Enhanced Tools for RISC-V Processor Development and Customization

Zdeněk Přikryl (prikryl@codasip.com)
Chris Jones (jones@codasip.com)
Who is Codasip?

- The **leading provider** of RISC-V processor IP
- Company founded in 2014 in the Czech Republic
- Founding member of the RISC-V Foundation, [www.riscv.org](http://www.riscv.org)
- Member of several working groups in the Foundation
- Actively contributing to LLVM and other open-source projects
- Now **Codasip GmbH**
  - Headquarters in Munich, Germany
  - R&D in Brno, Czech Republic
  - Offices in Silicon Valley, US, and Shanghai, Pudong PRC
Codasip introduced its first RISC-V processor in November 2015

- **Codasip Bk** = portfolio of RISC-V processors
- **Codasip Studio** = unique design automation toolset for easy processor modification
  - Performance/power efficiency and low-cost
  - Algorithm acceleration (DSP, security, audio, video, etc.)
  - Profiling tools of embedded SW for tailoring processor IP
- **CodAL** = Codasip’s own proprietary C-like language for processor architecture description
Bk: Customizable RISC-V Cores

Bk = the Berkelium series, Codasip’s RISC-V processors

✓ Available immediately
✓ Pre-verified, tape-out quality IP
  - Users do not need to verify IP
✓ Industry-standard interfaces
  - AMBA for instruction and data bus
  - JTAG (4pin/2pin) for debugging

✓ Fully customizable
  - Support for all RISC-V ISA standard extensions
  - Enable easy creation of performance-enhancing resources, such as:
    • Custom registers for computations
    • Custom control-status registers
    • Novel interfaces such as GPIO, FIFO, scratch-pad memory
  - Even pipeline modifications are possible
    • Bk core CodAL source as the starting point for your own RISC-V core
Bk Cores Roadmap

Comprehensive offering including new advanced designs

Bk3
- Entry-level 32bit RISC-V core

Bk5, Bk5-64
- 32bit and 64bit RISC-V cores with balanced pipeline

Bk7
- Linux-ready 64bit RISC-V core

Future Bk
- High-performance RISC-V cores
  - Advanced pipeline
  - Advanced DSP features
- Energy-efficient/low power RISC-V cores

All Bks
- Rich set of configuration options
- Fully customizable
Standard and Custom Extensions

RISC-V offers a wide range of ISA modules:
- I/E for integer instructions
- M for multiplication and division
- C for compact instruction
- F/D for floating point operations
- WIP: B, P, V, ...

However, it may not be enough for your application domain or if you are looking for a key differentiator...

RISC-V allows custom extensions
SDK must be aware of the custom extensions
High level of automation needed
Codasip has tools for this task: Codasip Studio
Why Customized Tools?

One of the biggest advantages of the RISC-V open ISA is **customization**. However, a customized processor also needs a customized SDK…

Standard customization (manually adding custom ISA extensions):

1. Model and simulate a new instruction
2. Modify the compiler
3. Modify assembler
4. Add support in the debugger
5. Verify, verify, verify…

→ **Challenging, time-consuming, expensive**

Benefits of automatic generation of customized tools:

- Reduced time needed for tool modification
- Reduced cost of custom processor development
- The resultant processor is easily programmable using standard C/C++
- Proven open-source technologies and frameworks allow for easy integration
What is Codasip Studio?

A unique collection of tools for **fast & easy modification** of RISC-V processors. **All-in-one**, highly automated. Introduced in 2014, **silicon-proven** by major vendors.

Customization of base instruction set:
- Single-cycle MAC
- Custom crypto functions
- And many more…

Complete IP package on output:
- C/C++ LLVM-based compiler
- C/C++ Libraries
- Assembler, disassembler, linker
- ISS (incl. cycle accurate), debugger, profiler
- UVM SystemVerilog testbench

**Codasip Studio**

**CodAL** – processor description language

```plaintext
element i_mac {
  use reg as dst, src1, src2;
  assembly { "mac" dst "," src1 "," src2 };
  binary { OP_MAC dst src1 src2 0:bit[9] };
  semantics {
    rf[dst] += rf[src1] * rf[src2];
  };
}
```

Integrated processor development environment

**RTL Automation**

**Verilog**  **VHDL**

**SDK automation**

**C/C++ LLVM-based compiler**

**C/C++ Libraries**

**Assembler, disassembler, linker**

**ISS (incl. cycle accurate), debugger, profiler**

**UVM SystemVerilog testbench**

Codasip GmbH
CodAL Models

- Easy-to-understand C-like language that models a rich set of processor capabilities
- All Codasip processors are created and verified using CodAL
- Multiple microarchitectures can be implemented in a single CodAL model
- CodAL models are provided to Codasip IP customers as a starting point for their own processor optimizations and modifications

```c
/* Multiply and accumulate: semantics
dst += src1 * src2 */

element i_mac {
    use reg as dst, src1, src2;
    assembler { "mac" dst "," src1 "," src2 };  
    binary { OP_MAC:8 dst src1 src2 0:9 }; 
    semantics {
        rf[dst] += rf[src1] * rf[src2];  
    }; 
}
```
Example: B Extension *Functional Model*

- Written in CodAL
  - in 10 days by a single engineer
- 900 lines of code
- Software development kit (SDK) automatically generated by Studio, including
  - Instruction set simulator (ISS)
  - Profiler to check the impact of the extensions
  - C compiler
    - Able to use a subset of instructions automatically (rotations, compact instructions, shifts, etc.)
Example: B Extension *Implementation* Model

- Written in CodAL
  - in 3 weeks by a single engineer
- 1500 lines of code
- Hardware design kit (HDK) automatically generated by Studio, including
  - RTL
  - Testbench
  - UVM-based verification environment

```c
#ifndef OPTION_EXTENSION_B
  case SLO:
    ex_result = ones_shifter_32(SLO, ex_aluop1, ex_aluop2);
    break;
  case SRO:
    ex_result = ones_shifter_32(SRO, ex_aluop1, ex_aluop2);
    break;
  case ANDC:
    ex_result = (uxlen)ex_aluop1 & (~ ex_aluop2);
    break;
  case ROTP :
    ex_rresult = ex_aluop1 >>> ex_aluop2;
    break;
  case ROTl :
    ex_result = ex_aluop1 <<< ex_aluop2;
    break;
  case CTZ:
    ex_result = codasip_ctlz_uint32(ex_aluop1);
    break;
  case CLZ:
    ex_result = codasip_cttz_uint32(ex_aluop1);
    break;
#endif
```
Processor IP Verification

- Strong methodology based on standardized approach, simulation, and static formal analysis
- Consistency checker
- Random assembler program generator
- UVM verification environment
  - Environment in SystemVerilog generated automatically by Codasip Studio
  - Checking if RTL corresponds to specification
Bk Core Customization with Codasip Studio

Start from Bk3/5/7 cores
1. Add instructions
2. Add resources
3. Modify pipeline
4. ...

Codasip Studio Toolset

Your RISC-V CodAL Models

Your RISC-V HDK
- RTL models
- Synthesis scripts
- Verification models and simulators
- Virtual prototypes

Your RISC-V SDK
- Compiler
- Assembler
- Linker
- Debugger
- IDE

Profiling of embedded application SW enables processor optimizations

ISA extensions are quickly implemented and analyzed during design space exploration
Summary

1. Codasip is the leading provider of commercial-quality RISC-V IP
   • Comprehensive off-the-shelf portfolio
     • From 32bit embedded to 64bit Linux-ready cores
     • Complete, fully verified IP packages
     • Available immediately
   • Full-time, highly professional customer support staff

2. Codasip offers easy, automatized way to customize RISC-V
   • Customization brings more performance, lower power/area, and differentiation
   • Codasip provides a complete set of tools and resources to customize:
     • CodAL – C-like language for processor description
     • Codasip Studio – a complete customization toolset
Thank you!

Questions?

prikryl@codasip.com
www.codasip.com
Codasip GmbH