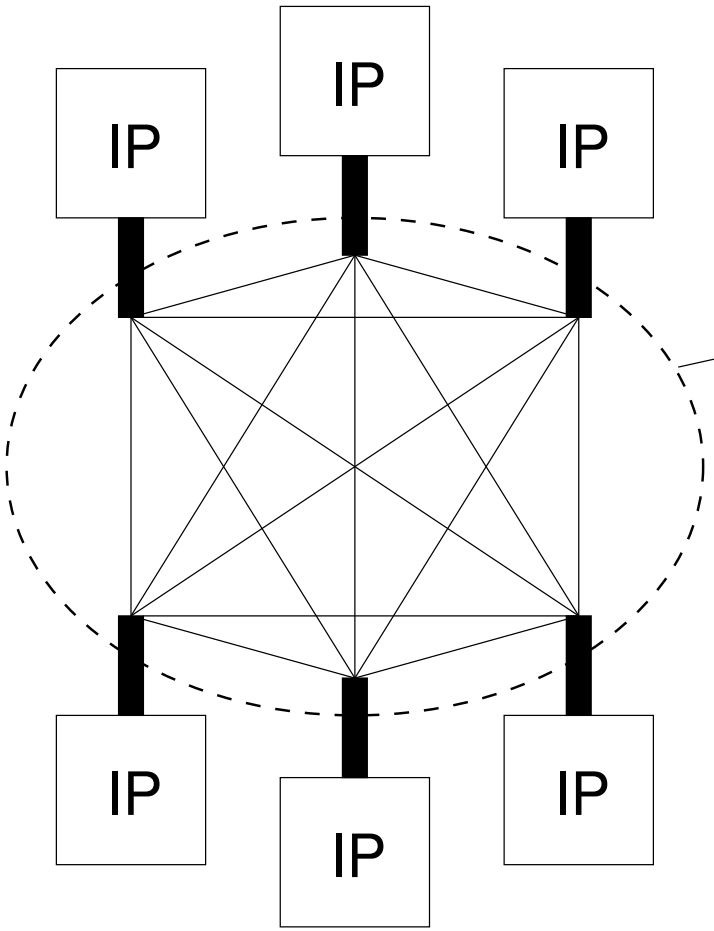
  - NOCS 2012: "A Statically Scheduled Time-Division-Multiplexed Network-on-Chip for Real-Time Systems"
  - + RTNS, + NORCHIP, + DATE

NoC for Chip-Multiprocessing

T-CREST

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

NoC Communication Cloud



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…

Real-Time Guarantees

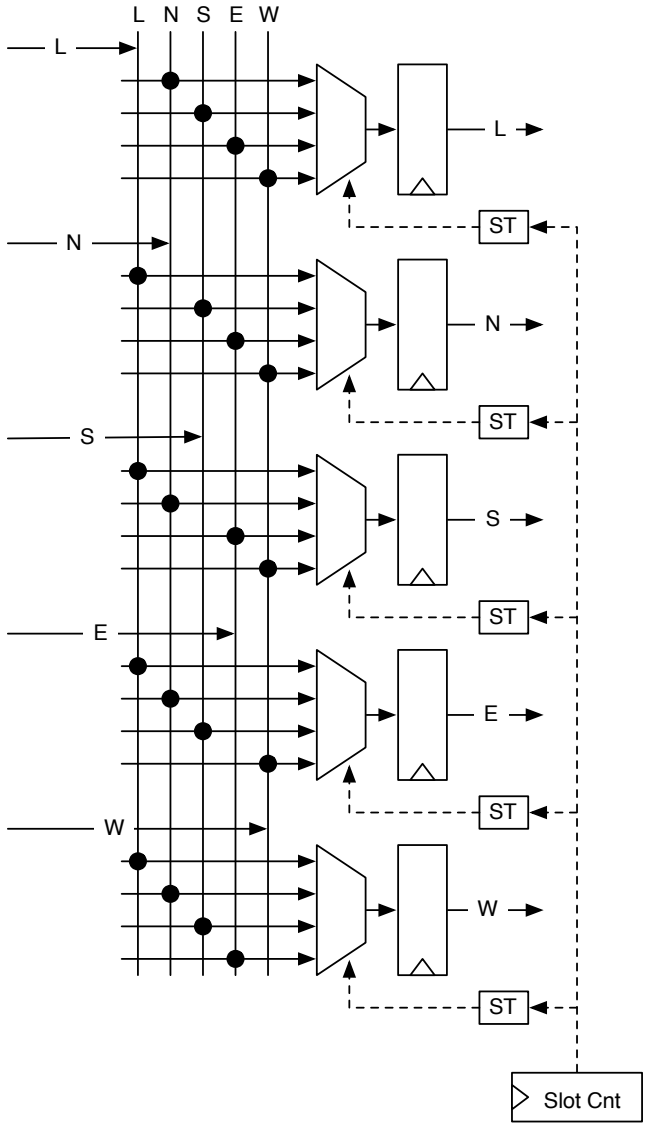
- NoC is a shared communication medium
- Needs arbitration
  - Time-division-multiplexing is predictable
- Message latency/bandwidth depends on
  - Schedule
  - Topology
  - Number of nodes

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 registers
  - But one pair for each channel
- Rx/tx register full/empty flags
  - 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
- TDM all-to-all routers are cheap
- Bandwidth relative to cost matters

NoC Summary

- Static TDM schedule is
  - Time-predictable
  - Simple (low HW resources)
- Network adapter in development
  - TDM from end to end
    - Memory – NoC – Memory
  - No flow control needed
  - Also simple
- Usability needs to be explored

T-CREST: More Info

- T-CREST web site
  - http://www.t-crest.org/
  - Most deliverables are public
  - Some first papers
- Development
  - Most artifacts are open-source
  - Hosted at GitHub
  - https://github.com/t-crest

Cooperation: PRET

- PRET – PTARM
  - Similar goal as Patmos
- Look at memory hierarchy
  - SPM – caches is a spectrum
    - SPM with address translation
    - SPM with blocks
    - Method cache
- Multicore
  - A PTARM CMP with the T-CREST NoC

Cooperation: Ptolemy

- T-CREST has no MoC and no OS
  - But we need one!
- Can we learn from Ptolemy?
  - Look into Ptolemy examples
- Actors fit nicely to many-cores
  - How can we map communication to NoC message passing?
  - Can we generate code for many-cores?

Cooperation: Time-predictable

- What does time-predictable mean?
  - There are some definitions out there
    - Don’t like them
  - Can it be quantified?
- What is worst-case performance?
- Difference between
  - Repeatable timing
  - Time-predictable

Summary

- New computer architecture for real-time systems
- WCET analyzability is of primary importance
- T-CREST is a platform
  - Processor, NoC, memory controller, compiler, WCET analysis
- Technology will be mostly open source