



T-CREST

Time-predictable Processor and Network-on-Chip

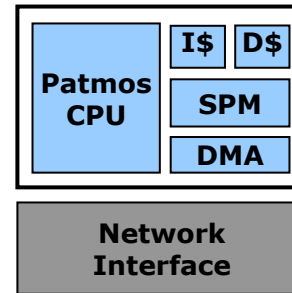
Jens Sparsø

Technical University of Denmark

T-CREST platform

■ Processor node

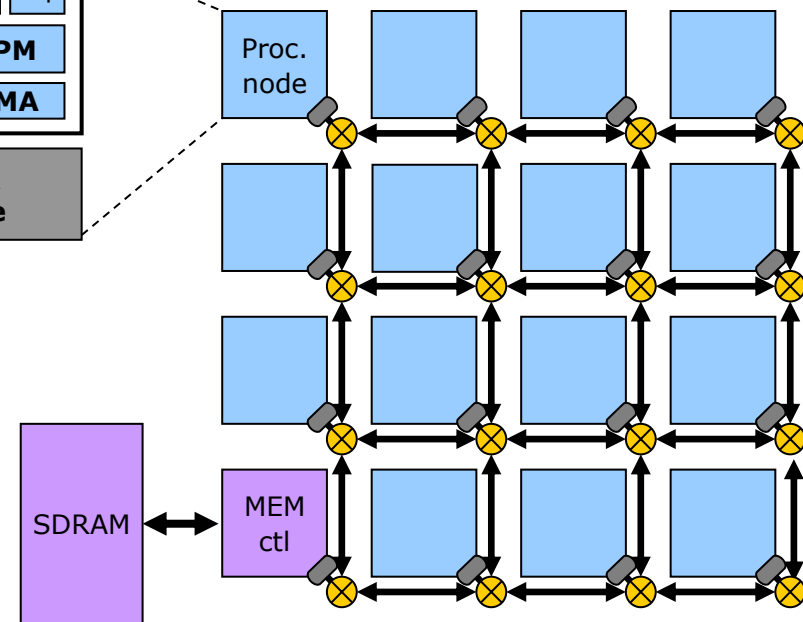
- ◆ Patmos CPU
- ◆ Local memories (caches, SPM)
- ◆ DMA controllers



■ SDRAM Memory controller node

■ Connected by NOC

■ Globally Asynchronous Locally Synchronous implementation



Research challenges

- A basic MIPS-style processor pipeline is predictable
 - ◆ Not too many research challenges
- *Communication and memory hierarchy* is where the action is in a CMP.

Research Challenges cont.

- Inside a processor node:
 - ◆ Scratch pad memories
 - ◆ Special caches (stack / method / ...)Supported by special instructions
- Processor to processor Communication NOC
 - ◆ DMA driven block transfers (message passing)
 - ◆ Nature: All-to-all
- Processor to shared memory (SDRAM)
 - ◆ Arbitration in NOC *and* in memory controller
 - ◆ Nature: All-towards-oneMemory NOC

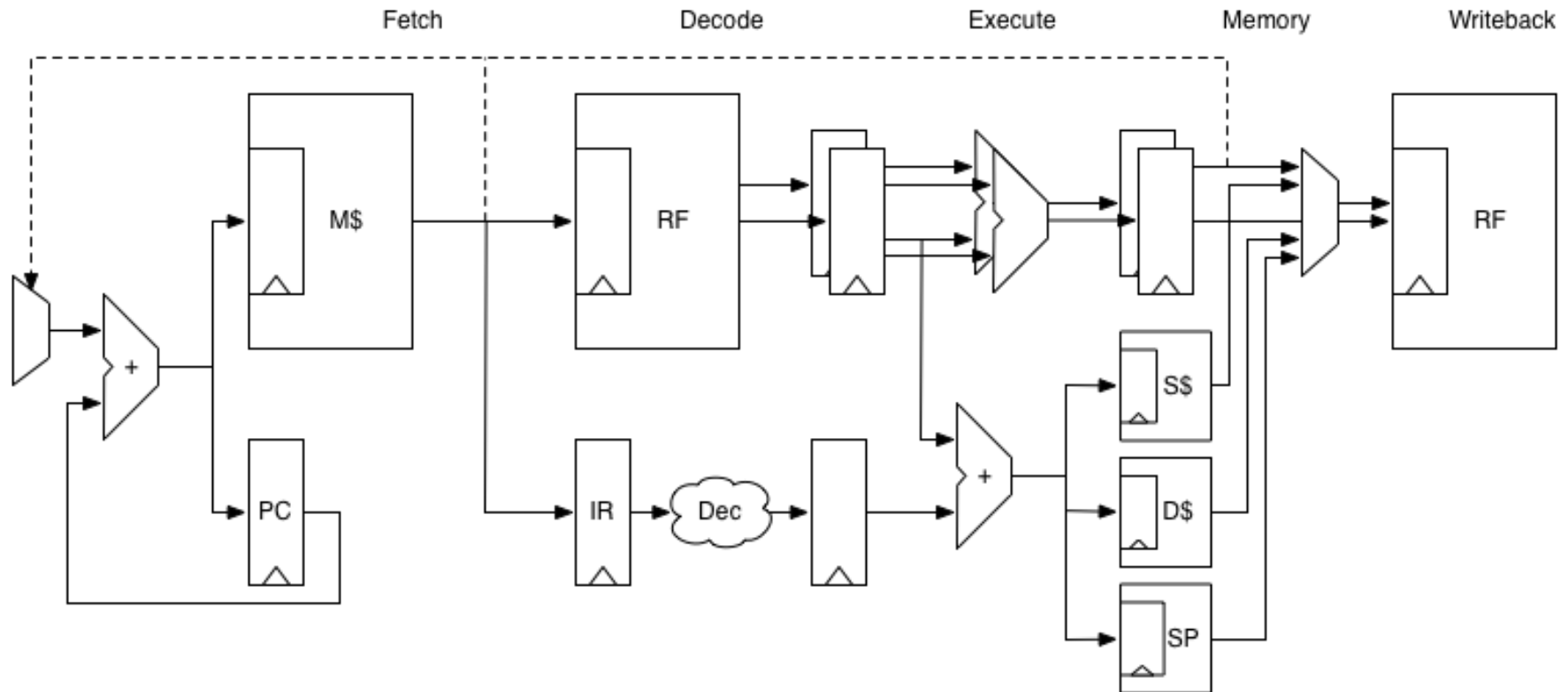
WP2: Processor

- 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

- Intended as research platform for real-time architecture
- Dual issue RISC style processor
- Full predication – all instructions
- Local caches and memories:
 - ◆ Scratch pad memory (SPM)
 - ◆ Stack cache
 - ◆ Method cache (FIFO and LRU replacement)
 - ◆ Data cache (Set associative, LRU, write-through, no-allocation)

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 load/store 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

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

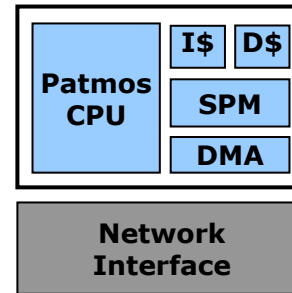
Patmos Next Steps

- Executing MiBench in the HW
- Stack cache implementation
- Dual pipeline
- More on caches, SPMs,...
- Attach the Network-on-Chip
- Integration with memory controller

T-CREST platform

■ Processor node

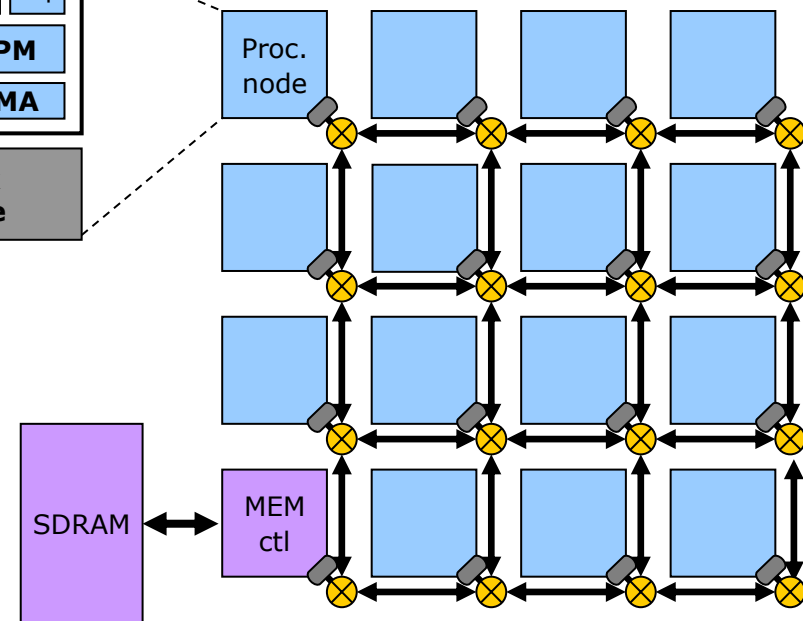
- ◆ Patmos CPU
- ◆ Local memories (caches, SPM)
- ◆ DMA controllers



■ SDRAM Memory controller node

■ Connected by NOC

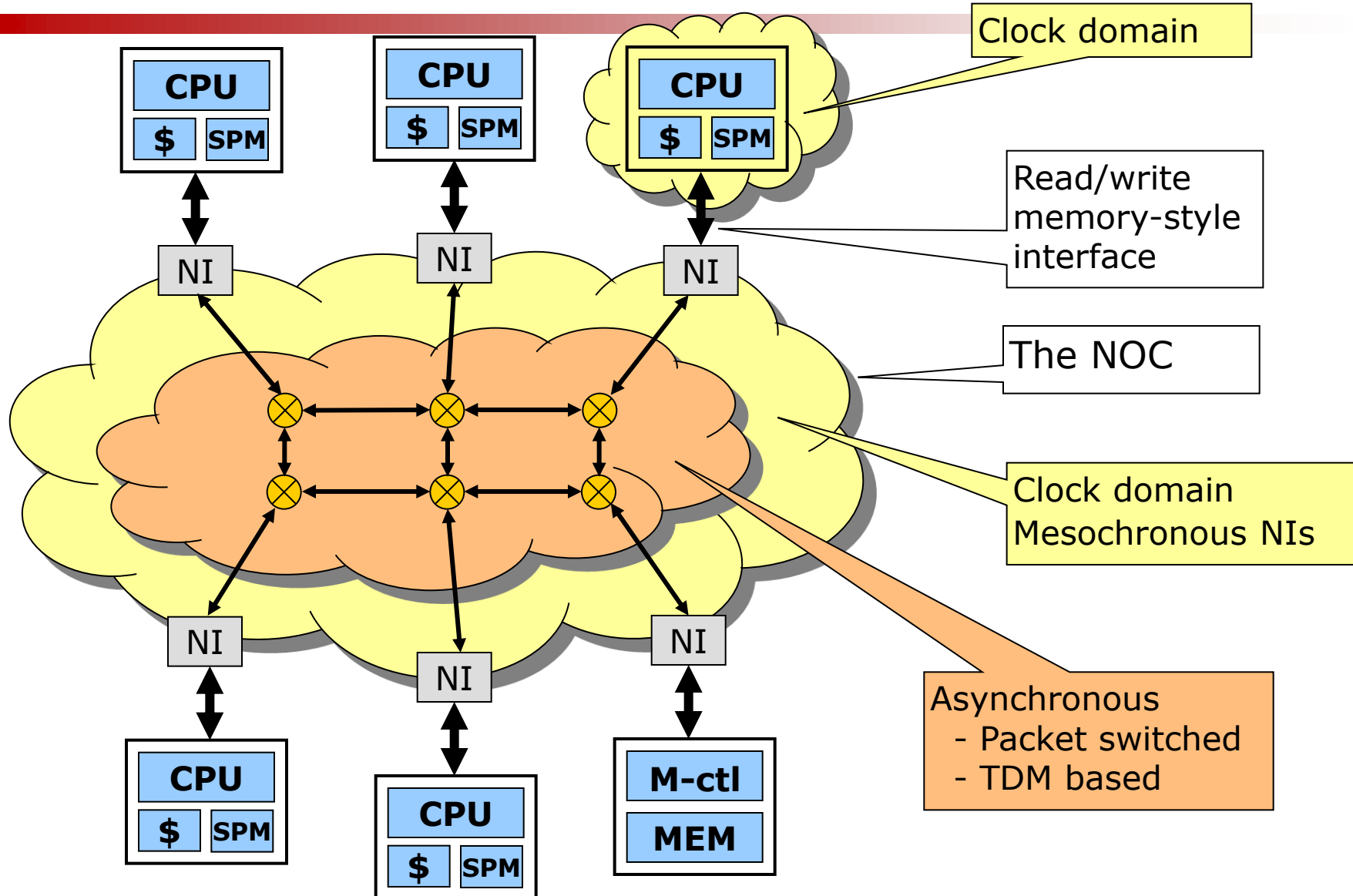
■ Globally Asynchronous Locally Synchronous implementation



WP3: Network on Chip

- Two types of traffic:
 - ◆ Processor to processor
 - DMA driven block transfers SPM → SPM (message passing)
 - Nature: All-to-all
 - ◆ Processor to shared memory (SDRAM)
 - Arbitration in NOC *and* in memory controller
 - Nature: All-towards-one
- Globally-Asynchronous
Locally-Synchronous (GALS)
implementation of platform

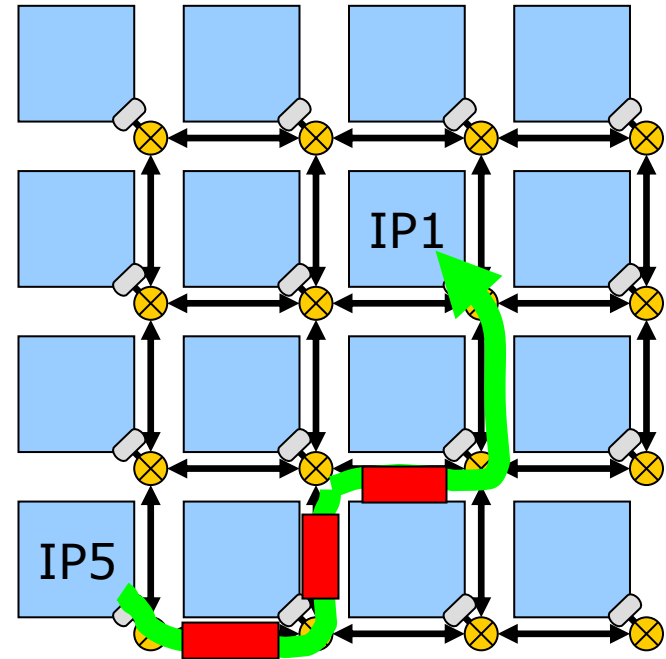
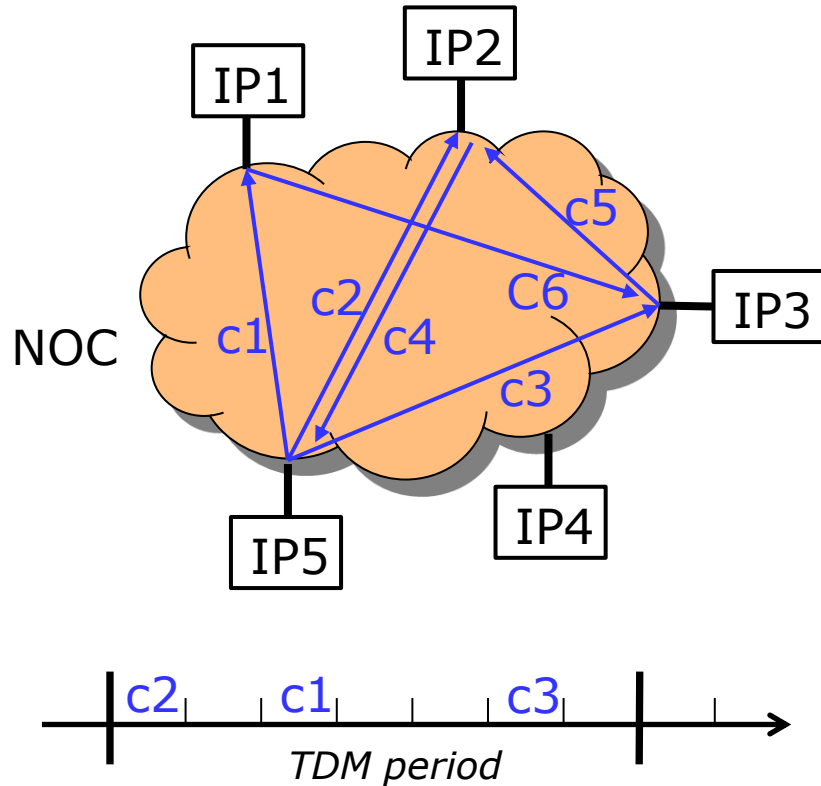
NOC centric view



Time predictable NOC

- Include NoC latency in WCET analysis
 - ◆ Note: most NOC's are not time predictable
- Time predictable:
 - ◆ Problem: Avoid interference of traffic
 - ◆ Solution: End-to-end (virtual) circuits
- T-CREST solution:
 - ◆ TDM (Time Division Multiplexing) with static schedule
 - ◆ HW implementation is simple and fast
 - ◆ WCET analysis is simple

Principles: TDM-based NOC



- NOC is a simple X-Y switched pipelined structure.
- No buffering. No arbitration.

Study of TDM scheduling

- Static schedule
- All-to-all
 - ◆ $N \times (N-1)$ virtual circuits
- Same bandwidth on all circuits
- Topologies:
 - ◆ Mesh
 - ◆ Torous
 - ◆ Bi-torous
 - ◆ Ring

TDM schedule period

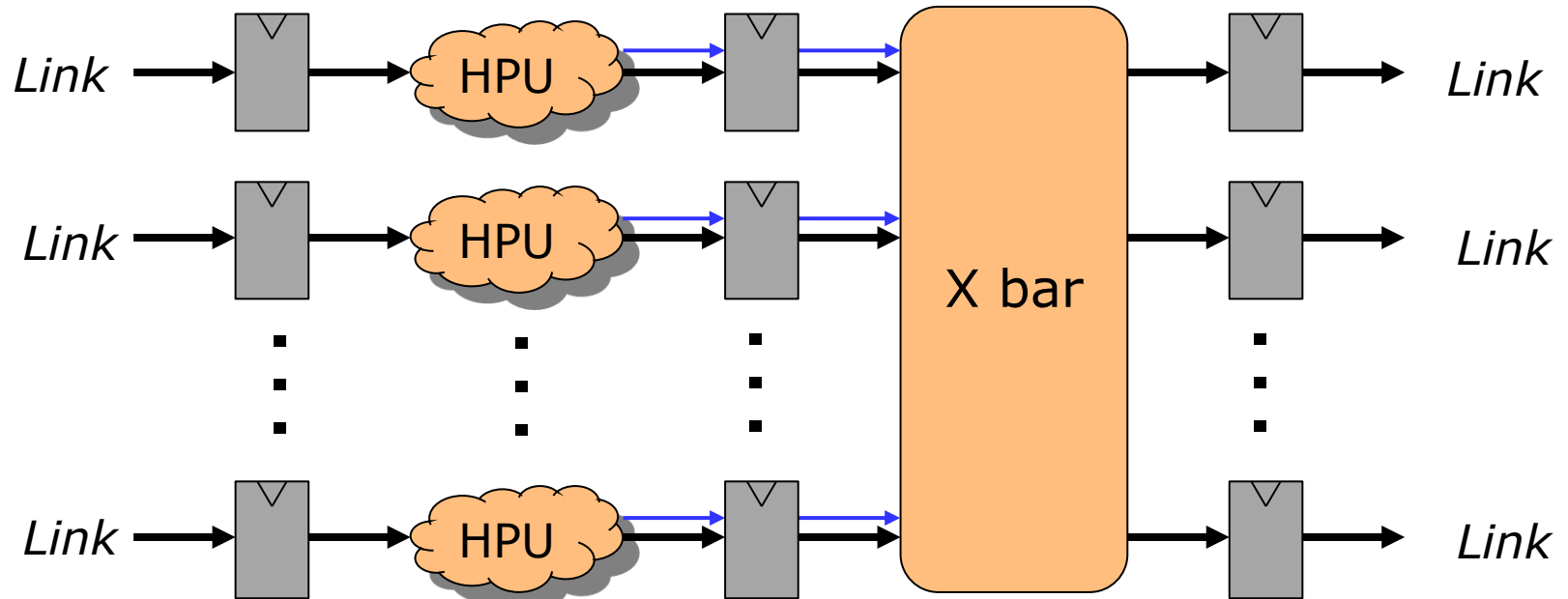
■ Bounds

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

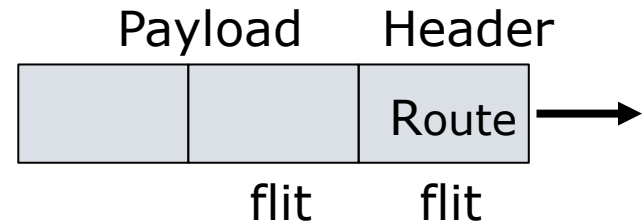
■ Minimum (or best found)

Size	Mesh		Torus		Bi-torus	
	bound	min	bound	min	bound	min
3x3	8	10	9	11	8	10(10)
4x4	16	18	24	26	15	18(19)
5x5	32	34	50	52	24	28(30)
6x6	54		90		35	(45)
7x7					48	(63)
8x8					64	(86)
9x9					92	(113)

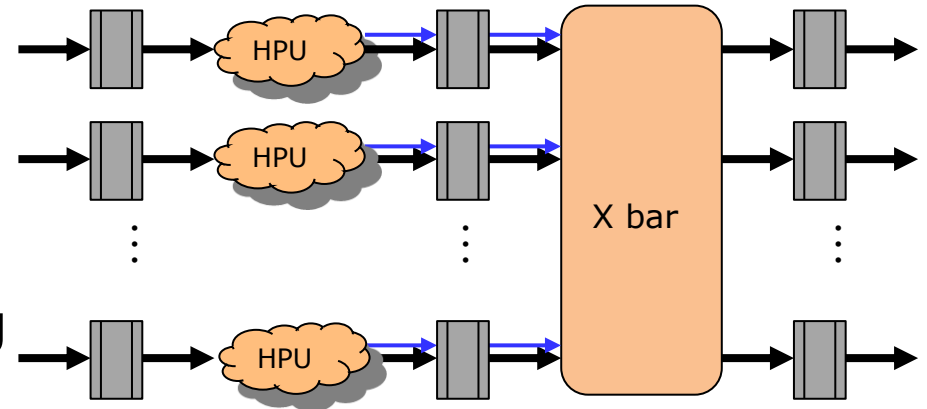
A TDM-based router



A. Hansson, M. Subburaman, K. Goossens,
 "Aelite: A Flit-Synchronous Network on Chip with
 Composable and Predictable Services," Proc. DATE 2009



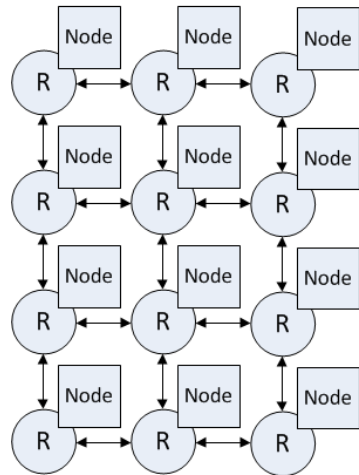
Asynchronous TDM router



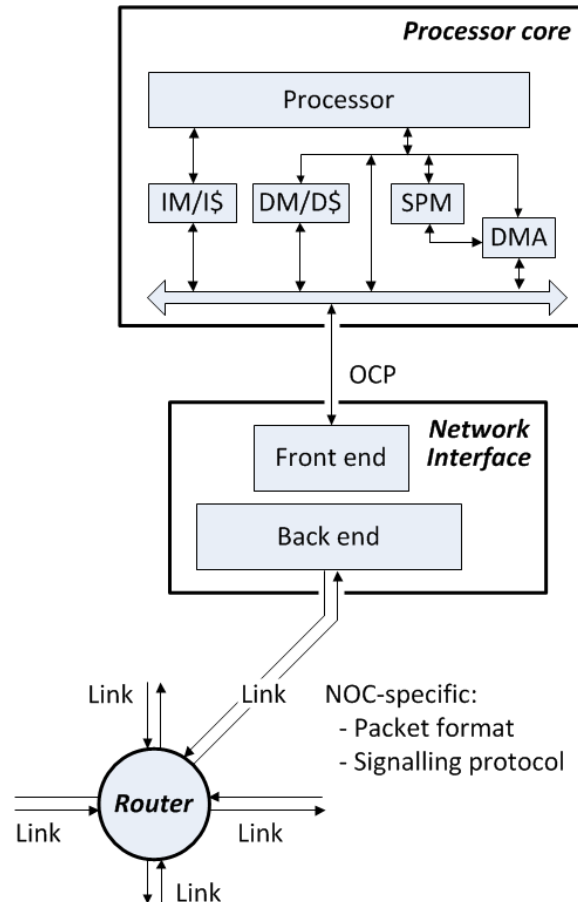
- Design is more challenging than it looks:
- Which asynchronous design style?
 - ◆ Delay insensitive / bundled data?
 - ◆ 2-phase or 4-phase signaling (events or signal levels)
- TDM requires some form of common time (ticking) but most ticks carry no data.
 - ◆ How to implement something that resembles clock gating?
 - ◆ (avoid excessive power consumption)

NI for T-CREST NOC

Original T-CREST plan (Traditional NI design)

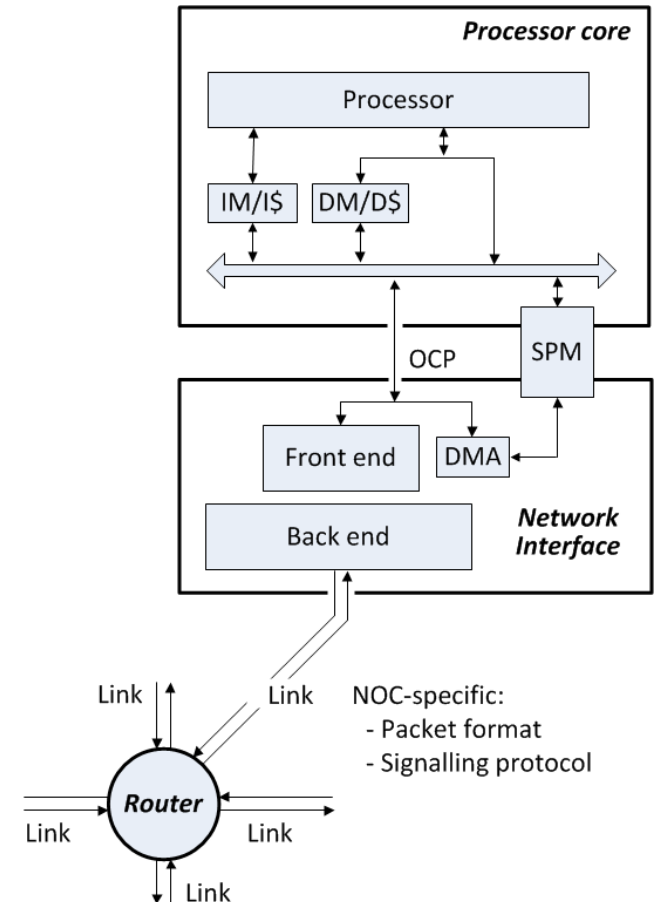


(a)



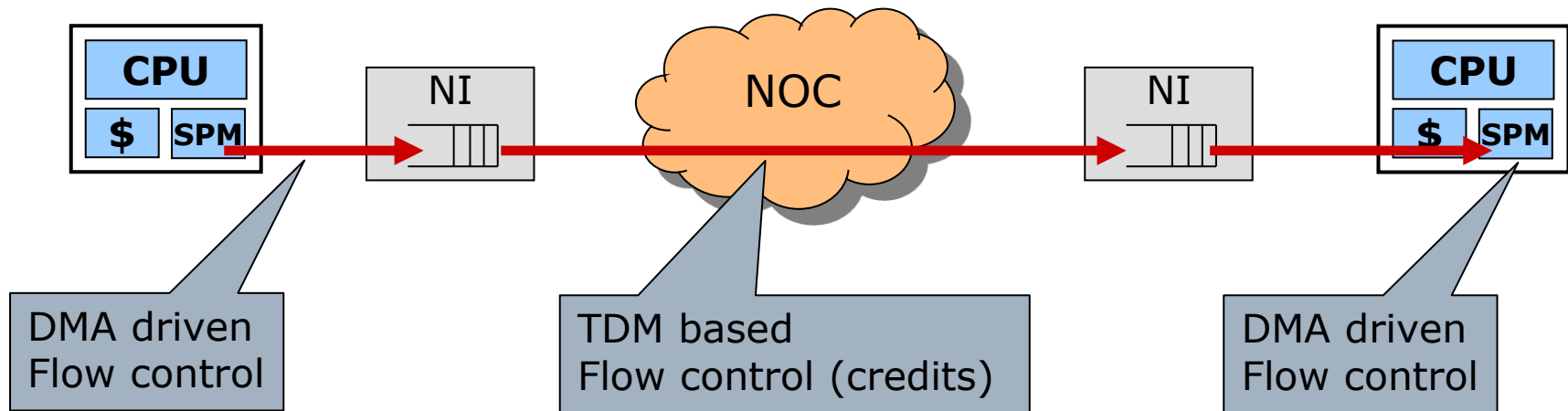
(b)

New NI micro-architecture



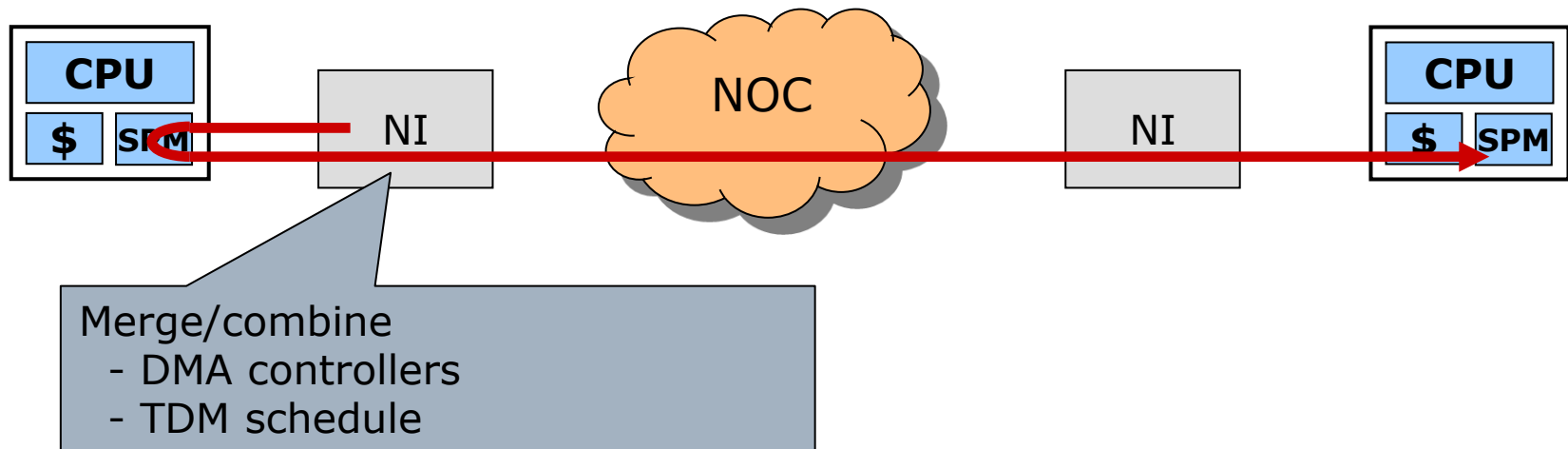
(c)

Traditional NI design (w. buffers)



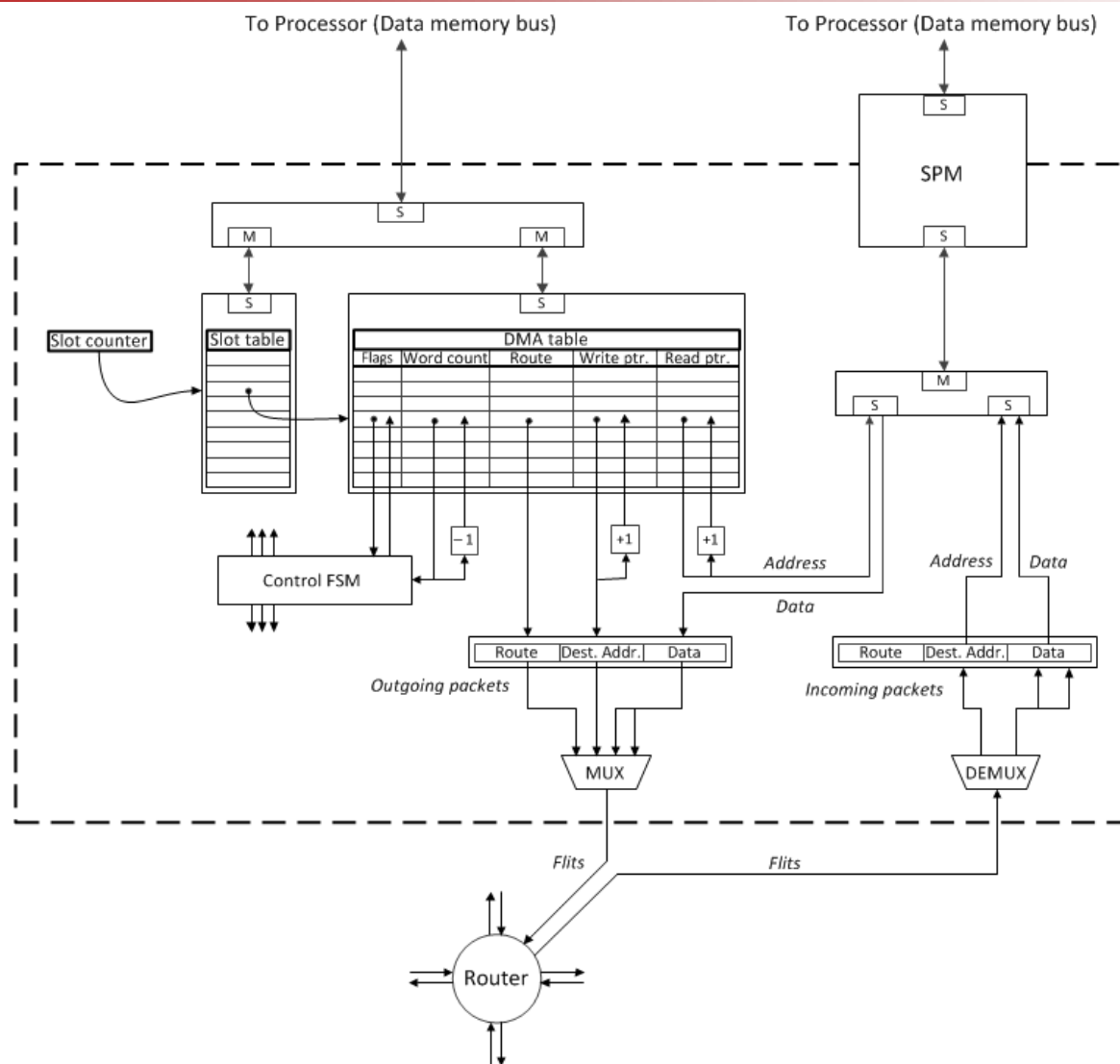
- Per-connection FIFO-buffers and flow control.
 - Accounts for 50-85% of NI hardware!

New T-CREST NI-microarchitecture



- DMA controllers in NIs and driven by TDM schedule.
 - Avoids FIFO-buffers and flow control.
 - Saves 50-85% of traditional NI hardware

New T-CREST NI-microarchitecture



Key features

- SPM used for clock domain crossing
- One DMA needed per connection, but only one active at any given time due to TDM. Enables efficient table-based implementation of DMA controllers.
- End-to-end (i.e., SPM-to-SPM) data transfer avoids buffering and flow control.

Summary

- Dual issue MIPS-style processor - Patmos
 - ◆ Predicated instructions
 - ◆ Typed load and store instructions
 - ◆ Special caches
 - ◆ Scratch pad memory
- Network on chip
 - ◆ Processor-to-processor
 - DMA driven block transfers (message passing)
 - Nature: All-to-all
 - TDM w. static schedule (aelite-like)
 - Asynchronous
 - ◆ Processor-to-memory controller (SDRAM)
 - Arbitration in NOC *and* in memory controller
 - Nature: All-towards-one

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>