Example of Extending RISC-V for AI/ML Domain Specific Processors

Pascal Gouedo, Damien Le Bars, Olivier Montfort
Dolphin Design

Lee Moore, Aimee Sutton, Larry Lapides
Imperas Software

16 March 2023

© 2023 Imperas Software Ltd.
Agenda

• Problem overview: design of a RISC-V based DSP subsystem IP
• Why RISC-V?
• Challenge 1: RISC-V processor design verification (DV)
• Challenge 2: Software development
• Solutions
• Summary
Agenda

• Problem overview: design of a RISC-V based DSP subsystem IP
  • Dolphin Design
  • Imperas Software
• Why RISC-V?
• Challenge 1: RISC-V processor design verification (DV)
• Challenge 2: Software development
• Solutions
• Summary
Dolphin Design: IP & Design Services

- Dolphin Design develops IP and provides SoC design services
  - Founded 1985
  - Based in Meylan, France
  - Nearly 200 people, 75% engineers

- SPEED: System Platforms for Energy Efficient Designs
  - Power management
  - Audio
  - MCU subsystem
  - Neural processing unit
  - “Panther” DSP: high throughput general purpose DSP, with extremely low power consumption
    - 1st generation built with RI5CY open-source RISC-V cores
    - 2nd generation should improve performance, power consumption, quality
    - 2nd generation design choice: OpenHW Group RISC-V core CV32E40Pv2
### Panther 2nd Generation Design Choice: OpenHW CV32E40Pv2

#### Motivations

- CV32E40Pv1 is an industrial-grade verified core, and has been implemented in SoCs
- CV32E40Pv2 improvements
  - DSP-like capabilities
  - Higher performance
  - Lower power consumption
  - Smaller code size
- CV32E40Pv2 adds
  - Hardware loops
  - Multiply and Accumulate
  - Post-increment load/store
  - 16- and 8-bit SIMD
  - Optional single-precision FPU (RISC-V F extension)
- Can be used as microcontroller, DSP and for neural network applications
OpenHW Group is Building Commercial Quality, Open Source RISC-V Processors

- OpenHW staff and member organizations are focused on processor implementation, verification and software (including tool chains)
- OpenHW Core-V-Verif
  - OpenHW users now on 3rd generation of Core-V-Verif DV flow
  - CV32E40P successfully taped out
  - CV32E40Pv2, CV32E40S, CV32E40X, CV32E20 using this flow today; all expected to complete DV this year
Imperas: Leading the RISC-V Ecosystem with Processor Model Based Solutions

- Imperas: RISC-V processor verification and software simulation solutions, based on instruction accurate Open Virtual Platforms (OVP) processor models
  - Founded 2007
  - Based in Oxford, England
- ImperasDV family of RISC-V processor DV products
  - Reference models
  - Verification IP
  - Standards for verification interfaces
  - Architecture validation tests and specialized test suites
  - Functional coverage
- M*SDK for virtual prototypes
  - Software simulation
  - MultiProcessor Debugger (MPD)
  - Software Verification, Analysis and Profiling (VAP) tools

OVP Processor Models support 15 different Instruction Set Architectures (ISAs)
Imperas RISC-V Customers and Partners

Users
- Marvell Semiconductor
- Nvidia Networking (Mellanox)
- NXP
- Silicon Labs
- Seagate
- Nagravision
- Dolphin Design
- lowRISC (Ibex)
- EM Micro US
- Top 10 semiconductor company with embedded, GPU use cases
- Top-tier systems company (AI application)
- Largest automotive ADAS/AI company
- Startup building accelerator based on multiprocessor RV64
- Japanese government projects “TRASIO” and “RVSPF”
- Numerous universities around the world
- 100+ organizations using free riscvOVPsimPlus

Processor IP Partners
- RISC-V Intl
- Andes
- Cobham Gaisler
- Codasip
- Imagination
- Intel FPGA
- MIPS
- Microchip
- NSITEXE
- OpenHW Group (Imperas is chair of the verification task group)
- SiFive
- Ventana

Tool Partners
- Breker
- Cadence (Palladium integration)
- Google (open source ISG)
- Synopsys
- Valtrix (test generation tools)
Agenda

• Problem overview: design of a RISC-V based DSP subsystem IP
• Why RISC-V?
• Challenge 1: RISC-V processor design verification (DV)
• Challenge 2: Software development
• Solutions
• Summary
RISC-V History

- RISC-V is an open standard instruction set architecture (ISA) that began in 2010 at the University of California, Berkeley
- Unlike most other ISAs, RISC-V is provided under open-source licenses that do not require fees to use
  - This is just the architecture, not the processor implementation
- Unlike other academic designs which are typically optimized only for simplicity of exposition, the designers intended that the RISC-V instruction set be usable for practical computers
- While the ISA is a comprehensive RISC architecture, the RISC-V specification allows for users to add custom features (instructions, CSRs, …)
- The freedom, and not the free, is why RISC-V usage is growing so fast
Freedom Enables Domain Specific Processing

- RISC-V is growing in market segments where x86 (PCs, data centers) and Arm (mobile) architectures are not dominant
  - Small microcontrollers for SoC management, replacing proprietary cores
  - Verticals such as IoT and AI/ML
  - Horizontal markets such as security
- The freedom of the open ISA enables users to develop differentiated domain specific processors
- RISC-V users include traditional semiconductor companies, and embedded systems companies now practicing vertical integration by developing their own SoCs

- RISC-V enables Dolphin Design to develop differentiated DSP subsystem IP (high throughput, low power consumption)
Agenda

• Problem overview: design of a RISC-V based DSP subsystem IP
• Why RISC-V?
• Challenge 1: RISC-V processor design verification (DV)
• Challenge 2: Software development
• Solutions
• Summary
The RISC-V Disconnect

RISC-V
The RISC-V Disconnect

RISC-V Core User
Expect core quality to be the same as Arm

RISC-V Core Developer
Unlikely to have resources needed to be able to develop all the technologies required to perform the same level of verification as Arm
Challenges in RISC-V Processor DV

• Feature selection and design choices require serious consideration due to implications of every decision
  • Experienced processor teams know the costs associated with every feature
  • Every addition dramatically compounds verification complexity
  • Adds schedule, resources, quality costs == big risks

• As of 2021, no off-the-shelf toolkit/products available for DV of processors
  • No EDA vendor has ‘RISC-V CPU DV kit’ product
  • There are in-house proprietary solutions in CPU developers (e.g. Intel, AMD, Arm, …)

• Current SoC cost is 50% for HW DV (with CPUs bought in as proven IP)
• Processor DV is 5x more difficult than SoC DV
• Solutions are generally either 1) post-simulation trace-log file-compare or 2) asynchronous-continuous-compare
1) Post-Simulation Trace Log File Compare (Entry Level DV)

- Process
  - use random generator (ISG) to create tests
  - during simulation of ISS write trace log file
  - during simulation of RTL write trace log file
  - at the end of both runs, run logs through compare program to see differences / failures

- ISS: riscvOVPsimPlus includes Trace and GDB interface
  - Free ISS: https://www.ovpworld.org/riscvOVPsimPlus

- ISG: riscv-dv from Google Cloud / Chips Alliance
  - Free ISG: https://github.com/google/riscv-dv
2) Asynchronous Step-Compare (Highest Quality DV Methodology)

- Design features needing this methodology include OoO and multi-issue pipeline, multi-hart processor, debug mode, interrupts, ...
  - Example SystemVerilog components
    - tracer: Reports instructions for checking and register writebacks
    - step_and_compare: Manages the reference model and checks functionality
    - interrupt_assert: Properties for interrupt coverage/checking
    - debug_assert: Properties for debug coverage/checking
- Typically hard, complex, and expensive to get working
  - Challenge is extracting async info from micro-architecture RTL pipeline

Example flow:

2nd generation CV32E40P OpenHW flow (2H2020) (Imperas model encapsulated in SystemVerilog)
Async Continuous Compare
(Highest Quality DV Methodology)

- Asynchronous events are driven into the DUT
- Tracer informs reference model about async events
- Verification IP handles async event predictive engine, scoreboard, comparison, pass/fail
Agenda

- Problem overview: design of a RISC-V based DSP subsystem IP
- Why RISC-V?
- Challenge 1: RISC-V processor design verification (DV)
- Challenge 2: Software development
- Solutions
- Summary
Software Development Challenges

• Too often software is the critical path to delivering a new SoC or IP product

• Software porting can be a barrier to customer adoption of new SoC or IP

• Second level issues
  • Software quality, safety, security
  • SoC and IP hardware/software architecture tradeoffs
  • Hardware-software co-verification

• Software simulation (virtual platforms), with ability to “shift left” schedule and provide a comprehensive, deterministic development environment, has been broadly adopted to address embedded software development

• Ideally, if building a new processor, want to use the same processor model as the golden reference model for processor DV, and as the instruction accurate model for software development
Agenda

• Problem overview: design of a RISC-V based DSP subsystem IP
• Why RISC-V?
• Challenge 1: RISC-V processor design verification (DV)
• Challenge 2: Software development

• Solutions
  • RISC-V processor DV
  • Software development

• Summary
Agenda

• Problem overview: design of a RISC-V based DSP subsystem IP
• Why RISC-V?
• Challenge 1: RISC-V processor design verification (DV)
• Challenge 2: Software development

• Solutions
  • RISC-V processor DV
  • Software development

• Summary
Asynchronous-Step-Compare for OpenHW CV32E40P (core-v-verif)

Success!

But that was a lot of work … How to improve? In general, and for CV32E40Pv2?
Asynchronous Continuous Compare Using ImperasDV

- Reference model needed for comparison of correct behavior
- Verification IP provides ease of use, saves time and resources
- RVVI standard provides communication between test bench and reference model subsystem
- riscvISACOV: functional coverage modules
- Test suites: riscvISATESTS, directed test suites for difficult extensions

RTL Tracer is the limiting factor for DV quality: determines information flow from RTL to reference model for comparison

- **Async continuous compare methodology, is needed to support features such as interrupts, privilege modes, Debug mode, multi-hart, multi-issue and OoO pipeline, ...**
RISC-V Model Requirements
Not Just for DV; Also for SW Development

• Model the ISA, including all versions of the ratified spec, and stable unratified extensions
• Easily update and configure the model for the next project
• User-extendable for custom instructions, registers, …
• Model actual processor IP, e.g. Andes, Codasip, MIPS, NSITEXE, OpenHW, SiFive, …
• Well-defined test process including coverage metrics
• Interface to other simulators, e.g. SystemVerilog, SystemC, Imperas virtual platform simulators
• Interface to software debug tools, e.g. GDB/Eclipse, Imperas MPD
• Interface to software analysis tools including access to processor internal state, etc.
• Interface to architecture exploration tools including extensibility to timing estimation

• Most RISC-V ISSs can meet one or two of these requirements
• Imperas models and simulators were built to satisfy these requirements, and matured through usage on non-RISC-V ISAs over the last 15+ years
Reference Model Needs to Support the Spec, Enable Custom Instructions

Imperas develops and maintains Base Model
• Base Model implements RISC-V specification in full
• Fully user configurable to select which ISA extensions
• Fully user configurable to select which version of each ISA extension
• Base Model built using Test Driven Development methodology
• Built using public APIs matured over 15 different ISAs

Imperas provides methodology to easily extend base model
• Custom instructions added using same APIs as in Base Model
• Separate source files and no duplication to ensure easy maintenance
• 100+ page user guide/reference manual with many examples
• User extension source can be proprietary (Apache 2.0 open source license)

➢ RISC-V Base Model is used in all Imperas RISC-V processor models
➢ RISC-V Base Model is used by > 150 organizations
Verification IP

- Data prep for functional coverage
- Data movement from SystemVerilog to C verification IP
- Logging data

- Reference model encapsulation
- Includes DUT reference state storage
- Includes synchronization technology
  - Can run sync, async, interrupts, debug, multi-hart
- Predictive engine is key for asynchronous event DV
- Includes comparison technology
  - Comparisons are done on DUT/Reference Model processor events; enables DV of multi-issue and OoO pipeline processors
Open Standard RISC-V Verification Interface: RVVI

- RVVI = RISC-V Verification Interface
  - [https://github.com/riscv-verification/RVVI](https://github.com/riscv-verification/RVVI)
- Work has evolved over 3 years
- Standardize communication between testbench and RISC-V VIP
- Two parts (currently):
  - **RVVI-TRACE**: signal level interface to RISC-V VIP
  - **RVVI-API**: function level interface to RISC-V VIP
RVVI-TRACE Enables DUT Introspection by Verification IP

- Defines information to be extracted by RTL Tracer
- SystemVerilog interface
- Includes functions to handle asynchronous events
  - e.g. interrupts, debug requests

https://github.com/riscv-verification/RVVI/tree/main/RVVI-TRACE
RVVI-API Enables Easy Implementation of Continuous Compare Flow

https://github.com/riscv-verification/RVVI/blob/main/include/host/rvvi/rvvi-api.h
riscvISACOV Is Automatically Generated SystemVerilog Functional Coverage

Building functional coverage is a lot of work:
- 10-40 lines of SystemVerilog for each instruction => minimum of 10K lines of code
- Then need crosses, etc.

- Imperas riscvISACOV provides functional coverage modules for the basic RISC-V instructions
- Imperas tools can automatically generate functional coverage code for custom instructions

Functional coverage results displayed in RTL simulator
Agenda

• Problem overview: design of a RISC-V based DSP subsystem IP
• Why RISC-V?
• Challenge 1: RISC-V processor design verification (DV)
• Challenge 2: Software development

• Solutions
  • RISC-V processor DV
  • Software development

• Summary
Key technologies/differentiators:

- **OVP Fast Processor Models**
  - Largest library of models (>275)
  - Highest quality
- **Simulator engine**
  - Highest performance
  - SlipStreamer API for non-intrusive tools
- **Tools**
  - MPD for platform-centric debug
  - VAP tools for comprehensive software analysis

Software Verification, Analysis & Profiling (VAP) tools:
- Trace (instruction, function, …)
- Profile
- Code coverage
- Memory monitor
- Protocol checker
- Assertion checkers
- Fault injection
- OS aware tools
- Timing estimation
- …

Multiprocessor / Multicore Debugger (MPD)

Eclipse IDE
OVP models are instantiated in SystemC virtual platforms as native SystemC models.

The OVP model calls the simulator; the Imperas simulator uses the SystemC simulator event scheduler:

- Not co-simulation
- No overhead from SystemC wrapper around OVP model

Imperas VAP tools can be used on processors.

GDB or similar debugger using RSP protocol for debug.
Dolphin Design SystemC Virtual Platform

• Using Accellera SystemC simulator
• Virtual platform echoes the actual hardware
  • Multiple RISC-V processor models
  • Memory
  • Models of behavioral components
• Executes production binaries compiled for the target architecture
• Used for software debug, analysis and optimization
Agenda

• Introduction to Imperas
• Why RISC-V?
• RISC-V processor design verification (DV) issues
• 5 levels of RISC-V DV methodology
• Key technologies: reference models, verification IP
• Summary
Summary

• Usage of the RISC-V ISA is growing

• Key technologies and methodologies are needed for continued usage by the early adopters, and new usage by mainstream semiconductor companies
  • RISC-V processor DV – asynchronous-continuous-compare – has been successfully used
  • Virtual platforms – software simulation – accelerate software development

• The same RISC-V processor model can and should be used for both DV and software simulation

• RISC-V can be effectively used for AI/ML architectures
Thank you!

LarryL@imperas.com