TECHNICAL **BRIEF**

cadence®

# Improving Aerospace and Defense Program Confidence with Commercial Best Practices from the Cadence Verification Suite

Getting the highest ROI within your running programs

## Introduction

As an aerospace or defense program manager, you have your work cut out for you—with expert planning bounded by fixed timetables and budgets, the execution of a defense project is well controlled. Experience from previous programs, carefully developed specifications, and anticipated challenges are combined to define the project constraints—but that assumes everything runs as planned. Even the "best-laid plans of mice and men" are subject to the unexpected, and when something unforeseen derails the carefully laid timetables of a project, you still must be ready to deliver the program, or risk having to go back for more budget.

Cadence can help. The tools you're already using have a variety of features that are easy to enable and can drastically improve the speed at which you finish verification as measured by complete coverage. If your verification simply takes less time, that's more resources you have available for the setbacks that any program can hit; it's time that "buffers" your program against overrun. Commercial companies have deployed these best practices to buffer their development programs and they're well-suited in the aerospace and defense context, too.

These features exist across the Cadence® Verification Suite, and this paper will highlight a few of the highest ROI ones you can efficiently put into practice. The Cadence Xcelium™ Parallel Logic Simulator has three features that you can get good use from—the new save and restart functionality, dynamic test reload (DTR), and parallel and incremental elaboration. For managing the verification process, there's the Cadence Interconnect Workbench and Cadence vManager™ Metric-Driven Signoff Platform's runner/triage and planning capabilities. For best practices in formal analysis and verification, there are the Cadence JasperGold® Coverage Unreachability (UNR) and Superlint Apps, available as a part of the JasperGold Formal Verification Suite. These features combine to enable you to pursue and reach 100% coverage faster than ever before, creating that buffer program managers want to assure on-time, on-budget, and on-spec delivery.

## Xcelium Simulation

The Xcelium Parallel Logic Simulator offers a wide variety of ways to increase program confidence. The flexibility of the multi-purpose Xcelium simulator comes into play here—no matter what you're doing, there's probably a setting or a feature you can enable to get better results than what you have. For all projects, the Xcelium simulator's save and restart functionality can vastly improve turnaround time by slashing the amount of time wasted on re-running long simulations. The dynamic test reload feature can provide productivity improvements by removing the need to re-run the design under test (DUT) initialization sequences between test runs. The parallel and incremental elaboration feature allows you to avoid re-elaborating your testbench between test cases, dramatically speeding up turnaround time, especially for large designs.

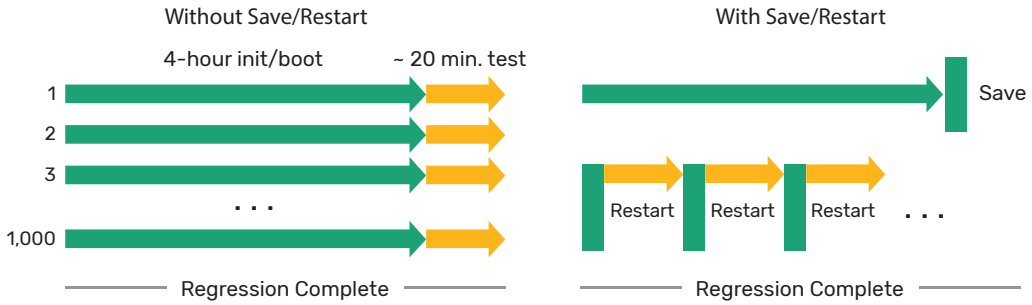


*Figure 1: A comparison between the old and new save and restart features.*

## Save and Restart

The new save and restart functionality in the Xcelium simulator is a vast improvement over the older approaches. You can begin using the new save and restart functionality with a simple run-time switch, -process_save, which is usable with xrun, xmelab, or xmsim commands. Once this is done, a snapshot with save and restart functionality is created.

There are two kinds of restarts that can be done with this system: cold and warm. A cold restart shuts the simulator down completely before restarting, so you can change the simulator invocation settings, the name of snapshot, randomization seeds, and so on. A warm restart doesn't completely reinvoke the simulator—the current simulation run will go back to the restart time. You will not be able to change any invocation settings; however it is still possible to reseed (change the random seeds of) the design. You can perform a warm restart by simply using the Tcl restart command—a warm restart will be done automatically. Alternatively, a warm restart can be done from the Xcelium command line, using `xcelium>restart<snapshot _ name>`.

This video on the process-based Save and Restart feature in the Xcelium simulator demonstrates usage of this feature in cold as well as warm restart in simulations (login required)

## Dynamic Test Reload

Dynamic test reload (DTR) is another new feature in the Xcelium simulator that can improve your ROI. DTR allows you to add a new SystemVerilog package to an existing simulation snapshot, such as one produced through a save, without requiring a full build of the design and run up to the interesting simulation time. It is most commonly used to replace UVM sequences without having to re-elaborate and restart the simulation, greatly speeding the time to develop and debug testbenches. It can also be used to create an "initial snapshot" of the Interconnect Workbench environment post-DUT initialization, removing the need to re-do that time-costly process every time you want to try some new tests to verify your interconnect. This requires some simple modification of your Interconnect Workbench environment.

For specific information on how to modify your Interconnect Workbench environment to enable DTR, see the "Creating Dynamic Tests with System Verilog" resource webpage (login required).

## Parallel and Incremental Elaboration

The Xcelium simulator's parallel and incremental elaboration feature can even further improve ROI by shortening the time it takes to re-elaborate a design so that you can complete your debug iterations faster. To do this, a testcase is split into two parts: one or more primary snapshot(s), which contains code that is stable and doesn't need to be changed, and an incremental snapshot, which contains the code that changes often. In simpler terms, the DUT is captured in the primary snapshot, while the testbench is in the incremental snapshot. The more code that's compiled in the primary snapshot, the more time you will save on re-elaboration.

Parallel and incremental elaboration allows large designs to share the elaboration task with the verification environment. For a team developing tests, new design blocks, or regression environments executing a large set of parallel tests, parallel and incremental elaboration allows you to build the most time-costly part of the design up-front and only once. The Xcelium simulator has the tools to take advantage of this feature—the simplest way being to use the single-step xrun flow, which keeps the primary snapshot-building and final simulation snapshot-building in a single step. It can also be used with a multi-step xrun/xmelab flow to allow for the sharing of primary snapshots between users. If you use parallel and incremental elaboration in this way, be advised that you should use the cds.lib/hdl.var library mechanism along with System Verilog or VHDL configurations to manage the locations of your primary snapshots and to choose which snapshots will be used during elaboration.

To learn more about parallel and incremental elaboration by visiting the Parallel Partitioning section of the Multi-Snapshot Incremental Elaboration manual (login required).

## Interconnect Workbench

The Cadence Interconnect Workbench is a tool you can use to automate your interconnect testing. Using the IP-XACT or CSV description of an interconnect system's RTL, the Interconnect Workbench can create either a UVM e or SystemVerilog testbench with all the necessary verification IP and the scoreboard. An Interconnect Workbench-generated testbench supports component, subsystem, and full SoC-level interconnects. The testbench automation capability alone can shave days off your schedule, but the Interconnect Workbench can also generate a few sets of tests to give you a solid starting point for verification and performance analysis, thus providing a significant productivity boost.

The Interconnect Workbench also includes a performance analyzer (IPA) tool (Figure 2), which produces metrics and graphics to help you easily understand how your tests are performing against a plan.

The Interconnect Workbench's testbench automation capabilities also work on the Cadence Palladium® series of emulation and verification platforms. Unlike the simulation flows for Interconnect Workbench, the Palladium system's Interconnect Validator (IVD) runs as a separate step and performs its checks outside the Palladium box, then loads its logs into the IPA afterward.

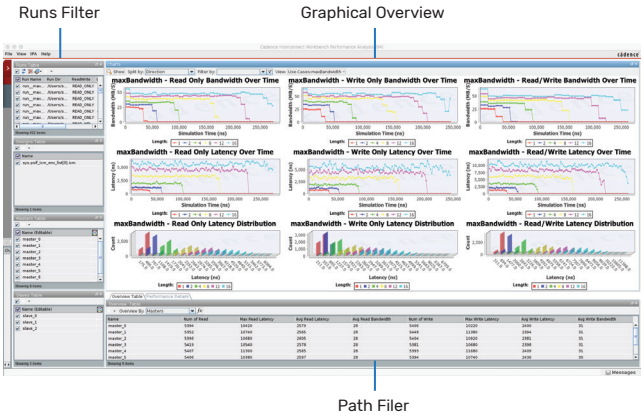


*Figure 2: The Performance Analyzer GUI*

For more information on how the Cadence Interconnect Workbench can be set up and used on both simulation and emulation flows, see the "Interconnect Workbench User Guide" (login required).

## JasperGold Formal Verification

The JasperGold Formal Verification Platform has several features (apps) that should be used to get the most out of your time. Among these, the JasperGold Coverage Unreachability (UNR) App and Superlint App are easily applicable to all use cases, are trivial to enable and use, and can provide significant improvements in and debug turnaround time, respectively.

### JasperGold UNR App

At a certain point in the verification process, adding tests to your test suite doesn't seem to generate any further useful coverage. The JasperGold UNR App addresses this problem by working with your test coverage database to determine if the code that has not yet been tested has any route to be accessed. By determining coverage for >99% of the FPGA or SoC, the JasperGold UNR App can help you avoid countless hours running around in circles looking for that golden test that will cover the uncoverable.

Additionally, the JasperGold UNR App is extremely easy to enable. It's an automated flow, so it can be used even by engineers with little formal knowledge, and it works independently of a testbench methodology. All you need to do is use `xrun -unr -jg -jgsynthesis -R -covdb -coverage all`, and the JasperGold UNR App will output a database showing where coverage holes are. As shown in Figure 3, the JasperGold UNR App is best run after 50-80% coverage (design-dependent) is reached to reduce the time needed for the analysis.

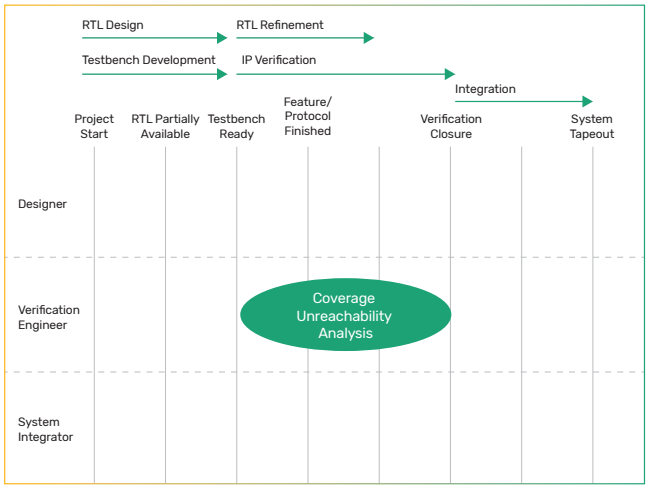


*Figure 3: When should you use the JasperGold UNR App?*

For further information, check the "JasperGold Coverage Unreachability App User Guide" (login required).

### JasperGold Superlint App

The JasperGold platform's linting application, the JasperGold Superlint App, combines RTL syntax-error finding and structural linting with automatic formal behavior analysis, finds coding style violations, sources of bugs, circuit structure violations, and potential sources of bugs. Reports generated by the Superlint App can prove quite useful in the debugging process and require no

testbench creation.

Starting the Superlint App is simple. First execute this command: `%jg -superlint smc.tcl`, then make sure any RTL files you need for compilation are in the rtl directory. This will start the Superlint GUI. From the GUI, you can select any checks you want to run, configure any options, and run the software.

For more information, you can download the "Introduction to JasperGold Superlint App" Rapid Adoption Kit (RAK) (login required)—this will teach you in further detail how to start using the Superlint App.

## vManager Platform

The vManager Metric-Driven Signoff Platform is a powerful management tool for your verification flow with the ability to vastly improve your verification experience, but the idea of adopting a comprehensive metric-driven verification methodology may be daunting to users constrained by an existing program. The vManager platform's ability to track verification progress, plan flows, and provide pertinent metrics has been refined over more than twelve years of program experience. Avoiding the features the vManager platform has to offer out of a concern that it's an all-or-nothing adoption means leaving a lot of useful tools on the table.

The vManager platform is designed to drive verification improvements through coverage and metrics (Figure 4). It can automate test generation with directed randomization and collect metrics on what verification has been done for future use. These metrics are used to further generate tests, easing the test-writing time bottleneck.

These improvements start with a vPlan, a set of coverage and analysis goals that you can outline in the vManager tool, allowing you to coherently set up and plan what you want your tests to target and to directly measure how those tests are doing by having a set of guidelines to compare to. The vPlan can start out as simple as making sure that all the tests that need to run have run successfully in a given regression.

The vManager platform is not difficult to start using with simulation. Cadence can provide vManager Deployment kits to ease the setup. Since the vManager platform is a part of the Cadence Verification Suite, no changes to existing tools are required—all of your Cadence tools will work with the vManager platform right out of the box. Using the vManager platform with multiple verification engines and comprehensive vPlans requires a bit more bring-up time and may be better suited to new programs but applying it in this way can vastly reduce the overall program time for verification and improve both quality and predictability.

For more information, see the vManager User Guide (login required).

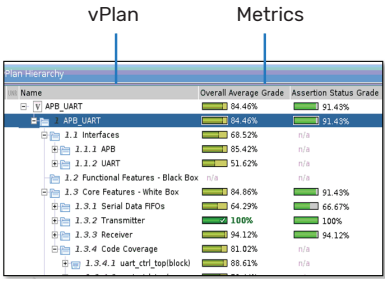


*Figure 4: vManager metrics*

## Confidence in 100% Coverage

Each verification feature in this paper will enable you to execute more verification, but what you really need is a measurement system to translate "more verification" to "increased program confidence". This can be done through coverage measurement, but a program needs a coverage goal and "verification contract" to measure the improvement in confidence.

Five coverage goals are identified in Figure 5 and ordered from more subjective to more objective. "Executed Code" is a more subjective goal because a program can achieve 100% code coverage with some failing tests. "Successfully Tested" is more objective than "Executed Code" because the goal is only achieved with 100% coverage and 100% successfully passing tests. Note that, in both cases, the completion target is 100%. This is possible because the program and customer start with a "verification contract" that can be clearly measured. "Run all the code (Executed Code) and assure that all tests are passing (Successfully Tested)" is an example of a verification contract. The vManager platform can then gather metrics, including waivers, toward 100% completion of these targets. Considering the "Annotated vPlan" goal, the agreement focuses on functional requirements and uses functional coverage, code coverage, test success, and other metrics to measure 100% completion of the targets. You can also use multiple Cadence engines, including the Xcelium, JasperGold, and Palladium platforms, and the Protium™ Prototyping Platform, to generate these metrics.

Regardless of where the program is focused on the coverage goals, the goal must be 100% achieved. As the program progresses, questions regarding progress toward that goal become increasingly urgent as the "verification complete" milestone approaches. By combining the Xcelium, JasperGold, and Interconnect Workbench features with the metrics tracking/reporting capability of the vManager platform, your programs can increase the amount of verification running within the allocated resources of the program and/or prioritize the verification with waivers applied to lower priority items with the

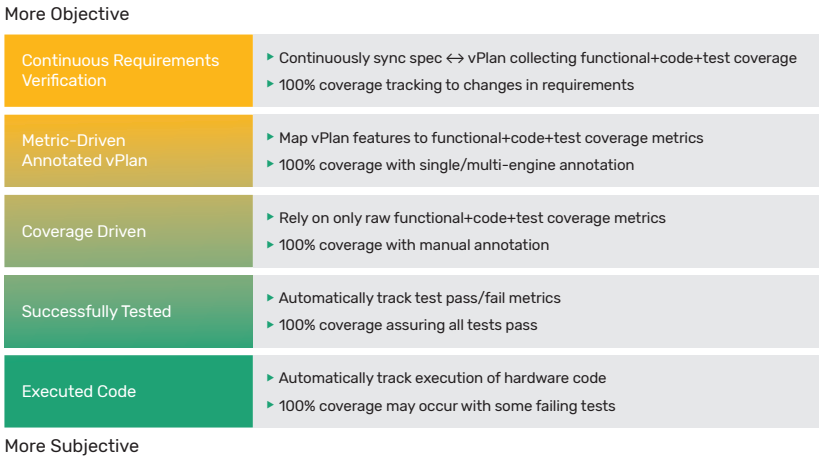


*Figure 5: Coverage goals and definitions*

vManager platform continuously measuring progress and increasing confidence in achieving 100% coverage at the "verification complete" milestone.

## Conclusion

Across Cadence's verification suite of tools, there are ways to improve program confidence by enabling you to make the most of your current tools or to learn new tools with a short learning curve, and using them effectively can bring huge improvements to your quality and verification signoff with minimal bring-up time. While this is not an all-inclusive list of everything you can do to speed up or improve the quality of your verification, this is a selection of options proven in commercial best practices that are the easiest to begin using.

In addition, the Cadence Verification Suite offers a more comprehensive array of of tools and technologies including the Palladium platform for hardware/software emulation, the Protium platform for FPGA prototyping and software verification, and the Perspec™ System Verifier for portible stimulus throughout the verification flow including the manufactured system.

With the options enabled and features in use as detailed in this paper, you can improve schedule confidence within running programs as well as improve your associated budget and personnel constraints.

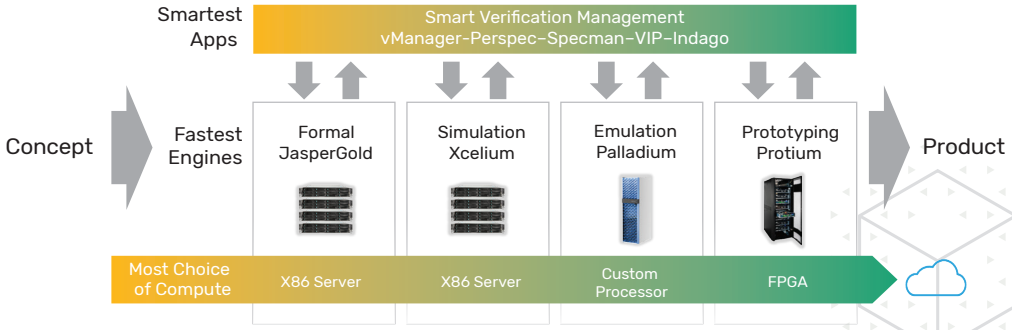


*Figure 6: The Cadence Verification Suite*

cadence®

Cadence is a pivotal leader in electronic design and computational expertise, using its Intelligent System Design strategy to turn design concepts into reality. Cadence customers are the world's most creative and innovative companies, delivering extraordinary electronic products from chips to boards to systems for the most dynamic market applications. **www.cadence.com**