

# Enhanced Tools for RISC-V Processor Development and Customization

Zdeněk Přikryl ([prikryl@codasip.com](mailto:prikryl@codasip.com))



# Who is Cudasip?



- 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 **Cudasip GmbH**
  - Headquarters in Munich, Germany
  - R&D in Brno, Czech Republic
  - Offices in Silicon Valley, US, and Shanghai, Pudong PRC

# Codasip Solutions

- **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
- **SweRV Support Package** = Contains everything needed to deploy a Western Digital's SweRV core(s)
  - Useful tools
  - Sample deliverables
  - Support for 3<sup>rd</sup> party EDA Tools
  - Technical support
  - Available exclusively from Codasip



*Codasip introduced its first RISC-V processor in November 2015*

# The Bk Series

Codasip RISC-V Cores

# Bk: Customizable RISC-V Cores

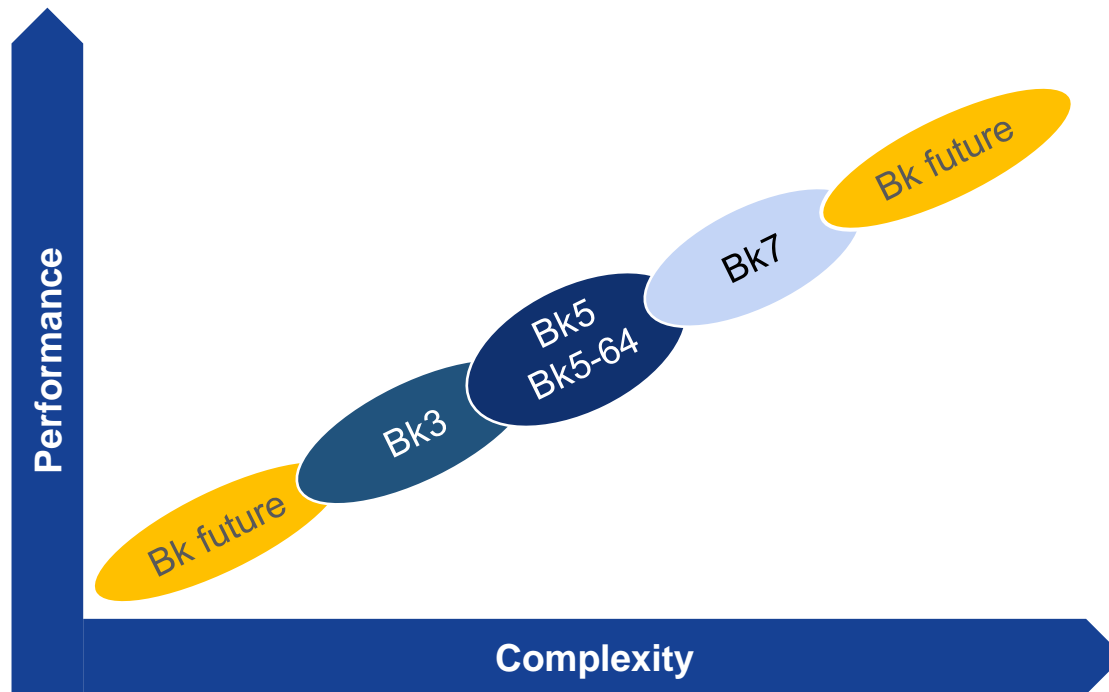


**Bk** = the Berkelium series, Cudasip's RISC-V processors

- ✓ **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*

# Bk3 Specification

- Base ISA
  - RV32I/RV32E (32bit)
- Available ISA extensions
  - Optional C, M, F extensions
- Possible ISA extensions
  - B, A, P and other extensions
- Available User (U) privilege mode

## Processor configuration options

### Multiplier implementation (M)

- Fully parallel high-performance design
- Small serial multiplier option

### Caches

- Instruction and data cache
  - Default: 16KB 4way I\$, D\$. User can ask for a different size
- Cache option with AHB or AXI Bus options
  - Default: AHB (AHB Lite) 32bit

### Debug/JTAG

- Nexus-based
- Available RISC-V compliant implementation

### Interrupt controller

- 32 interrupt inputs

# Bk5 Specification

- Base ISA
  - RV32I (32bit) or RV64I (64bit)
- Available ISA extensions
  - C, M, F, D extensions
- Possible ISA extensions
  - B, A, V, P and other extensions
- Available User (U) privilege mode

## Processor configuration options

### Multiplier implementation (M)

- Fully parallel high-performance design
- Small serial implementation option available

### Caches

- Instruction and data cache
  - Default: 16KB 4way I\$, D\$. Different sizes optional
- Cache option with AHB or AXI Bus options
  - Default: AHB (AHB\_Lite) 32bit

### Branch prediction

- BTB, BHT, RAS

### Debug/JTAG option

- Nexus-based
- Available RISC-V compliant implementation

### Interrupt controller option

- 32 or 64 interrupt inputs



# Bk7 Specification

- Base ISA
  - RV64I (64bit)
- Available ISA extensions
  - IMAC
  - GC
- Possible ISA extensions
  - B, V, P and others
- MMU for all configurations

## Processor configuration options

### Multiplier implementation (M)

- Fully parallel high-performance design

### Cache options (cache mandatory)

- Instruction and data cache
  - Default: 16KB 4way I\$, D\$. Different sizes optional
- Cache option with AHB or AXI Bus options
  - Default: AHB (AHB\_Lite) 32bit

### Branch prediction

- BTB, BHT, RAS

### Debug/JTAG option

- Available RISC-V compliant implementation

### Interrupt controller option

- 64 interrupt inputs

# 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**...



# 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

# Codasip Studio

Unique Toolset for RISC-V Customization

# What is Cudasip 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

## Cudasip Studio



RTL Automation

**Verilog**   **VHDL**

SDK automation



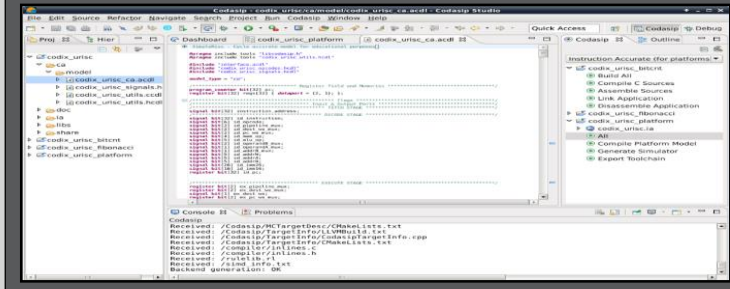
Verification Automation



### CodAL – processor description language

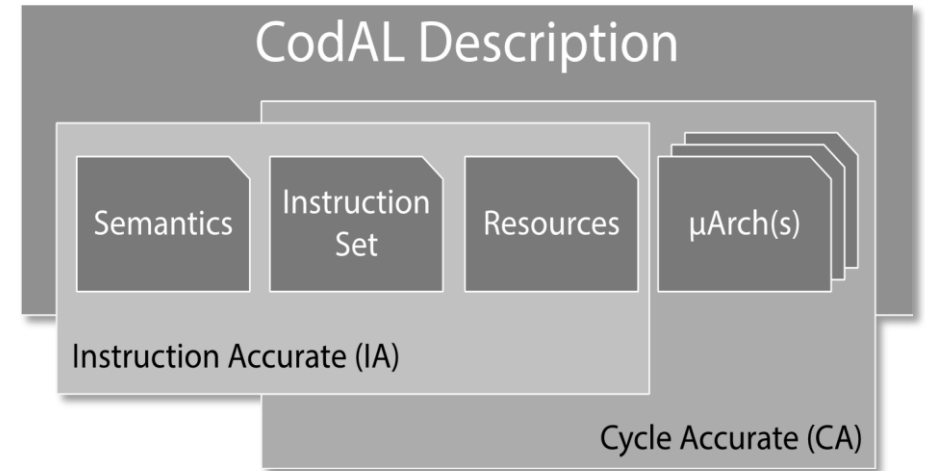
```
element i_mac {  
    use reg as dst, src1, src2;  
    assembly { "mac" dst ",," src1 ",," src2 };  
    binary { OP_MAC dst src1 src2 0:bit[9] };  
    semantics {  
        rf[dst] += rf[src1] * rf[src2];  
    };  
};
```

### Integrated processor development environment



# CodAL Models

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



```
/* 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.)

```
element i_gzip
{
    use opc_gzip as opc;
    use reg_any as dst, src1;
    use shift_imm as imm ;
    assembler { opc dst ", " src1 ", " imm};
    binary { opc[OPC_FRAG_SHIFT] imm src1 opc[OPC_FRAG1] dst opc[OPC_FRAG0] };
    semantics
    {
        rf_gpr_write (dst, gzip_uhlen(rf_gpr_read(src1), imm));
    };
};
set isa_b += i_gzip;
```

```
uXlen gzip_uhlen (const uXlen rsc1 , const uXlen mode)
{
    uint32 zip_mode, x ;
    x = rsc1;
    zip_mode = mode & 31;
    if(zip_mode & 1)
    {
        if(zip_mode & 2)
            x = gzip_stage (x, MASK_ZIP2_L, MASK_ZIP2_R, 1);
        if(zip_mode & 4)
            x = gzip_stage (x, MASK_ZIP4_L, MASK_ZIP4_R, 2);
        if(zip_mode & 8)
            x = gzip_stage (x, MASK_ZIP8_L, MASK_ZIP8_R, 4);
        if(zip_mode & 16)
            x = gzip_stage (x, MASK_ZIP16_L, MASK_ZIP16_R, 8);
    }
}
```

# Example: B Extension *Implementation* Model

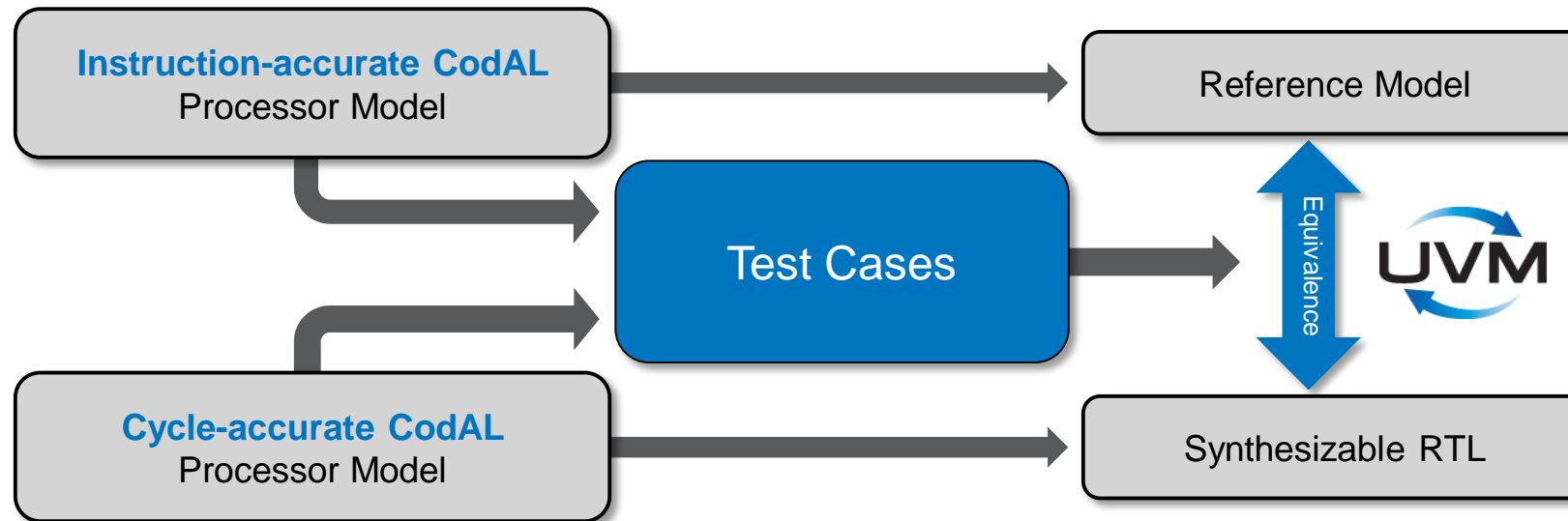
- Written in CodAL
  - in 3 weeks by a single engineer
- 1500 lines of code
- Hardware design kit (HDK) automatically generated by Studio, including
  - RTL
  - Testbench
  - UVM-based verification environment

```
#ifdef 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 ROTR :
        ex_result = 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 Cudasip Studio
  - Checking if RTL corresponds to specification



# Bk Core Customization with Codaship Studio

Start from Bk3/5/7 cores

1. Add instructions
2. Add resources
3. Modify pipeline
4. ...

Your RISC-V CodAL Models

Codaship Studio  
Toolset

Your RISC-V  
HDK

Hardware Design Kit

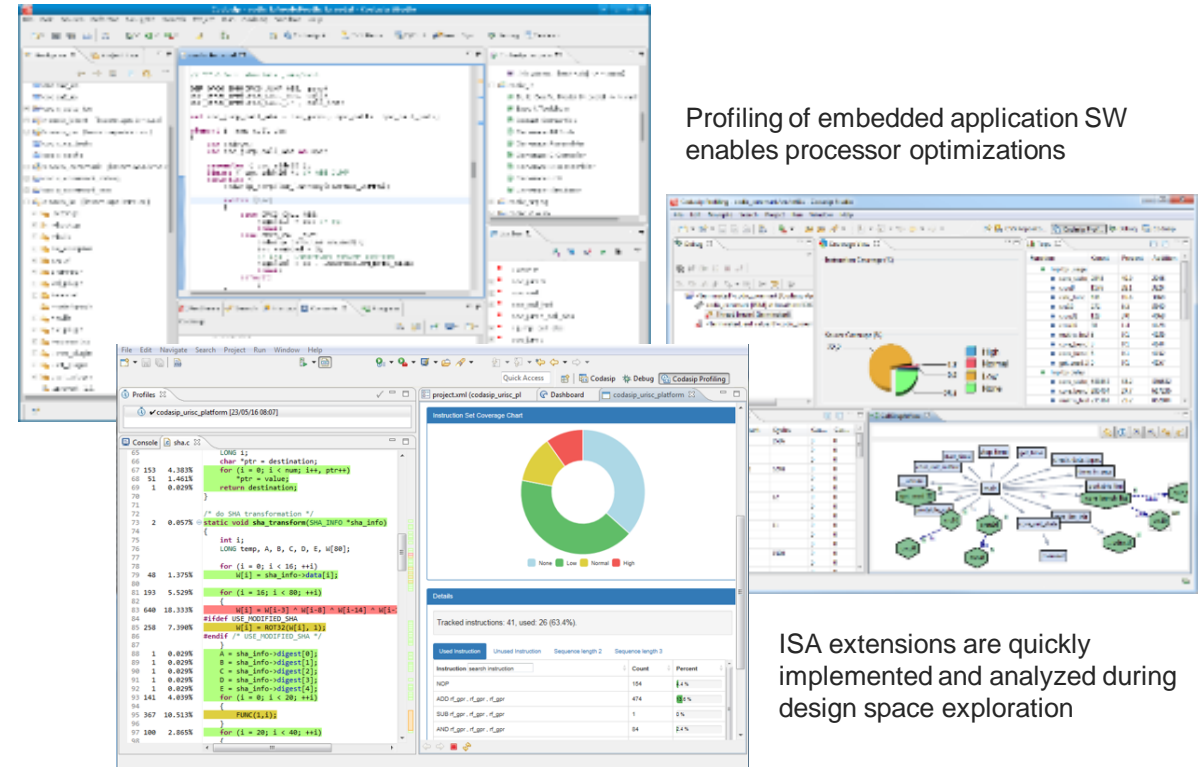
- RTL models
- Synthesis scripts
- Verification models and simulators
- Virtual prototypes

Your RISC-V  
SDK

Software Design Kit

- Compiler
- Assembler
- Linker
- Debugger
- IDE

Profiling of embedded application SW enables processor optimizations



ISA extensions are quickly implemented and analyzed during design space exploration

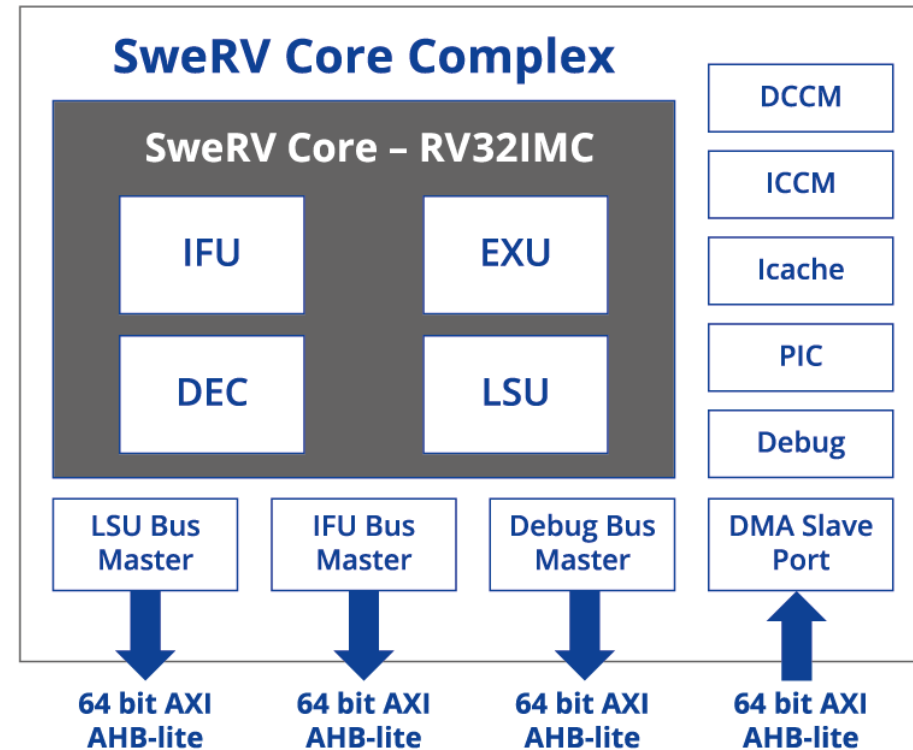
# SweRV Support Package

Getting the Support for Western Digital's SweRV Core(s)

# What is SweRV?

The first RISC-V-based open core developed by Western Digital

## SweRV Core™



# What is SSP?

## SweRV Support Package

- Contains everything needed to deploy a SweRV-based system-on-chip:
  - Useful tools
  - Sample deliverables
  - Technical support
- Available exclusively from Cudasip
- Open-source packaging system

# Why do I need the SSP?

- ✓ Provides **implementation files tied to 3<sup>rd</sup> party tools** that cannot be open-sourced:
  - Verification testbenches
  - Synthesis scripts
  - Integration into virtual platforms
- ✓ Includes access to **professional technical support**

Integrate the SweRV core with confidence  
while avoiding expensive license fees and royalties!

# SSP Contents: Hardware

The latest SweRV core RTL

Implementation support

- Synthesis scripts for Design Compiler and Genus
- Simulation scripts for VCS, Questa, and Incisive

FPGA bitstream

- FPGA target and dev board including sample subsystem
- Support for JTAG debug
- Example software

Benchmark data and documentation

# SSP Contents: Verification

Verification report with coverage data

Reference testbench

- System Verilog assertion monitors
- Verification IP



# SSP Contents: Software

## gcc compiler toolchain

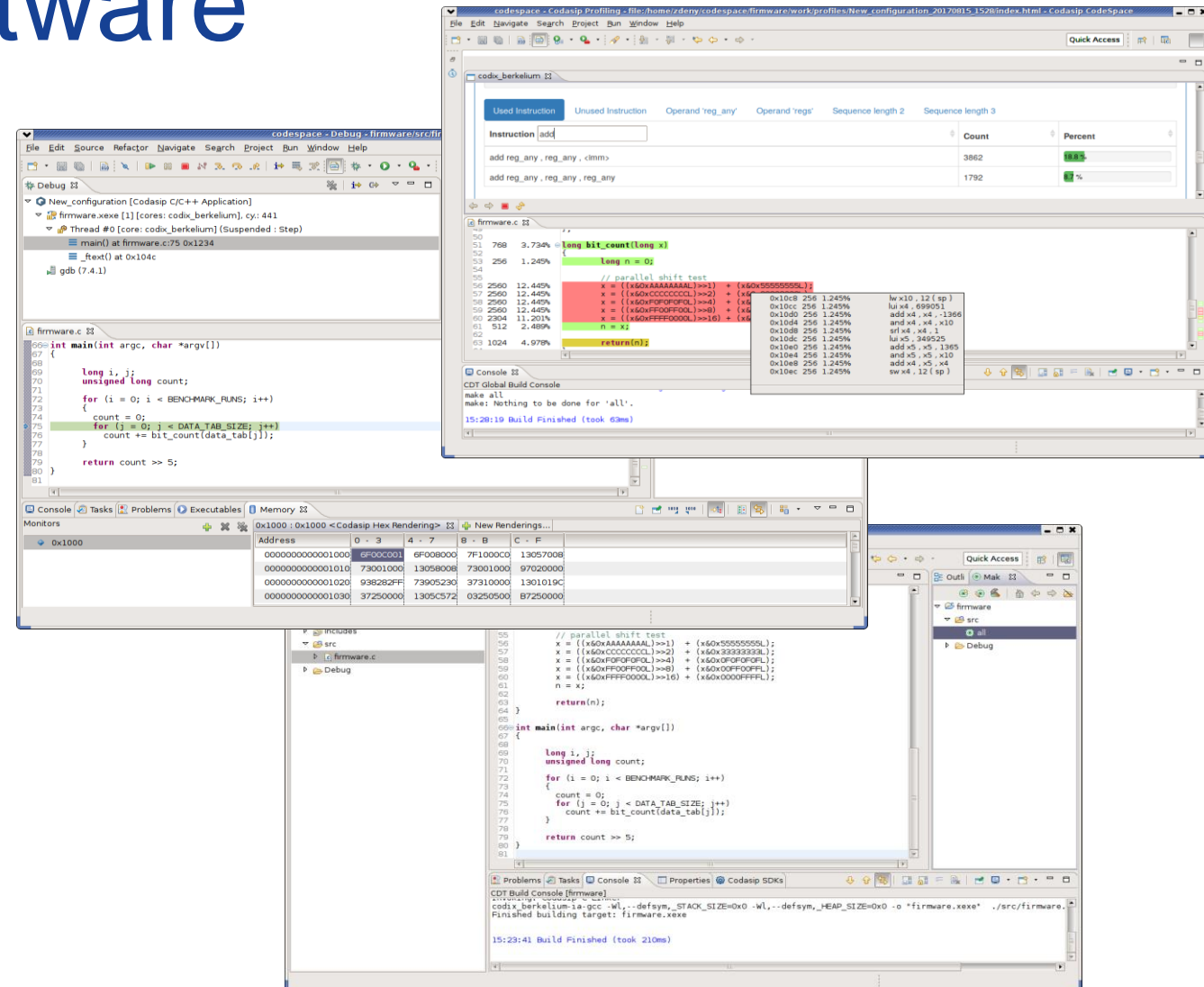
- Newlib
- Binutils
- Gdb

## Eclipse IDE

- Graphical debug
- Profiler

## Abstraction Layer

- Reset and initialization code
- OpenOCD for JTAG connectivity



# 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

## 3. Codasip offers SweRV Support Package

- Contains **everything** needed to deploy Western Digital's **SweRV** core(s)
  - Useful tools
  - Support for 3<sup>rd</sup> party EDA tools
  - Sample deliverables
  - Technical support
- Available exclusively from Codasip

# Thank you!

Questions?



[prikryl@codasip.com](mailto:prikryl@codasip.com)

[www.codasip.com](http://www.codasip.com)

Codasip GmbH