Improving Simulation Throughput Using the Xcelium Parallel Logic Simulator

Tyler Sherer, Technical Writer, Cadence

Simulation is an old process, but improvements in the field happen constantly. The Cadence® Xcelium™ Parallel Logic Simulator is Cadence's newest entry in a long line of these improvements. Featuring multi-core simulation features, expanded single-core capabilities, reworked save and restart functionality, parallel and incremental build, powerful and comprehensive debug tools and more, the Xcelium simulator is bound to suit any user's requirements. This paper details each of these items and discusses both what is new in the Xcelium Parallel Logic Simulator over the old Cadence Incisive® Enterprise Simulator and how these new improvements can be used to improve a user's simulation experience.

Contents

Introduction

Simulators have been around for a long time. First, there were interpreters in the '80s and '90s, and despite being relatively slow, they were a big step up from fabricating the design and hoping it worked. However, as designs continued to increase in size, the interpreters could not keep up with simulation needs, and innovation was required for simulators to keep pace with new technology. This gave rise to the compiled code simulators in the mid-'90s, which worked well enough for most people's simulation needs. However, designs continued their non-linear growth, and yet again, innovation in simulator technology was needed to keep up.

The Xcelium Parallel Simulator is that innovation—the first simulator optimized for multi-core processing.

At a glance, Table 1 shows the kinds of performance gains one can expect for a variety of basic use cases when using the Xcelium simulator and its new features.

	Customer	Description	Speed-up
Single Core	Mobile	Average on 33 UVM RTL tests	1.5X
	Mobile	Low Power SoC	1.3X
	Mobile	Incremental Build + Cloning for Replicated Modules	10.5X
	Memory	MultiCore Life	1.3 – 1.9X
	Automotive	Build Using Cloning for Replicated Modules	5X
	Automotive	Parallel Build	2 – 2.5X

	Customer	Description	Speed-up
Multi-Core	Mobile	RTL Design	2.6X
	Networking	Gate-level Zero Delay ATPG	>10X
	Automotive IP	Gate-level Zero Delay BIST	8X

Table 1: Xcelium simulator performance gains

This whitepaper aims to show some of the new features the Xcelium Parallel Simulator brings to the table as the first multi-core optimized simulator. The Xcelium simulator's state-of-the-art multi-core engine allows for speed improvements through unprecedented design acceleration previously considered impossible. Updated save/restart and incremental build features help ensure that less time is lost to repetitive activity that consumes farm resources without contributing to verification closure. X-propagation technology gives the user a way to catch X optimism bugs without having to spend resources on expensive gate-level simulations. Core engine updates improve both single- and multi-core performance. Couple these features with expanded capabilities for mixed-signal and low power systems as well as metric-driven verification capabilities, and you have a simulator fully prepared for today's designs.

All these features come together to form a cohesive simulating experience.

Single-Core Engine Improvements

The Xcelium simulator's single-core engine has gotten several improvements since the Incisive Enterprise Simulator. Users who currently struggle with RTL-related bottlenecks will find that the Xcelium single-core engine has a lot to offer them. Most use cases will see a 20-30% speedup for RTL verification, with increases of up to 2X possible, depending on coding style.

The randomization engine in the Xcelium simulator—called the Xceligen engine—has also received substantial upgrades in nearly all areas. Featuring a new soft constraint engine, a new word-level engine, a new satisfiability (SAT) engine, a new distribution engine, a new type reuse architecture (TRAT), and improved solve order, the Xceligen engine has an improvement to fit any user's case.

The new soft constraint engine features new semantics that allow for faster and more accurate handling of soft constraints. This engine has a new solver—called a propagation-based solver—based on Cadence Specman® Elite technology that has been designed to handle a mixture of Boolean and arithmetic expressions. A word-level SAT solver has been added to that engine, upgrading the legacy SAT solver's Boolean interface. There's also a new set of more modern algorithms, improving execution times. The new TRAT infrastructure allows for the reuse of types between calls to a randomize function, which speeds up constraints on arrays of handles. In most IP-level environments like GPUs and CPUs, one often sees `randomize` calls repeated hundreds of thousands of times—so an improved system results in a very significant speedup.

For more general enhancements, there is a set of distribution optimizations that help users reach better coverage of the solution space and improve capturing the user's intent, along with a wide-reaching minor speedup for all users. Word interface instances are now reused between partitions with similar sets of constraints. There is also an overall memory footprint reduction and solve-order heuristics are more efficient.

The Xceligen engine can also generate a SystemVerilog test case on a partition level—rather than a randomize-call level—which makes for more accurate test cases that are easier to debug. To accompany this, there's improved contradiction analysis and error messages, which should help speed up the debug process significantly.

Multi-Core Engine Improvements

There is also a wide array of improvements to the Xcelium simulator's multi-core engine. The most improved area is the Xcelium simulator's performance on gate-level automatic test pattern generation (ATPG). Users will find the most acceleration there while encountering the lightest resistance. The Xcelium simulator's multi-core engine scales well with the number of cores—scalability has been proven through 12 cores and beyond—but users are advised that there is a point of diminishing return with large numbers of cores.

The Xcelium simulator's multi-core engine works best when it has uninterrupted access to a cache. Exact figures on speed improvement are highly dependent on how much of a given test bench is accelerable.

MCLite

Multi-core lite (MCLite) is a program that works in tandem with the Xcelium simulator and the user's distributed resource management (DRM), such as load-sharing facility (LSF), runtime design automation (RTDA), and grid tools, to optimize the use of resources on a server. Normally, users' use job submission tools to run all their simulation jobs all at once. It is most typical to set up the DRM tool to provide one slot per core (or one slot per two cores, if hyperthreading is enabled). For example, if a server has 12 cores available, and there are 12 jobs running on that server, each job will run slower because they are all competing for the same shared resources, such as the last level cache and the double data rate (DDR) channels. With MCLite, the user can request additional slots from the DRM tool, and MCLite will ensure that the Xcelium simulator makes appropriate use of the extra resources. MCLite does not do full multi-core simulation, but still achieves significant speedups in farm environments by allocating additional resources to a signal core job.

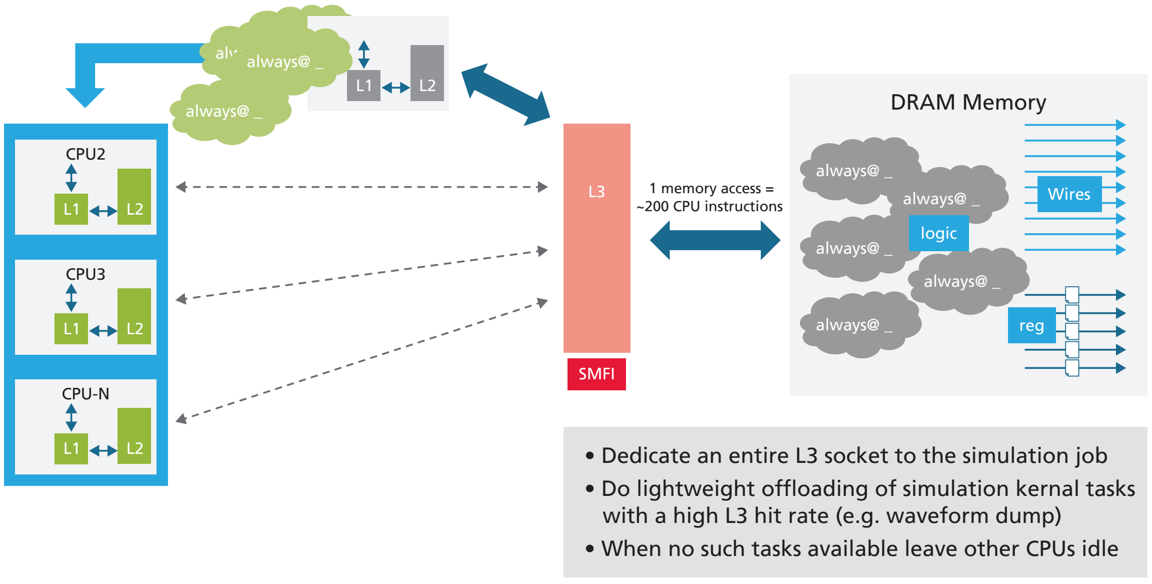
always@ _
always@ _
always@ _
L1
L2
CPU2
L1
L2
CPU3
L1
L2
CPU-N
L1
L2
L3
SMFI
1 memory access = ~200 CPU instructions
DRAM Memory
always@ _
always@ _
logic
always@ _
always@ _
always@ _
Wires
reg
• Dedicate an entire L3 socket to the simulation job
• Do lightweight offloading of simulation kernal tasks with a high L3 hit rate (e.g. waveform dump)
• When no such tasks available leave other CPUs idle

Figure 1: MCLite

As shown in Figure 1, MCLite reduces the load on a server's CPUs by dedicating an entire socket to the job. By redistributing the work required for simulation, MCLite speeds up the job significantly..

Parallel and Incremental Build

The Xcelium simulator features an industry-leading parallel build system. Previously called multi-snapshot incremental elaboration (MSIE), the parallel build uses an autopartitioning algorithm to determine how to best partition your design for parallelization and then performs a parallel build to speed up your build time. The speed increase is dependent on the use case, but SoC builds generally achieve greater than 2X speedup in build time. Parallel builds are also good for gate-level use cases. A good example is when a user receives multiple serial scan vectors from a manufacturing tool and has many different testbenches with different scan patterns. It doesn't matter that there are multiple scan patterns—the design isn't going to change at all, which means that the extensive build time (12 to 15 hours or more) becomes an even greater hassle. Using incremental build, this is no longer the case. No longer would one have to recompile the entire design for every scan pattern or functional pattern. This reduces build time drastically.

The parallel build works by placing the changing part of the model into an "incremental snapshot". The rest of the model—the unchanging portion—is placed into the "primary snapshot", where it is shared with the incremental pieces. This allows the system to compile only the parts of a design that change, providing huge speed gains, especially during debug, where only small parts of a design may change between recompilations.

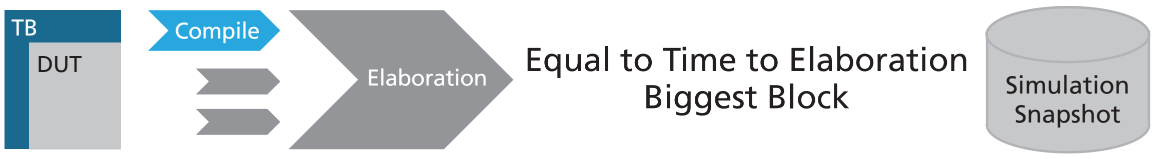

Figure 2: Parallel build flow

The important thing to note when comparing build flows is that the elaboration time is equal to the biggest block; everything else can be elaborated during the elaboration of that section.

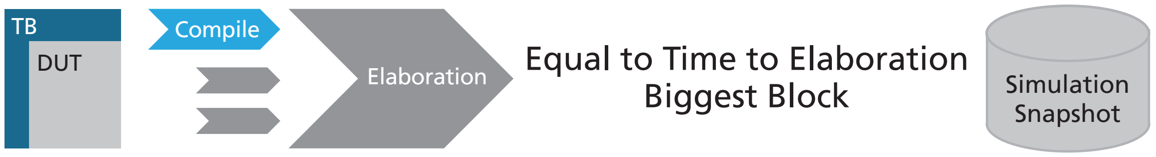

Figure 3: Incremental build flow

Incremental builds and parallel builds are used in opposite situations. One should use a parallel build flow when most or all the code needs to be rebuilt between runs. An incremental build flow is better for simulations where most of the code isn't changing.

For verification engineers, the simplified version of the parallel build system is especially useful—where there is a single precompiled portion of the model: the design and the unchanging portion of the verification environment. A verification engineer can then get a nearly instantaneous turnaround time while they are developing tests. What may have been previously a 30-minute build time now takes less than two minutes, when only the small portion of the verification environment must be built and linked with the pre-built portion of the model.

Save/Restart

The Xcelium simulator also includes an upgrade over the Incisive Enterprise Simulator's old save and restore system. This new save/restart capability solves many of the old issues or headaches caused by the old one. Now, a user is not forced to restart from a "clean" point, removing an old limitation on when it was possible to perform a save. The new system in the Xcelium simulator allows a user to save states from external code as well—meaning that users no longer need to make sure their C, C++ or other external models have been designed to work with save/restart. In addition, the save/restart system handles file I/O and multi-threaded applications, which makes the integration of the technology with a verification environment a largely trivial task.

When a checkpoint is created, the memory state of the entire model is saved, including any external code and the state of any files being read or written. If a user performs a warm restart, output continues where it was when the save was created; a cold restart gives the user the option of starting afresh with new command line options controlling the restart.

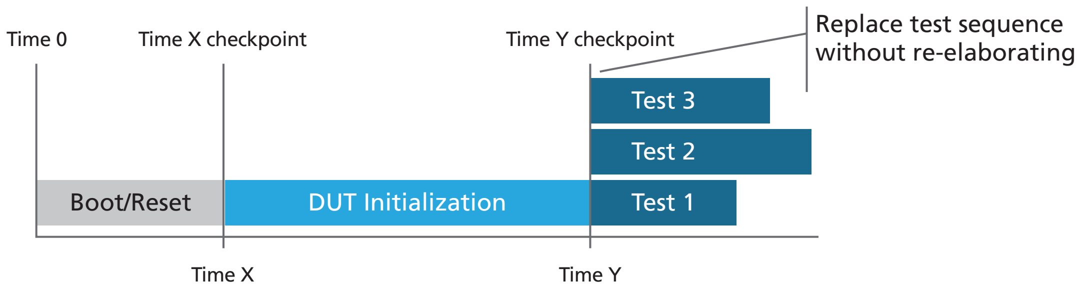

Figure 4: A breakdown showing how checkpointing can reduce or eliminate re-elaboration time

Figure 4 shows that process in action: restarting from the checkpoint at Time Y removes the need to re-elaborate—a user can simply run the tests again without the overhead from re-elaboration.

Note that the images created with this checkpointing system can be quite large; it is recommended that compression (`-zlib`) be used to reduce the size and that snapshots be managed. For example, if a snapshot is being saved every 30 minutes to keep multiple debug points in case of a failure, it is typically best to limit the number of saves to the previous five saves. Then, on the sixth save, remove the first saved snapshot to keep the amount of data controlled.

.

LBIST and MBIST

In automotive SoC development, reliability and safety are paramount. Everything must work perfectly the first time—and every subsequent time—and there's no room for error. Given that the aging cycle of many automotive SoCs is still poorly understood, it is vital that components in vehicles can prepare themselves for their own inevitable aging.

The best way to do this is through built-in self-test (BIST). The Cadence Modus DFT Software Solution brings these options to the table. BIST is a system in which devices can test themselves. For safety-critical devices, any failure can have catastrophic consequences, so a system that can test itself periodically to ensure its own components are working correctly is massively important to ensure the longevity of devices, such as computer-assisted braking systems in automobiles.

There are two types of BIST—memory built-in self-test (MBIST) and logic built-in self-test (LBIST). MBIST features a unified, programmable MBIST engine (PMBIST), with support for embedded SRAMs, ROMs, register files, and multiple unrestricted ports. A user can choose between accessing tests with JTAG or through direct access. Programmable algorithms can be hot-applied during runtime with diagnostics. It has self-repair and redundancy analysis tools built into the engine, along with shared test bus support.

The Xcelium simulator is the last part of PMBIST's multi-block bottom-up flow. After elaboration, the Cadence tool flow for PMBIST involves using the Cadence Genus™ Synthesis Solution to define the design for test (DFT) configuration mode for PMBIST, MBIST clocks, and test access methods then use the Genus Synthesis Solution to read memory views and add programmable MBIST logic. From there, the Modus DFT Software Solution and the Xcelium simulator work in tandem to write the PMBIST interface files and testbench and perform simulation verification.

LBIST is another critical aspect of functional safety. It applies pseudo-random patterns to a design and compares the resulting values against known good values to find faults. LBIST has high diagnostic coverage and low area overhead, and it's flexible; however, it destroys the state of the logic being tested, and, as such, that logic isn't available while the test is running.

LBIST features both OPCG support and JTAG-based control systems and can perform both integrated synthesis or RTL insertion. Programmable register sizing allows for the minimization of logic area overhead. LBIST can detect and block X-sources automatically and has programmable delay/dead cycles between scan/capture/scan states. For users interested in automotive applications of LBIST, there is a programmable interface for direct-access LBIST, which is critical for in-system automotive fault tolerant time interval (FTTI) requirements. A feature of LBIST diagnostics is that it custom-unloads MISR signatures.

The LBIST flow is like the MBIST flow—the Xcelium simulator is the final step once again. The Xcelium simulator is easily capable of verifying designs that feature BIST elements. For more information about the rest of this flow, refer to the Genus Synthesis Solution and the Modus DFT Software Solution web pages.

Verticals

The Xcelium simulator is a general-purpose simulator—if a user has a requirement, the Xcelium simulator has the tools and capabilities to meet that requirement. It works for all vertical applications.

For those concerned with massive digital applications such as telecommunications and mobile computing, the Xcelium simulator has long-latency simulation capabilities, MCLite, an industry-leading multi-core engine, incremental and parallel build, verification IP (VIP) protocols, and low power simulation.

If safety-critical systems are the user's focus—such as aerospace, defense, automotive, and medical— the Xcelium simulator brings safety features, industry-leading digital mixed signal verification, multi-language support including SystemVerilog and VHDL, concurrent fault simulation, FPGA support, and flow integration with formal analysis for unreachability to the table. It also features a strong connection with the Modus DFT Software Solution and the test flow, allowing for easy verification of DFT test flows that contain BISTs.

For those concerned with AI, neural nets, convolutional neural networks (CNNs), and advanced designs, the Xcelium simulator's multi-core engine will greatly help with the verification of designs with highly repeated structures, designs featuring cloning, and designs featuring learning algorithms that require many simulation passes to achieve signoff. The Xcelium simulator is also well equipped to handle designs of any size, even extremely large ones.

Mixed-Signal

The Xcelium simulator also features a wide variety of mixed-signal verification solutions. Integration with the Cadence Virtuoso® ADE Product Suite and the Cadence Spectre® Multi-Mode Simulator creates a cohesive simulation environment, allowing a user to perform full mixed-signal verification.

New to the Xcelium simulator in 2019 are advanced real-number modeling features: SystemVerilog can now control if real nets are shorted or disconnected at any specified time, making something like pixel design for image sensors a much less tedious process. Features such as coercion, multiple drivers and resolution support, wreal arrays, support for `` `wrealXstate `` and `` `wrealZstate ``, real assertions in both PSL and SVA, multi-language connections with VHDL, and AMSD control blocks for blocks without analog content have also been added.

The Xcelium simulator has expanded its port-binding capabilities over what a user may remember from the Incisive Verification Platform, and now has industry-leading bi-directional technology for SystemVerilog real number modeling, the smartest implementation of power intent, and distribution using the central power format (CPF).

X-Prop, Power, Debug

In today's designs, low-power needs are ubiquitous. Users need a low power debugging system that is accurate and doesn't slow build or runtime performance.

For low power devices, the Xcelium simulator has them covered. Using the Unified Power Format 2.0 (UPF), the power needs of a chip can be easily modeled. UPF does this using supply nets, which are SystemVerilog structs that contain a state and an integer called "voltage" that holds the value of the voltage in microvolts. The "state" field of the supply net can hold four different values: **OFF**, **UNDETERMINED**, **PARTIAL_ON**, and **FULL_ON**. With this flexibility, one can accurately model how the different sections of a chip power on, power down, or power off during testing.

For logic managing needs, the Xcelium simulator comes equipped with X-propagation technology. X-propagation allows the user to see how uncertainties (X's) move through a design as tests are applied. In warm-reset scenarios, the state of a chip must hold while other parts power down. Since there are countless numbers of different warm reset scenarios, a verification engineer needs their design to be ready for anything. With X-propagation, a user can see how an unresolved X-state caused by a power-down or other cause affects other parts of the design. X-propagation analysis can be run in either compute as ternary (CAT) mode, where X propagates exactly as it would in hardware, or in forward only X (FOX) mode, where X propagates disregarding inputs. The Xcelium simulator's X-propagation technology supports both SystemVerilog and VHDL and doesn't require the user to change their existing HDL designs.

For other debugging needs, there's the Cadence SimVision™ Debug and the Cadence Indago™ Debug. SimVision debug and Indago debug excel in different areas of the debug process—many users use both, but the amount each is used will vary depending on the use case. For more information about SimVision Debug and Indago Debug, look here and here, respectively.

Safety

As the electronic systems in cars become more complex, ensuring that those systems work perfectly for as long as possible is paramount. Any defect caused by even the most minor unexpected stimuli can cause a fault with possibly catastrophic effects. A safety engineer's task, then, is to test that system as thoroughly as possible, and even some beyond that—faults in every place in every conceivable manner, from every conceivable source. Older simulators lacked in-house functionality for a lot of important safety features, but the Xcelium simulator has everything a safety engineer would need—meeting the requirements of the IEC 61508 standard and the ISO 26262 automotive safety standard derived from it.

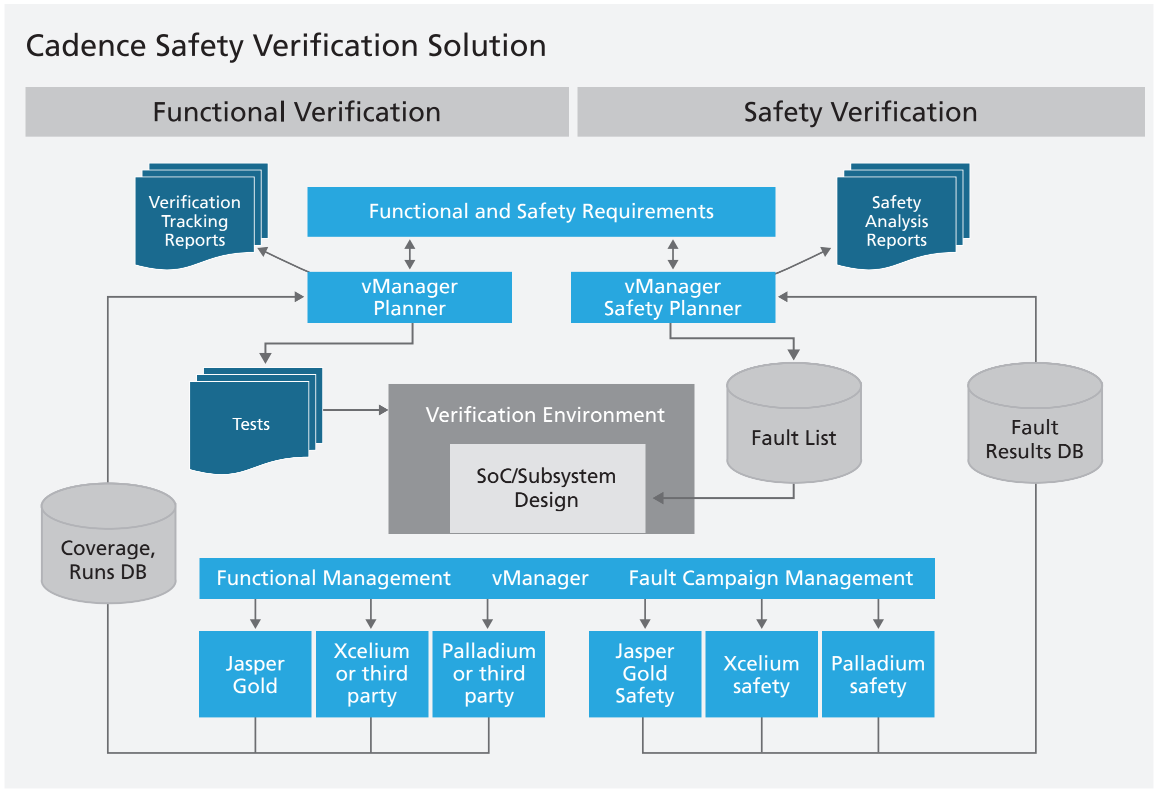

Figure 5: The Cadence safety verification solution

This is one of the places where the Cadence vManager™ Metric-Driven Signoff Platform comes in. The vManager platform controls a huge amount of the safety verification flow. One of the most important areas there is the vManager safety planner. The vManager safety planner produces failure modes, effects, and diagnostics analysis (FMEDA) plan, which allows a verification engineer to automatically schedule safety campaigns on a failure modes basis to measure safety and diagnostics coverage metrics.

The Cadence JasperGold® Formal Verification Platform is a part of this flow, too. It offers structural fault analysis with structural fault connectivity, activatability, and relation analysis; formal fault analysis with formal activatability and propagability analysis; and custom safety and security analysis, which can create custom strobes and fault specifications to model attacks from hackers. The JasperGold platform automatically annotates structurally-safe faults in a given database of faults, and the Xcelium simulator runs all the other faults from that database.

Functional safety is a highly optimized flow, and the Cadence solution is the most comprehensive one on the market. There's nothing the complete flow can't handle—from FMEDA capturing and optimized fault management to the fault injection and classification natively built into the JasperGold platform, the Xcelium simulator, and the Cadence Palladium® series, the Cadence safety flow suits all needs.

Conclusion

No matter the use case, there is an improvement in the Xcelium simulator that fits the need. For those interested in single-core performance, there are speed-ups across the board, including an overhauled randomization component of the Xceligen engine that fits any user's needs. For multi-core engine users, there are large improvements in the speed of gate-level ATPG and gate-level with SDF time. And there is always MCLite, which allows users to make a latency versus throughput tradeoff for any long latency test. There is a wide array of new features in the engine, too, like the save/restart functionality, incremental build, and x-propagation technology, with improvements in power handling and debug capabilities. Users with safety concerns can find everything they are looking for in the Xcelium simulator, too, with all its fault simulation and fault reduction technologies.

Users who worked with Incisive Verification Platform will find the Xcelium simulator to be a refreshing improvement on the simulator they already knew and loved—and users who are using a different simulator will find features that no other simulator on the market today has. The Xcelium simulator is the cutting edge—a new generation simulator for a new generation of designs.

The Xcelium Parallel Logic Simulator is part of the Cadence Verification Suite and supports the company's Intelligent Design Enablement strategy, which enables systems and semiconductor companies to create complete, differentiated end products more efficiently. The Cadence Verification Suite is comprised of the best-in-class Xcelium simulator, JasperGold platform, the Palladium® Z1 Enterprise Emulation Platform, and the Cadence Protium™ S1 FPGA-Based Prototyping Platform core engines, verification fabric technologies and solutions that increase design quality and throughput—all fulfilling verification requirements for a wide variety of applications and vertical segments.

cādence®

Cadence software, hardware and semiconductor IP enable electronic systems and semiconductor companies to create the innovative end products that are transforming the way people live, work, and play. The company's Intelligent System Design strategy helps customers develop differentiated products—from chips to boards to systems. **www.cadence.com**

© 2019 Cadence Design Systems, Inc. All rights reserved worldwide. Cadence, the Cadence logo, and the other Cadence marks found at www.cadence.com/go/trademarks are trademarks or registered trademarks of Cadence Design Systems, Inc. All other trademarks are the property of their respective owners. 12496 06/19 MC/RA/PDF