The Time-predictable Multicore Architecture

T-CREST

Martin Schoeberl

Technical University of Denmark
New Architectures Needed

- Design a computer architecture for real-time systems
  - Worst-case execution time (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*
T-CREST Architecture

- Chip-multiprocessor for high performance
  - Target: 64 cores in an FPGA

- Time-predictable
  - Processor
  - Network-on-Chip (NoC)
  - Local memory (SPM, $)
  - SDRAM controller

- Integration in WCET analysis
- Compiler support for WCET analysis
T-CREST Outcome

- Provide a complete platform
  - Hardware in an FPGA
  - Supporting compiler and analysis tool

- Resulting designs in open source
  - BSD license
  - Simplifies cooperation

- Up to compiler
  - No operating system research
  - No Model of Computation research
  - No automatic parallelization research
Platform - Hardware

T-CREST Chip

Memory Tree

Memory Controller

SDRAM Memory

Patmos

NI

Patmos

NI

Patmos

NI

R

R

R

NI

Patmos

Dec

M$

S/D$

SPM

M$

S/D$

SPM

M$

S/D$

Memory Tree

Memory Controller

SDRAM Memory
T-CREST Partners

THE Open GROUP

AbsInt

TU WENIEN

DTU

The University of York

Technische Universiteit Eindhoven
University of Technology

GMV Innovating Solutions

intecS the Brainware company
Processor

- Time-predictable processor
  - Written in Chisel

- Called Patmos

- Flexibility to define the instruction set
  - A compiler is adapted for Patmos at TUV

- Co-design for low WCET of
  - Patmos
  - Compiler
  - WCET analysis
Patmos

- A research platform for real-time architecture (e.g. caches, SPM)
- RISC style microprocessor
- Dual issue VLIW
- Full predication – all instructions
- Split caches
  - Scratchpad memory (SPM)
  - Stack cache
  - Data cache
  - Method cache
Pipeline Overview

Fetch - Decode - Execute - Memory - Writeback

- M$  - RF
- IR   - Dec
- PC   - IR
- IR   - Dec
- RF   - S$
- RF   - D$
- RF   - SP
Patmos WCET Features

- No timing anomaly
  - As far as I know ;-
- Constant execution time of instructions
- 0 timing dependency between instructions
  - All caches miss in the same pipeline stage
  - Load/use dependency scheduled by the compiler
Compiler

- Adaption of LLVM
- Full support of Patmos ISA
- Support for stack cache instructions
  - Reserve, free, ensure
- Support for method cache
- Optimize for WCET
  - Integration with AbsInt aiT tool
  - Delivers flow facts for aiT
  - Single-path code generation
Network on Chip (NoC)

- 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

- **Clock domain**
- **Mesochronous NIs**
- **Asynchronous**
  - Packet switched
  - TDM based

The NOC

- CPU
  - $SPM

- NI

- CPU
  - $SPM

- CPU
  - $SPM

- CPU
  - $SPM

- CPU
  - $SPM
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
Argo – the T-CREST NoC

- TDM based NoC – statically scheduled
- Different packet sizes possible
- 3 stage pipelined router
  - Synchronous and asynchronous
- Network interface (NI)
  - Ticks at TDM clock (mesochronous)
    - *Drives* the asynchronous network
  - Time shared DMA machinery
  - SPM for clock domain crossing
A TDM-based Router

Link → HPU → Link

Link → HPU → Link

Link → HPU → Link

X bar

Payload

Header

flit

flit

Route
Network Interface for Argo

Traditional NI design

Argo micro-architecture
Argo NI 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
TDM Tree

- The TDM schedule at the memory interface
  - No buffering in the memory controller
- Pipelined tree and response channel
- Pipeline delay in tree is known
  - No buffering in tree nodes
- Distributed TDM arbitration at nodes
  - Just with the right offset
  - The packet knows when to go
WCET Analysis

- Adaption of AbsInt aiT tool
- Support of Patmos ISA
  - Predicated instructions
  - Dual-issue pipeline
- Simple method cache analysis
- Stack cache analysis
- Integration with compiler
  - platin
Software and Tools

- Operating systems (single core)
  - RTEMS
  - ARINC 653 OSPAT

- Operating system multi-core
  - MOSSCA from Augsburg

- NoC schedule generator

- NoC user library

- Two use cases from industry partners for platform evaluation
Using T-CREST

- Open source
  - Industry friendly BSD license
- Hosted at GitHub
  - https://github.com/t-crest
  - You can join the team and get write access
- Only (mostly) free tools used
- Public mailing list patmos-processor at Yahoo
- http://patmos.compute.dtu.dk/
FPGA Platforms
Simulation (SW)

- Processor simulation
  - SW simulator
  - HW based emulator
- LLVM compiler
- Just your laptop
Simulation (HW)

- Processor simulation
  - HW based emulator
  - Free wave viewer

- Multicore simulation
  - ModelSim, license for multi language
The Real Thing - Hardware

- Full T-CREST execution
  - 9 cores
- On $300 FPGA board
- Only free tools needed

Martin Schoeberl
Future Work

- EC project finished
  - Continue with DK funded project PREDICT
- We will keep the T-CREST name
- Platform for future EC funded projects
- Immediate next steps
  - Programming models
  - Full system simulator (C++)
  - Integration with SWEET (platin)
The T-CREST Team

[Images of team members]
Want to Learn at SYCHRON

- How to program T-CREST
- We need a programming model
  - That includes computation and communication
- Maybe synchronous languages would fit
- Support for multicore processor
  - Using message passing NoC
- Let’s team up
Summary

- We need 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 is mostly open source
- https://github.com/t-crest