Bringing up cycle-accurate models of RISC-V cores

ed.jones@embecosm.com

EMBECOSM®

About me and Embecosm

Work on open source tool chains at Embecosm

LLVM, GNU Binutils, Newlib, Energy efficiency in compilers (TSERO)

EMBECOSM®

Story time!

Synopsis: The Embecosm team surveys the RISC-V ecosystem, evaluating cores and the software tool chain, testing and benchmarking open source softcores.

Open source cores in the RISC-V ecosystem
Evaluation of the software tool chain
Simulating cores and testing
Benchmarking
Going forward

EMBECOSM®

Project goals

Evaluate existing open source cores
Evaluate the maturity of RISC-V tool chain
Basis for a project for a new specialized RISC-V core

Benefits of open source

- Rapid development – Start from a mature basis
- Give and take from the open source community

Risks

- Licensing issue – But most cores have liberal licenses
- Support – No problem, this is our expertise

EMBECOSM®

Requirements

Open source (obviously)
Bare metal
Small footprint
Reasonable performance
Extensible – Expecting to need custom extensions

Survey of existing cores

ASTC Systems
Bluespec
Clarvi – Simon Moore / Rob Mullins, Cambridge University
Codasip
F32c
lowRISC - lowRISC not-for-profit / Cambridge University
mriscv - OnchipUIS / Ckristian Duran
Orca – VectorBlox
Phalanx - Jan Gray / GRVI
PicoRV32 - Clifford Wolf
RI5CY - PuLP Platform / ETHZ
River – GNSS Sensor
Rocket – Free Chips Project / UCB-Bar
Shakti – IIT Madras
Sodor – UC Berkeley / Christopher Celio
Tom Thumb – Maik Merten
TurboRav – Sebastian Boe
URV – CERN / Tomasz Wlostowski
Yarvi – Tommy Thorn
Z-Scale – UC Berkeley / Yunsup Lee, Albert Ou, Albert Magyar

EMBECOSM®

Shortlist

ASTC Systems
Bluespec
Clarvi – Simon Moore / Rob Mullins, Cambridge University
Codasip
F32c
lowRISC - lowRISC not-for-profit / Cambridge University
mriscv - OnchipUIS / Ckristian Duran
Orca – VectorBlox
Phalanx - Jan Gray / GRVI
PicoRV32 - Clifford Wolf
RI5CY - PuLP Platform / ETHZ
River – GNSS Sensor
Rocket – Free Chips Project / UCB-Bar
Shakti – IIT Madras
Sodor – UC Berkeley / Christopher Celio
Tom Thumb – Maik Merten
TurboRav – Sebastian Boe
URV – CERN / Tomasz Wlostowski
Yarvi – Tommy Thorn
Z-Scale – UC Berkeley / Yunsup Lee, Albert Ou, Albert Magyar

EMBECOSM®

PicoRV32

Creator	Clifford Wolf
ISA	RV32IMC
Open source	Yes – ISC Licence
Bare metal	Yes
Small footprint	Very (750-1K LUTS / 398 PLBs)
Reasonable performance	Yes (400-700Mhz on Xilinx 7-series)
Extensible	Yes, also has an interface for coprocessors
Comments	Clifford has also been formally verifying core with his riscv-formal verification framework

https://github.com/cliffordwolf/picorv32

RI5CY – PuLP Platform

Creator	ETH Zürich
ISA	RV32IMC + F + PULP extensions
Open source	Yes – Solderpad Hardware License
Bare metal	Yes
Small footprint	“Yes” - Also see zero-riscy and micro-riscy (part of the PULPino project)
Reasonable performance	Maybe? (50-75MHz on Zynq)
Extensible	???
Comments	Mature. Has been taped out at 65nm (PULPino). Achieves better IPC than PicoRV32

https://github.com/pulp-platform/riscv

Rocket Chip Generator

Creator	Free Chips Project
ISA	Any (32 or 64)
Open source	Yes
Bare metal	Yes
Small footprint	Possibly not
Reasonable performance	???
Extensible	Yes through Chisel (Scala based HDL)
Comments	The canonical way of generating SoCs. Has been used to tape out multiple chips, some of very high performance.

https://github.com/freechipsproject/rocket-chip

Tool chain Implementation

Using existing GNU tool chain for bare metal.
Snapshots of:

- Binutils
- GCC
- GDB
- Newlib

Also generated Verilator models of PicoRV32 and Ri5CY
Created a GDBServer to interface with GDB and drive the models through RSP protocol

Overall tool chain is pretty mature, but needed some customizations

Tool chain Customization

Implemented a number of changes to the tool chain for bare metal targets

- Update startup code (crt0), set stack pointer (+align)
- Implement File I/O in GDBServer
- Implement syscalls in GDBServer
- Add RI5CY interrupt vector table
- Minimize size of newlib

+ myriad minor bug fixes + tweaks:
RI5CY always starts at boot address, unsupported SystemVerilog in Verilator, trial and error with RI5CY memory interface, gdb fixes, implementing software breakpoints ...

EMBECOSM®

Newlib Customization

Minimal binary was ~2kB, a basic call to puts and printf were ~16kB and ~40kB respectively (way too big!).

To reduce minimum binary size:

- Put each syscall in a separate file
- Don’t call memset to zero .bss in crt0
- Don’t call constructors/destructors on startup/exit

To reduce puts/printf size:

- Build newlib with -ffunction-section -fdata-sections + use gc-sections
- Build newlib with all the “size reduction” flags enabled

Newlib Customization

‘configure’ flags to reduce footprint of newlib:

--disable-newlib-fvwrite-in-streamio
--disable-newlib-fseek-optimization
--enable-newlib-nano-malloc
--disable-newlib-unbuf-stream-opt
--enable-target-optspace
--enable-newlib-reent-small
--disable-newlib-wide-orient**
--disable-newlib-io-float**
--enable-newlib-nano-formatted-io**

	Before	After
puts	15836	6756
printf	40504	9432

**Aggressive flags which change behaviour of sprintf/printf

EMBECOSM®

Testing and benchmarking

RISC-V ISA Test suite

- Basic validation of correctness

GCC regression tests

- Assorted compilation and execution tests

BEEBS

- Bristol Embecosm Embedded Benchmark Suite
- From WCET, MiBench, DSPStone
- Used for the MAGEEC project
- No I/O, low memory, short runtime

EMBECOSM®

PicoRV32 GCC Results

Outcome	Count
Expected passes	86143
Unexpected failures	530
Unexpected successes	4
Expected failures	147
Unresolved tests	124
Unsupported tests	2540

Causes of failures: No support for I/O
Unresolved: Short timeouts

RI5CY GCC Results

Outcome	Count
Expected passes	86842
Unexpected failures	27
Unexpected successes	4
Expected failures	147
Unresolved tests	189
Unsupported tests	2540

Causes of failures: No ctor/dtor support, fence, low RAM, libgcc unwind, hosted env, upstream compilation failures
Unresolved: Short timeouts. More time gives only 7 timeouts

EMBECOSM®

BEEBS

- Some benchmarks excluded (timeouts + self check issues)
- Results consistent between PicoRV32 and RI5CY
- PicoRV32 took around 4x the cycles of RI5CY
- Although PicoRV32 could be clocked higher, RI5CY gets greater IPC

Morals

Using open source components simplified things a great deal. Had a decent tool chain and core to start from. No reinventing the wheel.

There are already a lot of open RISC-V cores out there – Less on high perf side, and more softcores than ASICs

The RISC-V tools are very good.

Moving Forward

RISC-V Ecosystem up and running
https://github.com/embecosm/riscv-toolchain

- “orconf” branch
- See README.md

RISC-V tools provided everything with minimal pain
Plenty of open RISC-V cores to choose from

Now we’re moving forward with a RI5CY based core

EMBECOSM®

Ongoing Work

Supporting development of a new core based on RI5CY
Extends RI5CY to 64-bits
Standard RI5CY + custom extensions
Multicore

Working on a new RI5CY assembler using CGEN
Want to quickly add support for new instructions to tool chain

Thanks for listening

https://github.com/embecosm/riscv-toolchain

ed.jones@embecosm.com

EMBECOSM®