SYNOPSYS®

WHITE PAPER

Understanding UVM Coverage for RISC-V Processor Designs

Authors

Prabha Krishnaswami
Senior Staff Applications Engineer, Synopsys

Rohit Narkar
Director, Application Engineering, Synopsys

Amit Goldie
Principal Applications Engineer, Synopsys

Bipul Talukdar
Senior Director, Application Engineering, Bluespec

Introduction

Design verification for modern chips is a difficult and daunting problem. The sheer size and complexity of these devices scale the verification effort faster than the design effort. In fact, functional verification is usually the most resource- and time-intensive phase of a chip development project. Many innovative technologies have been brought to bear on this problem: static and formal verification, verification reuse and verification IP (VIP), constrained random stimulus generation, advanced coverage metrics, coverage driven verification plans, and more. Only through effective application of such techniques do development teams have any chance of taping out functionally correct designs on time and on budget.

The situation grows dramatically more complicated for system-on-chip (SoC) designs with embedded processors. Historically, processor companies have led innovation in verification because additional layers of complexity are involved. Modern processor designs have a high degree of parallelism, overlapping operations, complex cache structures, and many other elements that make verification much harder. In addition, the very nature of processors and SoCs means that both hardware and software work together to provide the required functionality. This means that thorough hardware verification is not possible without taking software into account.

When an SoC team licenses a proprietary processor core from an intellectual property (IP) provider, the engineers place great faith in the process performed by the provider's verification team. The end users typically rely on the petacycles of instructions verified during processor development and focus only on integration testing at their end. This is changing with the recent proliferation of RISC-V based designs. The CPU core now requires thorough verification by the SoC team. They need to budget additional resources and time, while considering the impact on their project schedule.

RISC-V presents special challenges because its specification is designed to provide a configurable and customizable solution for general purpose processors. There are many options and variations defined in the instruction set architecture (ISA) specification, and users are permitted to extend the ISA even further by adding custom instructions. Therefore, any RISC-V verification solution must be flexible enough to accommodate customizations. Constrained random test suites must be able to scale as per the chosen extensions. Similarly, the definition and application of coverage metrics must scale to accommodate both standard variations and custom instructions. A complete RISC-V verification environment requires a coverage model built accordingly.

Attempting to achieve complete RISC-V verification requires multiple methodologies employing a wide range of relevant tools, including:

- Coverage driven simulation based on UVM constrained random methods and compliant with the Universal Verification Methodology (UVM) standard
- Static and formal property verification
- Equivalence checking
- Emulation and FPGA based verification
- Low power verification depending on the intended design application
- Verification using the Portable Stimulus Standard (PSS)

This white paper focuses on the first technique in this list. It explains the basics of UVM functional coverage for RISC-V cores using the Google RISCV-DV open-source project, Synopsys verification solutions, and RISC-V processor cores from Bluespec.

Background on RISC-V

Unlike traditional proprietary processor architectures, RISC-V is an open ISA originally developed at the University of California, Berkeley. It is considered the fifth generation of processors built on the concept of the reduced instruction set computer (RISC). Due to its openness and its technical merits, it has become very popular in recent years. The standard is now managed by RISC-V International, which has more than three thousand members and which reported that more than ten billion chips containing RISC-V cores had shipped by the end of 2022. Many implementations of RISC-V are available, both as open-source cores and as commercial IP products.

RISC-V is a load-store ISA, with all arithmetic and logical instructions operating on general purpose registers and additional instructions to move data between registers and memory. The base instruction set defines a set of integer operations on thirty-two registers and a 32-bit memory address space. There are options for 64-bit and 128-bit addresses as well as a reduced set of sixteen registers for embedded designs. Because it was designed to be applicable for a wide range of applications, most ISA features are optional. For example, thirty-two floating-point registers are added if floating-point instructions are included. The optional features are defined in a set of extensions including:

- M: Multiplication and division (integer)
- A: Atomic operations
- F: Floating point (32-bit)
- D: Double floating point (64-bit)
- Q: Quad floating point (128-bit)
- C: Compressed instructions
- B: Bit manipulation
- H: Hypervisor
- S: Supervisor

Background on UVM Coverage

UVM provides a foundation for building a modular, reusable testbench that is easy to modify for various designs. It defines the testbench architecture, test sequences, and verification environment, which configures the design, configures the testbench, and produces test stimulus for the design. The constrained random approach makes it possible to test a lot of input situations quickly and effectively. Constraints are established on the input signals to guarantee the validity and realism of the generated stimulus. The random stimulus generator takes advantage of these restrictions to provide input values that adhere to the constraints.

Coverage metrics are essential for constrained random verification approaches such as UVM. With hand-written directed tests, each test is explicitly developed to verify a particular part of the functionality, so there is a clear correlation between tests and parts of the design. With automatically generated tests, there is no such direct connection. Coverage shows which parts of the design are stimulated by which tests, and which parts have not been exercised by any test. Verification engineers often tweak and bias constraints to focus tests on unverified functionality. Only when all coverage is hit—or at least a high percentage as defined by project targets—is the design considered well enough verified to consider taping out.

Structural code coverage (line, block, condition, expression, etc.) is an important part of the process. If any part of the register-transfer-level (RTL) design code has not been exercised, it has not been verified, and bugs may be hiding there. However, code coverage must be supplemented by functional coverage that reflects the intended functionality of the design. SystemVerilog, the basis for UVM, includes coverpoints, covergroups, and cross coverage as constructs for specifying coverage. This resulting powerful coverage model provides a faster, scalable approach to verify a RISC-V design quickly and reliably.

Background on RISCV-DV

As noted earlier, verifying processor and SoC designs requires a combination of hardware and software. The software aspect includes programs running on the embedded processors, and often these programs are automatically generated sequences of processor instructions designed to stress-test the design. In the case of the RISC-V ISA, there is a widely adopted solution. RISCV-DV is an instruction generator for RISC-V processor verification developed by Google and available as open-source. It is designed for use in UVM verification environments, generating handshakes between the generated code and the testbench, as well as an instruction generation coverage model. It supports the base ISA and many of the extensions.

Bluespec RISC-V Example Core

Bluespec provides a wide variety of soft (RTL) cores compliant with the RISC-V ISA. The MCU embedded processor soft core used in this white paper is a low footprint multi-cycle RISC-V controller optimized for minimal hardware usage. It implements the RV32I instruction set, consisting of the base 32-bit (set I) RISC-V instructions. Other cores implementing higher extensions of the ISA are available from Bluespec. Figure 1 shows a block diagram of the MCU core.

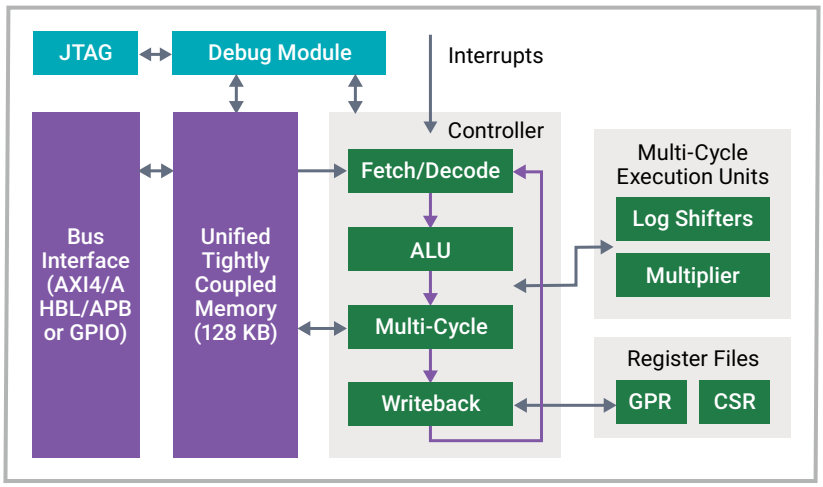

Figure 1: Block diagram of the Bluespec MCU core

The MCU is easy to integrate and operate. It runs on a single edged-triggered clock and a single reset pin. Instructions and data are stored in a unified Tightly Coupled Memory (TCM). The core supports a maximum of 128 kilobytes of TCM and provides an AXI4 Bus Interface and a Debug Module. This version of the MCU does include a multiplier although, as shown in Figure 1, it is an option if higher RISC-V extensions are selected. The controller fetches and passes instructions to the execution unit, where the ALU executes basic operations in one cycle. Loads and stores to the TCM take an extra cycle. More complex operations such as shift and multiply use additional cycles. The controller updates the RV32I architectural state stored in the PC (program counter), GPRs (general purpose registers), and CSRs (control-status registers) after executing each instruction.

MCU Verification Methodology

The core described in the previous section was verified using a methodology that could be applied to any RISC-V processor design. The basic functionality of the core is verified using bring-up tests, simple C programs that are bare metal compiled to generate instruction binaries. These binaries are then loaded into a simple memory model that drives the stimulus for the core.

To stress-test the ISA, the open-source RISCV-DV is used to generate constrained random stimulus. It comprises two parts: a UVM testbench generator that produces ISA-based random stimulus and an assembly language generator that creates assembly files corresponding to the constrained random stimulus generated by the first step. The RISCV-DV environment provides instruction level, sequence level, and program level randomization, and thus helps in obtaining good stimulus coverage for the ISA test suite. The assembly files are converted into binary executables of random valid instructions that can run on any core supporting the RISC-V ISA.

Checking the results of running the tests on the RTL design requires some sort of golden reference. This is obtained by running the same generated tests on the Spike open-source RISC-V instruction set simulator (ISS). The results from the golden source are compared with the results of the simulation on the Bluespec core. In both the bring-up and random tests, Synopsys VCS® is used as the simulator and the Synopsys Verdi® Automated Debug System is used to view waveforms and coverage results.

The current MCU simulation environment is spread across two testbenches:

- MCU testbench: top level testbench where the MCU RTL is simulated using VCS. The stimulus for this testbench can be created either by bring-up tests or by constrained random stimulus. Code coverage is enabled in this testbench.
- RISCV-DV testbench: UVM testbench used to generate constrained random RISC-V instructions. Functional coverage is enabled in this testbench. The stimulus generated here is eventually simulated in the MCU testbench. Spike ISS is also run in this environment to create the golden reference for the MCU testbench.

MCU Coverage Measurement

As noted earlier, coverage metrics ratify the completeness and success of a verification strategy. They measure how well the verification plan has been executed and whether the key features of the design have been exercised. Both functional and code coverage are used in the Bluespec MPU verification environment. The previously described RISCV-DV generates a UVM testbench with a readily available functional coverage model for the ISA test suite. Since this model is tightly coupled to the random generator engine that stress-tests the RISC-V core, it is used for functional coverage of the MPU core. Figure 2 shows a strategy that merges functional coverage generated in the RISCV-DV environment with code coverage generated in the MCU testbench. The merge can be performed with the Synopsys VCS Unified Report Generator (URG) or with Synopsys Verdi Coverage.

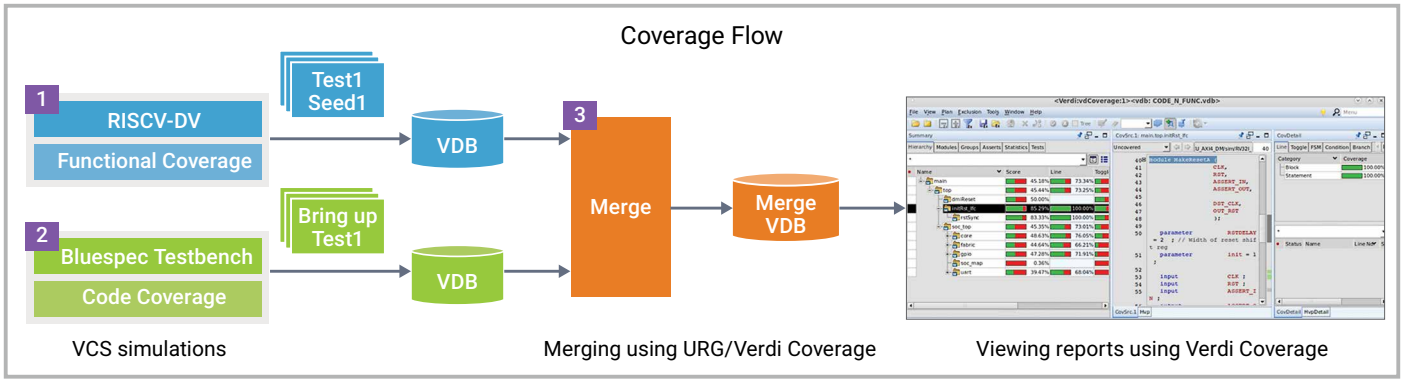

Figure 2: Coverage flow for the MPU core

Functional coverage definitions for RISC-V implementations must cover compliance to the ISA specification, ideally spanning all possible instruction combinations. It turns out that this ideal is simply not achievable if one considers possible sources, destinations, and data values. Thus, it is necessary to define realistic 100% RISC-V ISA coverage.

100% RISC-V ISA coverage

For each RV32I instruction, it is mathematically possible to enumerate coverage points that cover all possible modes, register addresses, register contents, and immediate values. There are thirty-two possibilities for each register referenced in the instruction, and covering all these values is easy. For an instruction with two source registers and one destination register, there are 32x32x32 (2^{15}) combinations, also within the reasonable realm for simulation. However, each 32-bit source register has one of 2^{32} possible data values, exploding the number of coverage points beyond all practical limits. Figure 3 shows example results for addition and load instructions.

Instruction	Possible Register Values	Possible Data Values	Possible Immediate Values	Mathematically Possible Coverage	Practical Coverage	RISCV-DV Coverage	Improved Coverage
Add rd,rs1,rs2	$2^5x2^5x2^5$	$2^{32}x2^{32}$		2^{79}	2^{15}	?	?
LW rd,rs1,imm	2^5x2^5	2^{32}	2^{12}	2^{54}	2^{22}	?	?

Figure 3: Examples of RISC-V ISA coverage

One way to define a practical number of coverage points is to consider all possible instruction modes, register addresses, and immediate values, but not all possible data values for register contents. As Figure 3 demonstrates, this reduces coverage for the addition instruction down to the 2^{15} number practical for simulation. This table may be filled in for all RISC-V base instructions and extensions in the core being verified.

Defining 100% RISC-V ISA coverage and sharpening the definition is an iterative process of upgrading the definition and re-running the test suite to achieve the coverage goals. Building such a methodology is currently in progress as a collaboration between Synopsys and Bluespec.

Synopsys Verdi Coverage and Debug Support

As shown in Figure 2, the code coverage results from the MCU testbench and the functional coverage results from the RISCV-DV testbench are combined by merging the verification database (VDB) files generated by Synopsys VCS. The merged VDB is loaded into Synopsys Verdi, where all coverage results can be viewed and analyzed. Figure 5 shows an example of viewing code coverage in the Synopsys Verdi Coverage graphical user interface (GUI) and Figure 6 an example for functional coverage.

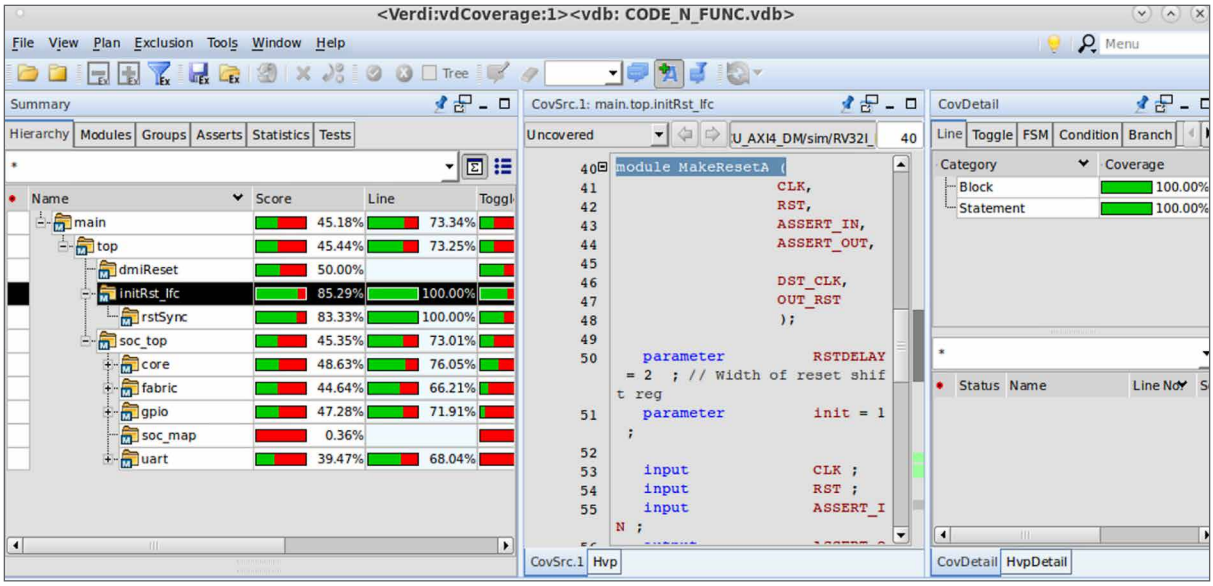

Figure 5: Synopsys Verdi used to view code coverage using the merged VDB

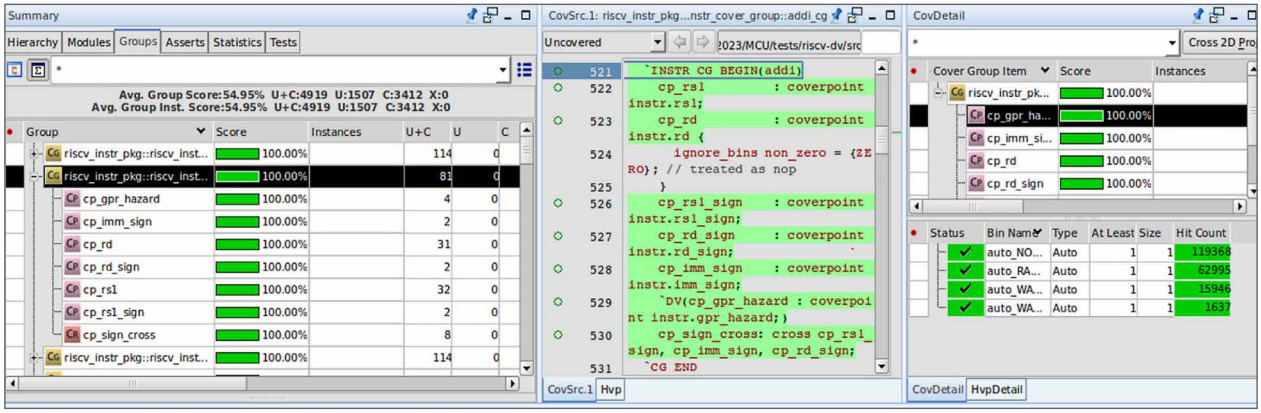

Figure 6: Synopsys Verdi used to view functional coverage using the merged VDB

The Synopsys Verdi Automated Debug System, when combined with the Synopsys Verdi Hardware Software Debug (HW/SW Debug) Solution, provides a method for debugging embedded software. This approach integrates debug of the RTL core design at the hardware level and debug of the embedded code at the assembly or C level. The C/assembly code, C variables, and stack are visible, with the hardware and software debugging synchronized in time. Multiple cores in an SoC design can be debugged simultaneously.

This solution proved useful during the Bluespec MCU verification by providing more visibility into the generated RISC-V instructions. It was especially helpful while debugging constrained random sequences, where user visibility into the software side may not be that high. Figure 7 shows the MCU RTL design, and the executing RISC-V code opened in the Synopsys Verdi HW-SW Debug GUI. The cursor locations demonstrate how simulation time is synchronized between the two sides.

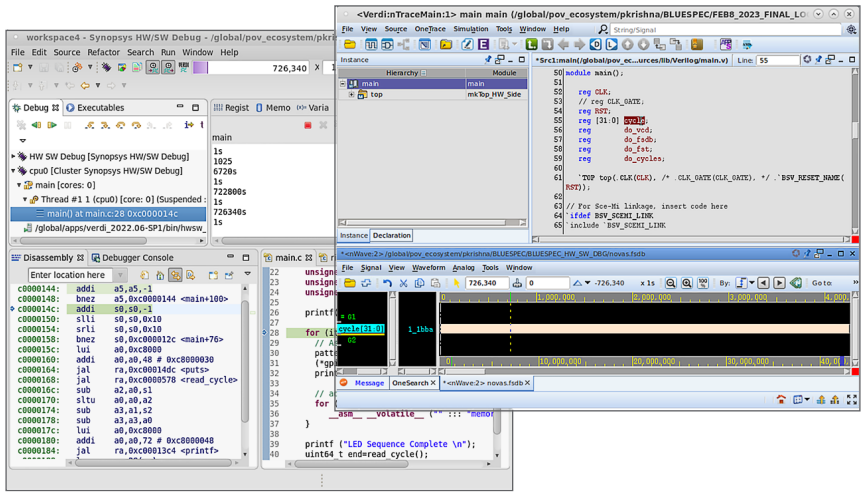

Figure 7: Synopsys Verdi used for combined hardware-software debug

Verification Reference Cookbook

Synopsys also provides a Verification Reference Cookbook for Bluespec RISC-V processors. This features Bluespec's RV32I.MCU. AXI4.DM processor core and includes the recommended verification methodology to be used with Synopsys tools. Please reach out to any Synopsys contact for more details.

Conclusions

This white paper has presented a methodology for RISC-V ISA coverage, using a commercially available Bluespec MCU core as an example. Advantages of this approach include:

- Seamless integration of RISCV-DV and the readily available ISA functional coverage model
  - Leveraging the robust RISCV-DV setup
  - Providing the verification team a mechanism to gauge the impact of the constrained random tests on coverage
  - Addressing the initial phase of testbench development when custom functional coverage is not yet defined
- Ability to merge functional and code coverage, using Synopsys VCS and Synopsys Verdi to quickly identify coverage holes

A limitation of the current approach is having two testbenches, requiring multiple simulations and adding some maintenance overhead. Future work will likely include the integration of functional and code coverage into a single testbench.

The combination of RISCV-DV and Synopsys verification tools provides a powerful and flexible solution to RISC-V verification and coverage. Its value and ease of use has been demonstrated on the Bluespec MCU core, and the methodology developed can easily be extended for use in verifying any RISC-V processor design. Given the tremendous adoption of this ISA in recent years, having such a robust solution readily at hand is a benefit for the entire chip industry.

05/16/23.CS1103084633-RISV-V-WP.

Pub: May 2023