Validation of a High-Performance Emulator for SVF

Alastair Pidgeon, Paul Robinson, Sean McClellan, SciSys UK Ltd.
Fabrice Bellard, FFQTECH
Paulo Marques, Luís Pureza, University of Coimbra
Felice Torelli, ESA ESTEC TEC-SWS

Presented by:
Alastair Pidgeon

Outline

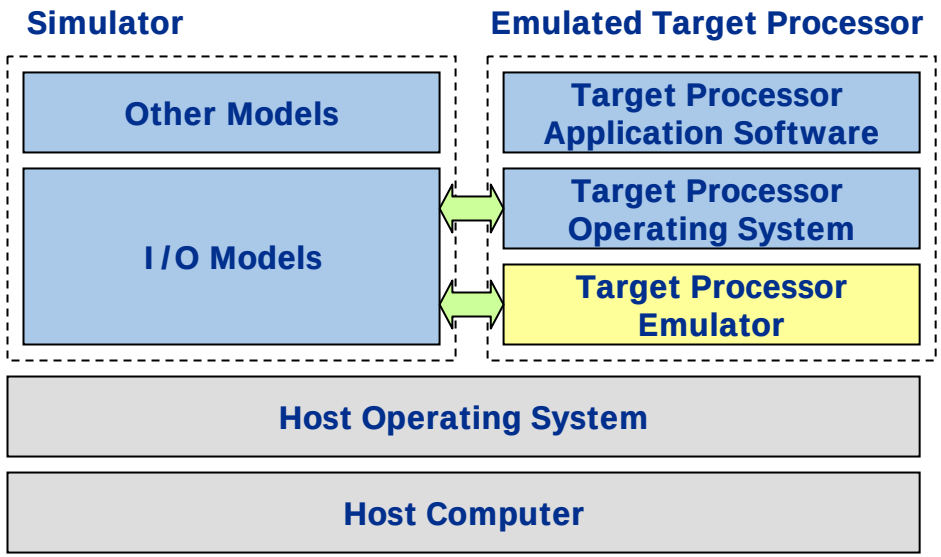
- Software Processor Emulator
- Motivations
- Dynamic Translation in a Nutshell
- QEMU => QERC/QERL
- Project Objectives
- Enhancements Performed
- Validation for SVF Use
- Results So Far
- Future Work
- Summary

Introduction

- Innovation Triangle Initiative (ITI) Project
  - ⇨ FFQTECH: Inventor
  - ⇨ University of Coimbra: Academia
  - ⇨ SciSys: Developer and End-User
- Objective is to demonstrate that a QEMU-based emulator can be used within an SVF
- Milestones
  - ⇨ KO: February 2008
  - ⇨ SRR: July 2008
  - ⇨ MTR: September 2008
  - ⇨ Delivery: Early 2009

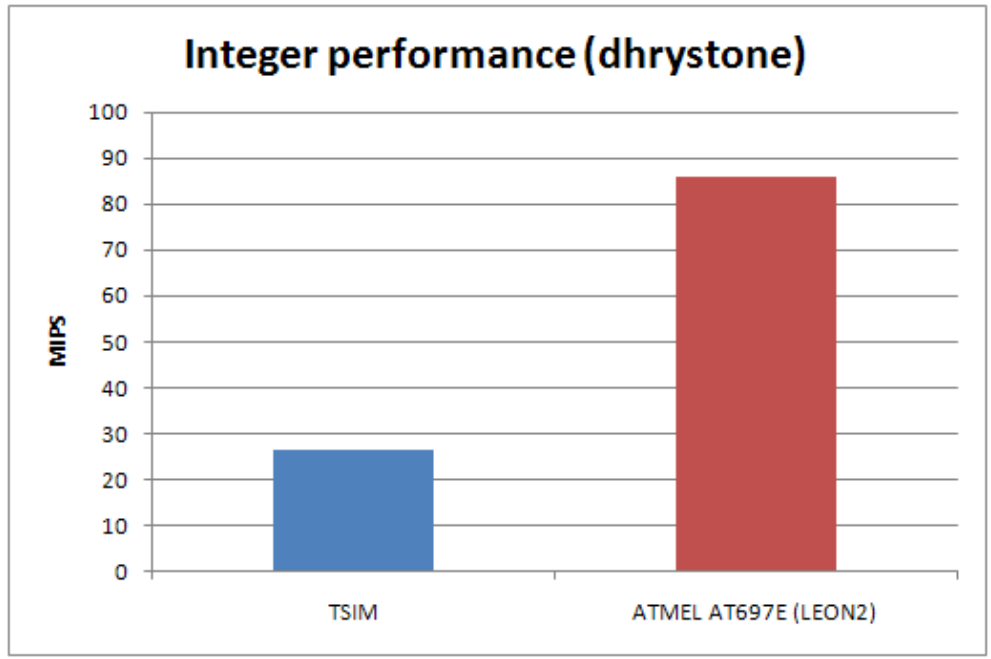
Software Processor Emulator

- Main Uses
  - ⇨ OBSW Development & Debug
  - ⇨ Open Loop Tests for OBSW
  - ⇨ Closed-Loop Software Validation
  - ⇨ Operations/ Training Simulators



Motivations

- Current processor emulators are 50-100 times slower than host
  - ⇨ Work for older generation of space processors
  - ⇨ Relied on increasing clock-speed to keep pace
  - ⇨ Mainly serial in nature - multi-core machines little help
- LEON2/3
  - ⇨ 3-5 times ERC32 performance
- Need a new approach
  - ⇨ Hardware
  - ⇨ Software
- Align to TSIM, the de facto standard

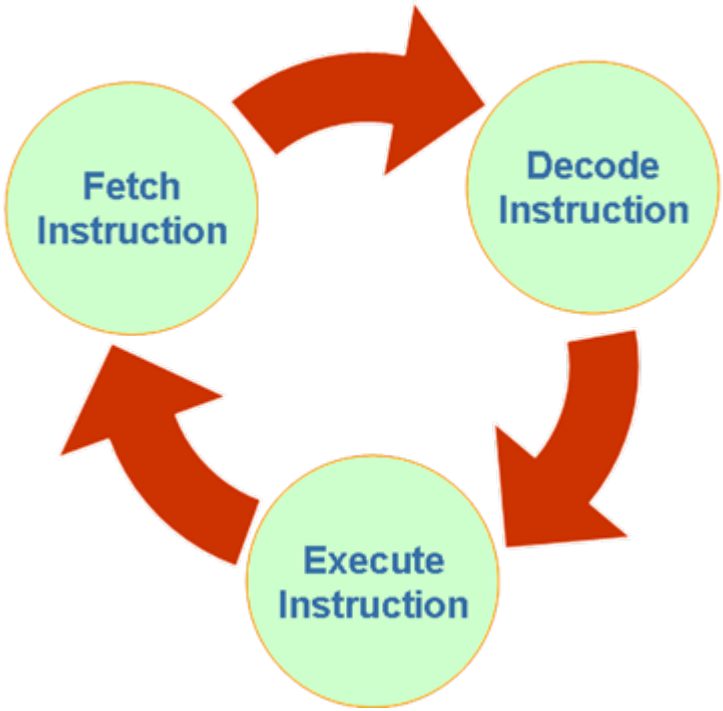


Dynamic Translation – In a Nutshell

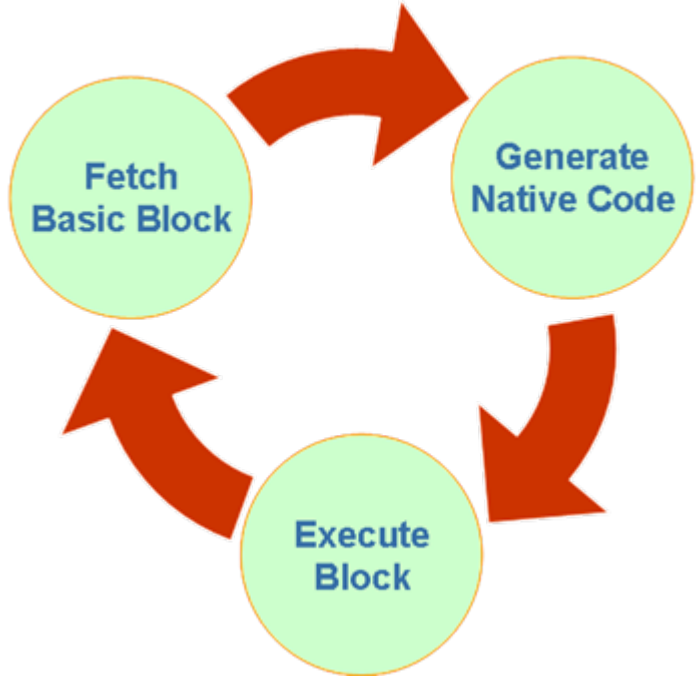
- Traditional emulators read each *target* instruction and translate to the equivalent *host* instruction(s) – every single time they are encountered
- Dynamic translation compiles *blocks* of target instructions to host instructions
- Blocks typically between branches, ~5-10 instructions long
- Blocks compiled on the fly and stored in memory
- When a block is encountered again it is retrieved from memory and executed
  - ⇨ This is the where the performance gain comes from
- BUT
  - ⇨ Execution at block level raises issues with the processor clock and I/O timing.

Intepretation vs. Dynamic Translation

Interpretation

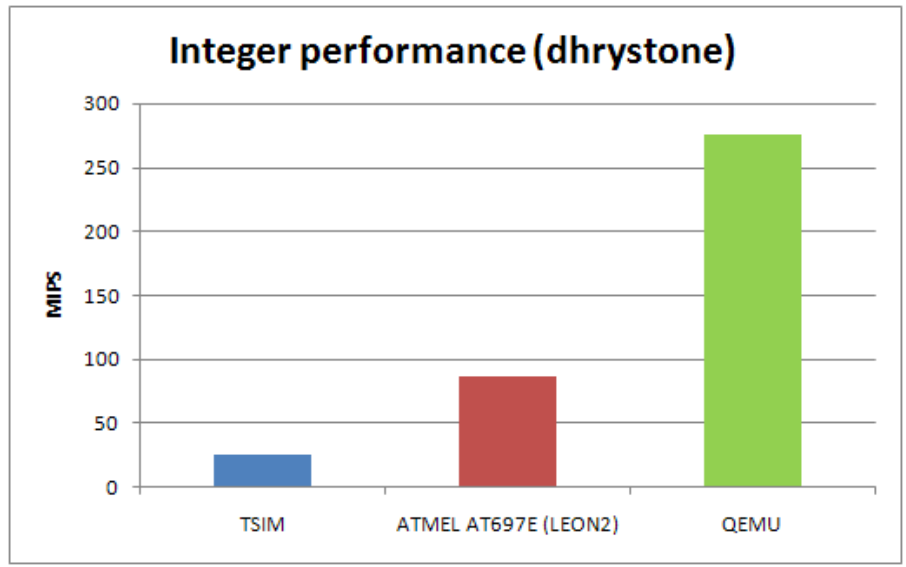


Dynamic Translation



QEMU => QERC/QERL

- QEMU Dynamic Translation Emulator
  - ⇨ Open Source
  - ⇨ Supports many target processors including SPARC
  - ⇨ FAST
- BUT
  - ⇨ Mainly used to emulate complete machine (not just a processor)
  - ⇨ No support for ERC32
  - ⇨ No support for LEON
  - ⇨ Virtual timers rely on host clock – no link to instructions executed
  - ⇨ No shared library
  - ⇨ No TSIM interface



Intel Pentium 4 EM64T @ 3.6 GHz

Generic QEMU Target and Host Support

CPU Architecture	TARGET	HOST	
x86	OK	OK	
x86-64	OK	OK	
ARM	OK	Testing	
SPARC	OK	Testing	
SPARC64	Dev only	Dev only	
PowerPC	OK	OK	
PowerPC64	Dev only	Dev only	
MIPS	OK	Testing	Dev only TCG
m68k (Coldfire)	OK	Dev only	
SH-4	Dev only	No	
Alpha	Dev only	Testing	Dev only TCG
CRIS	Dev only	No	
HP PA-RISC	No	Dev only	

Project Objectives

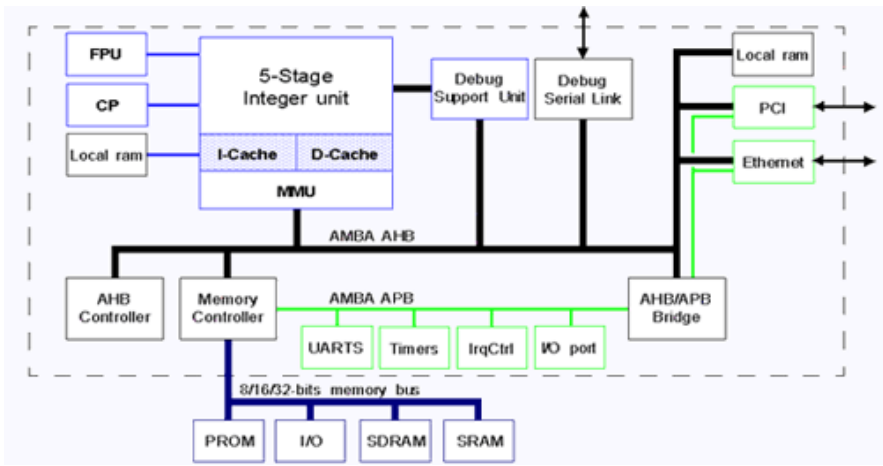
- Assess areas where QEMU would need to be enhanced
  - ⇨ Instruction Timing/IO
  - ⇨ Removal of compiler dependencies
  - ⇨ Further improved performance
- Perform selected enhancements and compare benchmarks with baseline QEMU
- Add a TSIM-like interface to allow plug-and-play
- Validate QERL/QERC
  - ⇨ Boot and run Linux/RTEMS and applications within QERL
  - ⇨ VSRF/RSVF 6 using “EagleEye” OBSW
  - ⇨ Validate in Aeolus Simulator

Enhancements Performed

- Many of the identified improvements were done by the open source community
- Tiny Code Generator – minimises compiler dependencies
- Performance Improvements
  - ⇨ Alignment checks
  - ⇨ Condition codes
  - ⇨ Register allocation
  - ⇨ Global register allocation for the register window pointer
  - ⇨ Better exception handling
  - ⇨ ~50% improvement over baseline QEMU 0.9.1
- Cycle Counting
  - ⇨ Addition of instruction latencies in each block
  - ⇨ No cache or memory timing taken into account
  - ⇨ Floating-point instruction timings approximate
- Added key elements of TSIM interface
  - ⇨ QEMU gdb interface used instead of TSIM

LEON2 AT697 Support Status

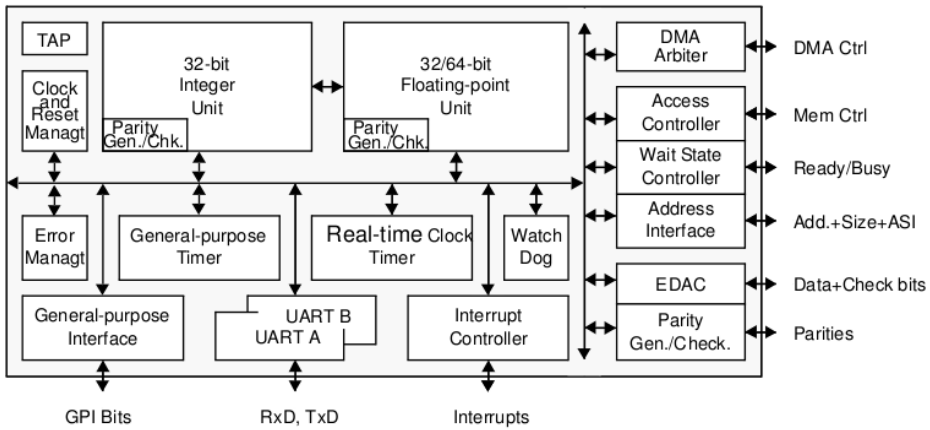
Peripheral	Status	Observations
Interrupt Controller	✓	
Secondary Interrupt Controller	✗	Not in AT697
Timer Unit	✓	No second timer yet
UARTs	✓	No UART2 yet
Parallel IO	✓	Handled as MMIO
LEON Configuration Register	✓	
Memory Configuration	✓	



esa FFQTECH University of Coimbra scisys

ERC32 Support Status

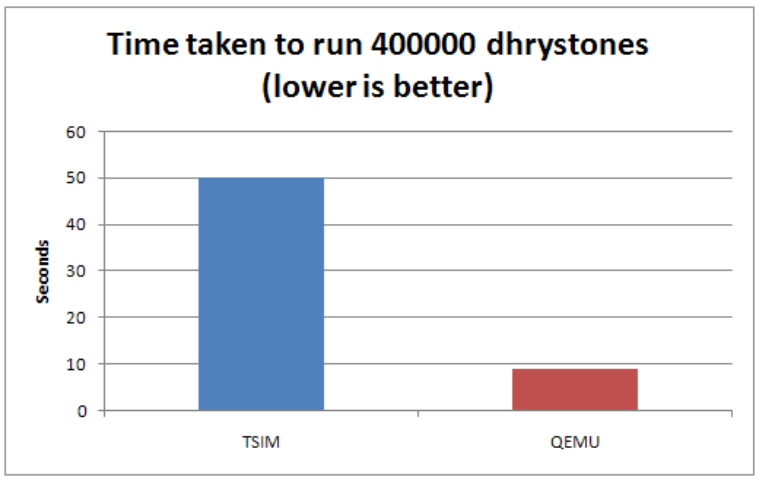
Peripheral	Status	Observations
Interrupt Controller	✓	
System Control Register	✓	
DMA	✗	Not used in OBSW
UARTs	✓	No UART B yet
Timers	✓	No watchdog yet
General Purpose Interface	✓	Handled as MMIO
Memory Configuration	✓	



esa FFQTECH University of Coimbra scisys

Results So Far

- Supports both ERC32 and LEON2
  - ⇨ The parts actually used by OBSW
  - ⇨ Configurable to ERC32 or LEON by command
  - ⇨ TSIM interface largely supported
- Approximately 5x faster than TSIM on same hardware
- Runs Linux & RTEMS within QERL
- Has run in VSRF/RSVF via TSIM interface but work ongoing
- Some fixes provided back to QEMU



Future Work

- Complete validation on VSRF/RSVF6
  - ⇨ Resolve VSRF/RSVF 6 issues
  - ⇨ Run with ERC32 version
  - ⇨ Run with LEON2 version
- Validate within the Aeolus Simulator
  - ⇨ Run real OBSW in a real-time, closed loop context for ERC32

Summary of QERC/QERL

- Derived from open source QEMU
- LGPL Licence applies
- Alignment with QEMU now maintained
  - ⇨ Allows improvements to QEMU to be incorporated
- Performance is ~5x that of conventional emulators
- Provides the ability to run OBSW in a simulated LEON2 faster than real-time
- Provides a low-cost alternative to TSIM for ERC32 and LEON based OBSW development and simulations
- Allows multiple processor emulations on a single machine
  - ⇨ But TSIM interface would need to be extended to specify because it does not allow a processor instance to be specified