RWTH AACHEN UNIVERSITY

MPSoC 2007

HySim: Fast Hybrid Processor Simulation for MPSoC Virtual Platforms

Rainer Leupers
Stefan Kraemer, Lei Gao

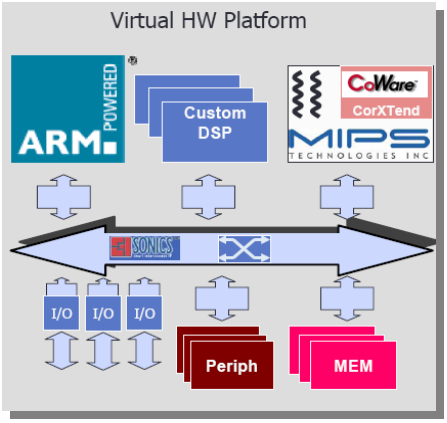
Software for Systems on Silicon (SSS)
RWTH Aachen University

ISS

Institute for Integrated Signal Processing Systems

What is a virtual platform?

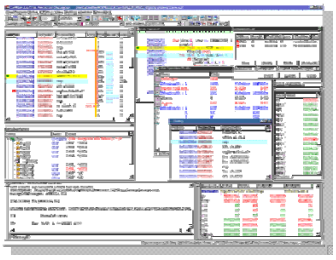
- **A SW model of a HW SoC platform**
- **Enables...**
  - HW platform architecture exploration and optimization
  - SW development, debugging, and optimization
  - Concurrent HW/SW design („HW/SW codesign“)
- **Requirements**
  - High simulation speed
  - Speed/accuracy trade-off
  - Flexibility
  - Usability for non-HW-experts



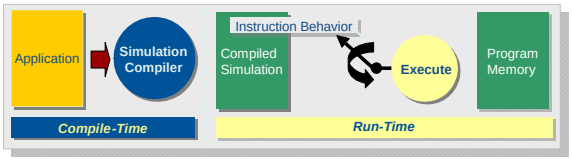
See e.g. MPSoC 2006 presentations from CoWare, Vast

Need for speed

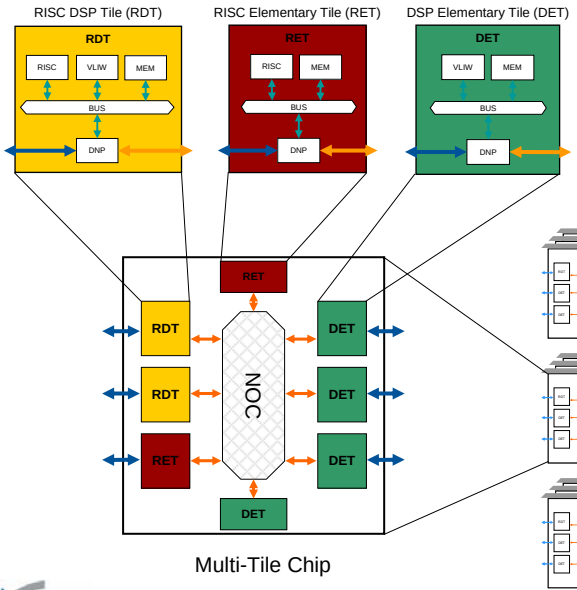
- **Instruction set simulator (ISS) is at the heart of virtual platforms**
- **ISS speed evolution**
  - Early interpretive: few KIPS
  - Fast interpretive: ~100 KIPS
  - Compiled: ~10 MIPS
  - SW Sim. Cache: ~10 MIPS
  - Binary translation: 50-100 MIPS



- **Challenge:**
  - Instruction-accurate ISS technology has been pushed to its limits
  - How to handle future many-core MPSoC?



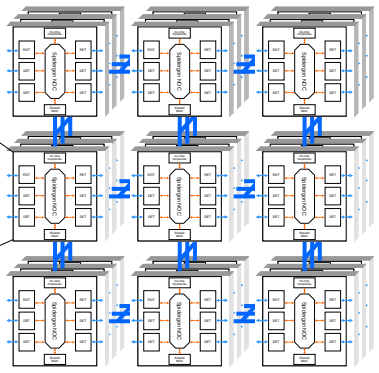
Example: SHAPES platform



Multi-Tile Chip

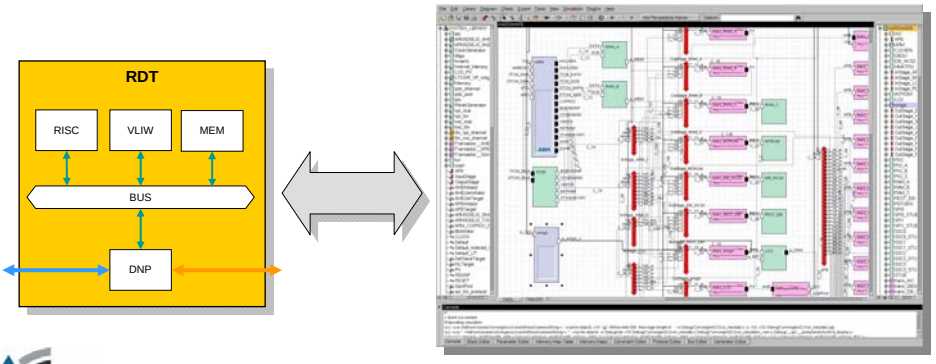
Scalable HW architecture for high-performance applications

Multi-Chip System



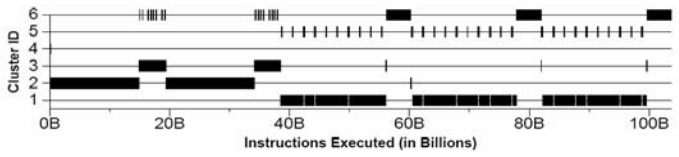
Virtual SHAPES platform

- **Single-tile model (ARM9, Magic VLIW DSP, memories, peripherals) developed with CoWare Virtual Platform Designer**
- **Reasonable speed for one tile**
- **But: difficult to scale to multi-tile, multi-chip**



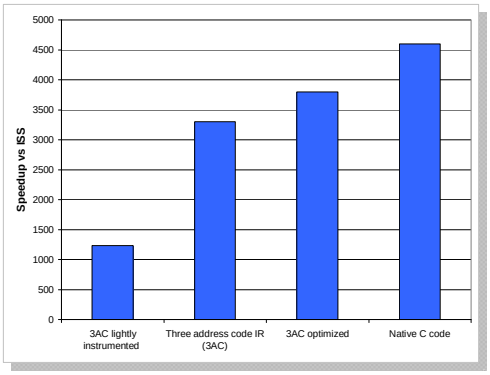
Alternative simulation technologies

- **Raising abstraction level is key**
- **Statistical simulation**
  - Profile application to collect statistics (block exec frequencies, instruction mix etc.)
  - Use statistical simulator to estimate performance probabilistically
  - No SW debugging facilities
- **Sampling based simulation**
  - Real applications show periodic behavior during execution
  - Detailed simulation only at representative program pieces
  - Fast forwarding in between
  - No random toggling between the two modes



What is left for accelerating simulation?

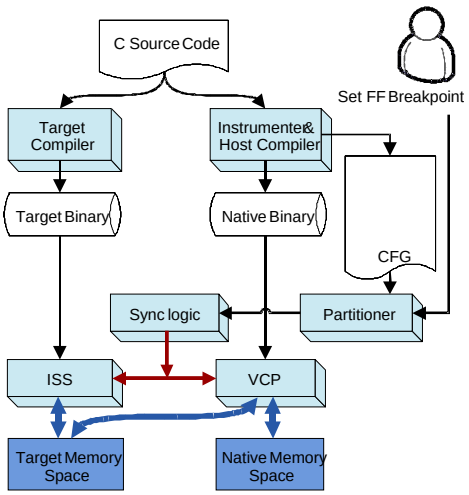
- **Experiment with DES application and MIPS core**
- **Baseline 1: traditional IA ISS**
  - Relatively slow
- **Baseline 2: native C code execution**
  - No target-specific information
- **~1000x speedup between ISS and native code theoretically available**
- **~100x may be realistic**
- **Goal: hybrid simulator, toggling between**
  - Fast native execution
  - Accurate target ASM simulation
  - Debugging enabled
  - Much faster than ISS



See also [Marques, Buels, et al., HPCA04]

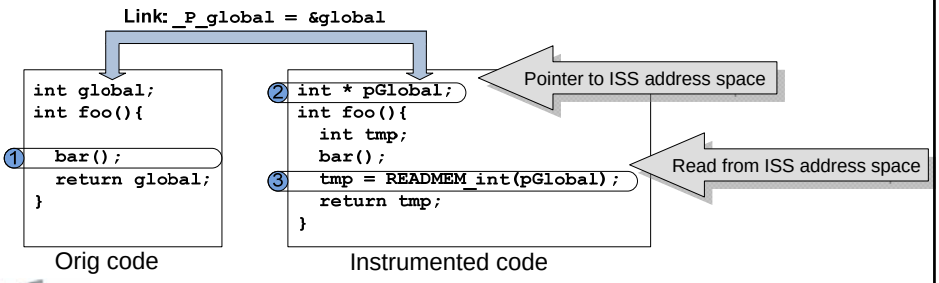
HySim approach

- **Target ISS and C virtual co-processor (VCP) co-exist in simulation environment**
- **User has single graphical debug interface**
- **Use scenario 1:**
  - User sets breakpoint at program point of interest
  - HySim fast-forwards to that breakpoint
  - ISS continues as usual
- **Use scenario 2:**
  - FF breakpoint at end of program
  - Fast SW performance estimation
- **Major challenge:**
  - Synchronisation of processor state between ISS and VCP
  - Requires C code instrumentation



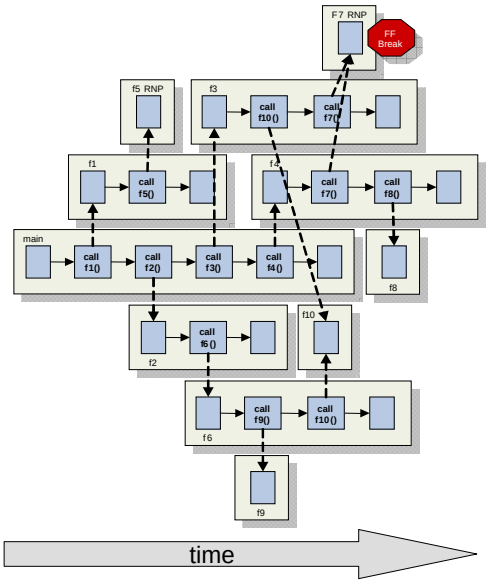
Code instrumentation principle

- Internally: translation of C app code to three address code to enable fine-grained source instrumentation
- Instrumented code employs interface functions to encapsulate function calls, global var access etc. for ISS/VCP toggling (+ optionally performance estimation on VCP)
- Current restriction: toggling only at function boundaries



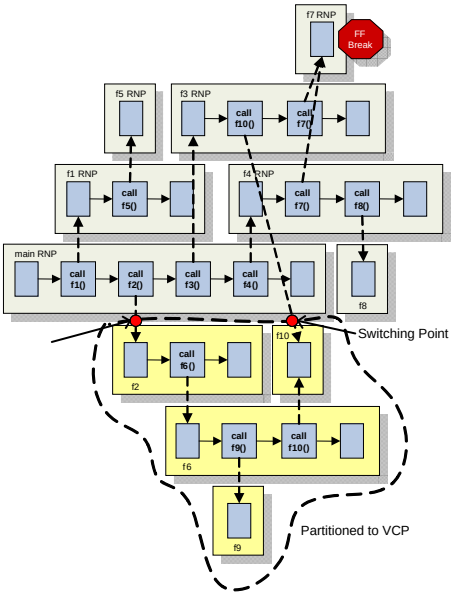
ISS/VCP code partitioning

- Function is called partitionable if it can be executed on VCP
- Some functions are non-partitionable by default
  - E.g. ASM libs, malloc, ...
- Setting FF breakpoint dynamically causes additional non-partitionable functions
  - Function f containing FF breakpoint
  - All functions on call stack from „main“ down to f
  - Otherwise no return from breakpoint function



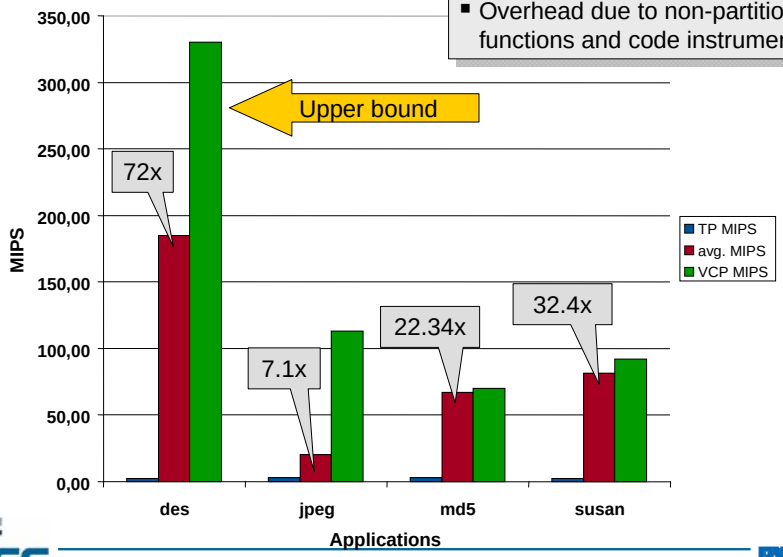
ISS/VCP code partitioning

- Code partitioner assigns remaining functions to VCP
- Induces switching points where control and state are transfered between ISS and VCP
- User perspective: FF breakpoint in debugger reached ASAP



Very preliminary results (HySim for MIPS core)

- FF breakpoint at end of program
- Overhead due to non-partitionable functions and code instrumentation



Status and future work

- **HySim prototype indicates**
  - High speedup potential vs instruction accurate ISS
  - Practicality (transparent to SW developer)
- **Short term:**
  - Minimizing instrumentation overhead
  - Accurate SW performance estimation
    - Currently only 10-20% off in cycle count, but only for MIPS
    - Cache simulation on-the-fly in VCP?
  - Porting to SHAPES virtual platform
- **Long term:**
  - Include fast bus/NoC simulation
  - Checkpoint/restart facility for MPSoC simulation
  - Parallel simulation on multicore host

Institute for Integrated Signal Processing Systems

Thank you !

www.iss.rwth-aachen.de

Institute for Integrated Signal Processing Systems