Bringing up cycle-accurate models of RISC-V cores

ed.jones@embecosm.com
About me and Embecosm

Work on open source tool chains at Embecosm

LLVM, GNU Binutils, Newlib, Energy efficiency in compilers (TSERO)
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
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
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
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
# PicoRV32

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

https://github.com/cliffordwolf/picorv32
## RI5CY – PuLP Platform

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

[https://github.com/pulp-platform/riscv](https://github.com/pulp-platform/riscv)
# Rocket Chip Generator

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

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 ...
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**

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>puts</td>
<td>15836</td>
<td>6756</td>
</tr>
<tr>
<td>printf</td>
<td>40504</td>
<td>9432</td>
</tr>
</tbody>
</table>

**Aggressive flags which change behaviour of sprintf/printf
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
## PicoRV32 GCC Results

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected passes</td>
<td>86143</td>
</tr>
<tr>
<td>Unexpected failures</td>
<td>530</td>
</tr>
<tr>
<td>Unexpected successes</td>
<td>4</td>
</tr>
<tr>
<td>Expected failures</td>
<td>147</td>
</tr>
<tr>
<td>Unresolved tests</td>
<td>124</td>
</tr>
<tr>
<td>Unsupported tests</td>
<td>2540</td>
</tr>
</tbody>
</table>

Causes of failures: No support for I/O
Unresolved: Short timeouts
**RI5CY GCC Results**

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected passes</td>
<td>86842</td>
</tr>
<tr>
<td>Unexpected failures</td>
<td>27</td>
</tr>
<tr>
<td>Unexpected successes</td>
<td>4</td>
</tr>
<tr>
<td>Expected failures</td>
<td>147</td>
</tr>
<tr>
<td>Unresolved tests</td>
<td>189</td>
</tr>
<tr>
<td>Unsupported tests</td>
<td>2540</td>
</tr>
</tbody>
</table>

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.
• 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
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