Robert F. Kramer (SBN 181706)
rkramer@kramerday.com
David Alberti (SBN 220625)
dalberti@kramerday.com
Sal Lim (SBN 211836)
slim@kramerday.com
Russell Tonkovich (SBN 233280)
rtonkovich@kramerday.com
Robert C. Mattson (*pro hac vice to be filed*)
rmattson@kramerday.com
James P. Barabas (*pro hac vice to be filed*)
jbarabas@kramerday.com
Hong S. Lin (SBN 249898)
hlin@kramerday.com
Robert Y. Xie (SBN 329126)
rxie@kramerday.com
KRAMER ALBERTI LIM & TONKOVICH LLP
577 Airport Boulevard, Suite 250
Burlingame, California 94010
Telephone: (650) 825-4300
Facsimile: (650) 460-8443

Attorneys for Plaintiff
Semiconductor Design Technologies, LLC

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN JOSE DIVISION

SEMICONDUCTOR DESIGN TECHNOLOGIES, LLC,

Plaintiff,

v.

CADENCE DESIGN SYSTEMS, INC.,

Defendant.

Case No. 5:23-cv-01001

COMPLAINT FOR PATENT INFRINGEMENT

DEMAND FOR JURY TRIAL

COMPLAINT FOR PATENT INFRINGEMENT

1. Plaintiff Semiconductor Design Technologies, LLC ("Semiconductor Design" or "Plaintiff"), by its attorneys, demands a trial by jury on all issues so triable and for its Complaint against Cadence Design Systems, Inc. ("Cadence" or "Defendant") alleges the following:

NATURE OF THE ACTION

2. This action arises under 35 U.S.C. § 271 for Cadence's infringement of Semiconductor Design's United States Patent Nos. 7,603,636 (the "'636 patent") and 7,971,167 (the "'167 patent") (collectively, the "Asserted Patents").

THE PARTIES

3. Semiconductor Design is a corporation organized under the laws of the State of Delaware with a principal place of business at 1000 N. West Street, Suite 1200, Wilmington, DE 19801.

4. Upon information and belief, Cadence Design Systems, Inc. is a company organized and existing under the laws of the State of Delaware, with a place of business at 2655 Seely Avenue, San Jose, CA 95134. Cadence may be served through its registered agent, CT Corporation System, for service at 330 North Brand Blvd, #700, Glendale, CA 91203.

5. Upon information and belief, Cadence is a global supplier of electronic system design tools, including electronic design automation ("EDA") and analog design environment ("ADE") software tools used to design, develop, and test semiconductor chips.

JURISDICTION AND VENUE

6. This Court has jurisdiction over the subject matter of this action pursuant to 28 U.S.C. §§ 1331 and 1338(a).

7. Upon information and belief, jurisdiction and venue for this action are proper in the Northern District of California.

8. This Court has personal jurisdiction over Cadence because Cadence has purposefully availed itself of the rights and benefits of the laws of this Judicial District. Upon information and belief, Cadence resides in the Northern District of California by maintaining a regular and established place of business at 2655 Seely Avenue, San Jose, CA 95134.

9. This Court also has personal jurisdiction over Cadence because Cadence has done and is doing substantial business in this Judicial District both generally and, upon information and belief, with respect to the allegations in this Complaint, including Cadence's one or more acts of infringement in this Judicial District.

10. Venue is proper in this Judicial District under 28 U.S.C. §§1391(b) and (c) and §1400(b). Cadence has committed acts of infringement through, for example, installing its Stratus HLS software on computers it uses to test the functionality of its Stratus HLS software, as well as providing support for its customers. Moreover, Cadence has a regular and established place of business in this District. Cadence's Principal Executive Offices are physically located in San Jose in the District.

INTRADISTRICT ASSIGNMENTS

11. Pursuant to Local Rule 3-2 (c), this case involves intellectual property rights and is subject to assignment on a district-wide basis.

THE ASSERTED PATENTS

12. Semiconductor Design is the lawful owner of all rights, title, and interests in the '636 patent titled "Assertion Generating System, Program Thereof, Circuit Verifying System, and Assertion Generating Method," including the right to sue and recover for infringement thereof. The '636 patent was duly and legally issued on October 13, 2009, naming Takamitsu Yamada as the inventor. A true and correct copy of the '636 patent is attached as Exhibit A.

13. The '636 patent relates to generating and verifying design data of a semiconductor integrated circuit. For example, the '636 patent discloses graphically editing a specification of an integrated circuit and generating a property that verifies the specification of the semiconductor integrated circuit based on design data. The property is in turn converted into an assertion description language if the property is to be verified during asset verification.

14. The claims of the '636 patent do not merely recite the performance of a preexisting method that generates assertions, but rather are directed to specific technological improvements to semiconductor design and verification technology. Other methods generate assertions from the RTL description, and as a result the assertion description is not guaranteed to match the circuit

specification.

15. The claims of the '636 patent do not preempt all ways of generating assertions, but are rather directed to specific approaches of generating assertions based on a property read from design data generated from a specification of a semiconductor integrated circuit.

16. Accordingly, each claim of the '636 patent recites specific improvements to semiconductor design and verification technology and/or inventive concepts.

17. Semiconductor Design is the lawful owner of all rights, title, and interests in the '167 patent titled "Semiconductor Design Support Device, Semiconductor Design Support Method, and Manufacturing Method for Semiconductor Integrated Circuit," including the right to sue and recover for infringement thereof. The '167 patent was duly and legally issued on June 28, 2011, naming Yasutaka Tsukamoto as the inventor. A true and correct copy of the '167 patent is attached as Exhibit B.

18. The '167 patent relates to a semiconductor design support device for designing a semiconductor integrated circuit. A behavioral description describes an algorithm of processing performed by hardware at a motion (*e.g.*, operation) level. A tool, such as a behavioral synthesis tool, generates an RTL description from the behavioral description, and specifies in the RTL description registers and clock synchronisms particular to the hardware. The latency analyzer analyzes a result of a logic simulation performed on the RTL description to calculate a latency in each block representing an operation in a predetermined unit in the behavioral description.

19. The claims of the '167 patent do not merely recite a preexisting method of performance, but rather are directed to specific technological improvements to semiconductor design, analysis, and simulation technology. Preexisting methods do not efficiently check the latency of the blocks of the behavioral description.

20. The claims of the '167 patent do not preempt all ways of generating an RTL description or performing a logic simulation on the RTL description, but are rather directed to specific approaches of mapping states in the RTL description to blocks in the behavioral description in order to determine the latency of each block from the RTL simulation.

21. Accordingly, each claim of the '167 patent thus recites a combination of elements

sufficient to ensure that the claim amounts to significantly more than a patent on an ineligible concept.

22. Semiconductor Design is the owner of all right, title, and interest in and to each of the Asserted Patents with full and exclusive right to bring suit to enforce the Asserted Patents, including the right to recover for past damages and/or royalties prior to the expiration of the Asserted Patents.

23. The Asserted Patents are valid and enforceable.

COUNT 1 – INFRINGEMENT OF U.S. PATENT NO. 7,603,636

24. Semiconductor Design incorporates by reference the allegations contained in paragraphs 1-23 above.

25. Cadence provides software products for verifying a graphically-edited specification of a semiconductor integrated circuit ("the 636 Accused Products"), that when created, stored, or used by Cadence or its customers, infringes, either literally or under the doctrine of equivalents, one or more claims of the '636 patent in violation of 35 U.S.C. § 271(a). Stratus HLS is referenced herein as an exemplary 636 Accused Product in connection with Semiconductor Design's allegations of infringement.

26. Upon information and belief, Cadence has directly infringed and continues to directly infringe at least claim 8 of the '636 patent because Cadence makes, uses, sells, and/or offers to sell its 636 Accused Products, which are stored on computer-readable media encoded with a program for a computer in an assertion generating system. Cadence's infringing use of the 636 Accused Products includes its internal use and testing of those products, its demonstration of the 636 Accused Products to third parties, its storage of 636 Accused Products on servers for transmitting the 636 Accused Products to customers or for hosting those products, and its distribution of copies of the 636 Accused Products to customers.

27. Upon information and belief, by at least as early as the filing or service of this Complaint, Cadence had actual knowledge of the '636 patent and the infringing nature of its products.

28. Upon information and belief, the '636 patent has been cited by the following

Cadence patents: U.S. Patent Nos. 7,712,060, 7,810,056, 9,842,183, and 10,922,469. Thus, Cadence was aware of the '636 patent prior to the filing of this Complaint.

29. The Stratus HLS software is an example of the 636 Accused Products and causes a computer provided in an assertion generating system to generate an assertion description.

> ▸ Synthesis of SystemC assertions and C++ asserts to SystemVerilog assertions (SVAs)
>
> Integrated with the Cadence verification suite, Stratus HLS supports automated mixed-language (SystemC and RTL) verification and debug including assertions, debugging, waveforms, and linkage back to the original SystemC design.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf (annotated).

30. The Stratus HLS software is used for assertion verification of a semiconductor integrated circuit.

> ▸ Synthesis of SystemC assertions and C++ asserts to SystemVerilog assertions (SVAs)
>
> Integrated with the Cadence verification suite, Stratus HLS supports automated mixed-language (SystemC and RTL) verification and debug including assertions, debugging, waveforms, and linkage back to the original SystemC design.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf (annotated).

31. The computer upon which the Stratus HLS software is installed executes a specification inputting step that generates design data of the semiconductor integrated circuit by graphically editing a specification of the semiconductor integrated circuit based on user operations.

GUI

The Stratus GUI incorporates an IDE, making SystemC development easy and intuitive for new users and advanced users alike. In addition to typical IDE features, the Stratus IDE makes it easy to quickly create new models using pre-defined design templates to reduce design and debugging time.

The Stratus analysis environment includes SystemC and RTL source linking, control and dataflow graphs, schematic viewer, and pipeline analysis, as well as QoR reporting and visualization to judge the impact of architectural optimizations. Although most commonly used via the GUI, this analysis is also available via the Stratus Tcl API.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf (annotated).

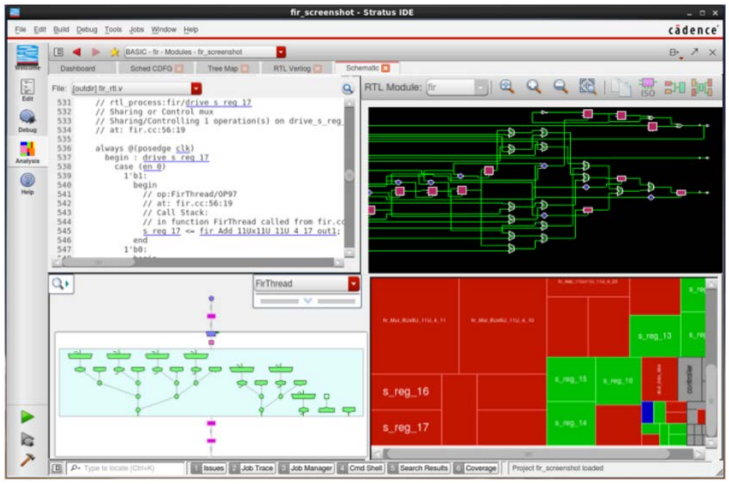
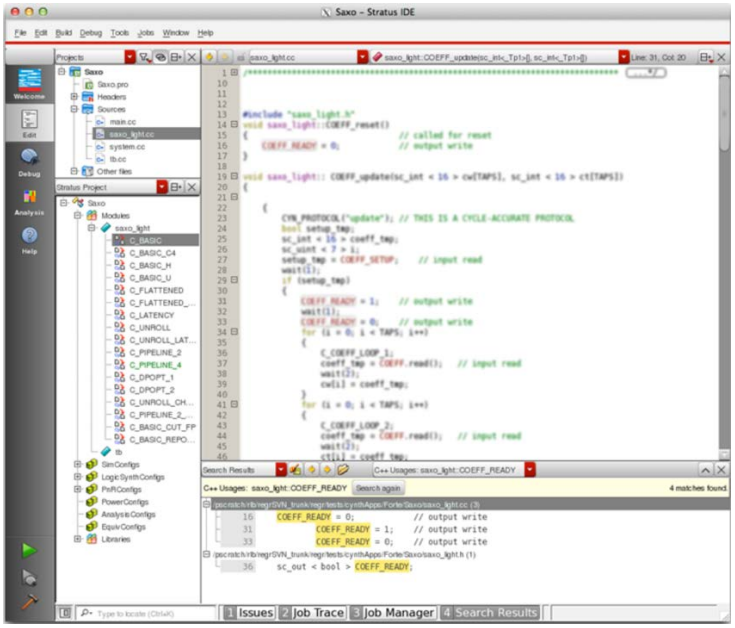
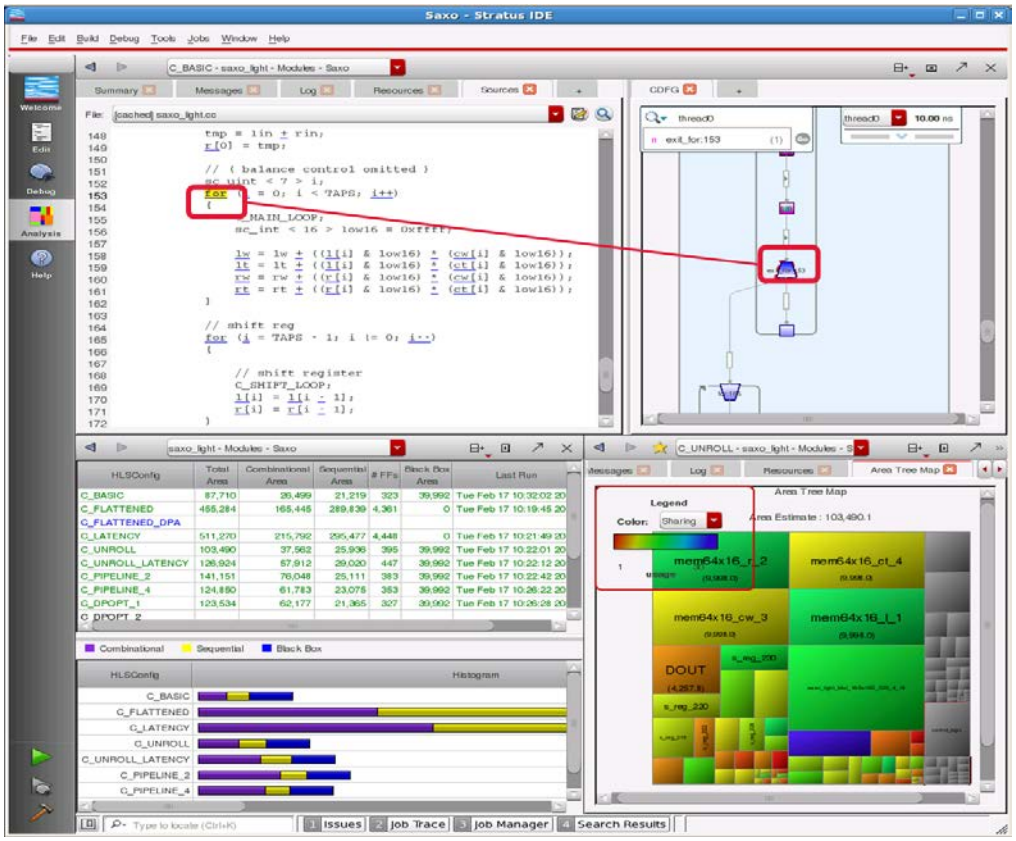


Figure 2: Complete graphical analysis with links to source code

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf



Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

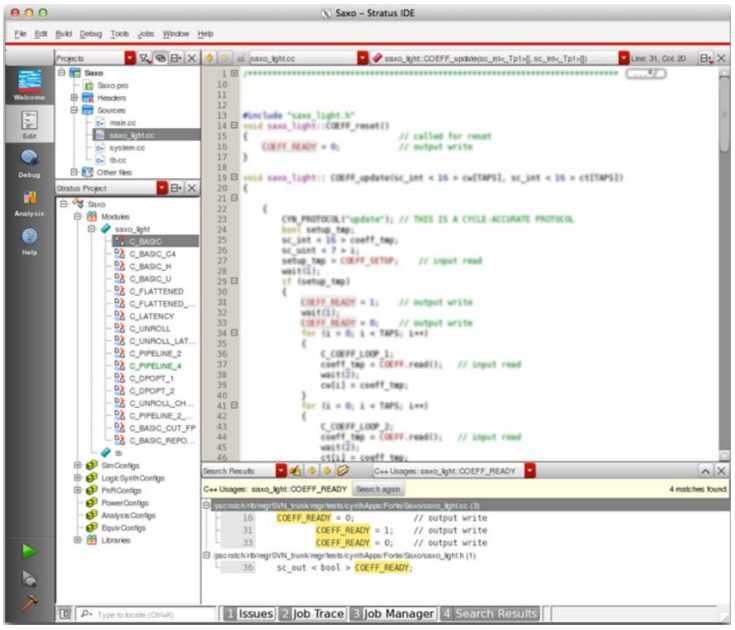


Source: https://www.linkedin.com/pulse/how-dramatically-reduce-time-from-architecture-spec-tapeout-laviv/

> Integrated with the Cadence verification suite, Stratus HLS supports automated mixed-language (SystemC and RTL) verification and debug including assertions, debugging, waveforms, and linkage back to the original SystemC design.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

32. As part of the specification inputting step, the computer upon which the Stratus HLS software is installed inputs the design data in storage.



Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

33. The computer upon which the Stratus HLS software is installed executes a property generating step that reads the design data generated at the specification inputting step from the storage.

> ▸ Synthesis of SystemC assertions and C++ asserts to SystemVerilog assertions (SVAs)

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

> Integrated with the Cadence verification suite, Stratus HLS supports automated mixed-language (SystemC and RTL) verification and debug including assertions, debugging, waveforms, and linkage back to the original SystemC design.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf (annotated).

> Another way to improve DFV is for designers to add assertions in their code to identify illegal conditions. For example, the high-priority thread should never get a NACK response on a memory access request. Using Stratus™ HLS, SystemC/C++ assertions can be synthesized into the generated RTL implementation by turning on the *synthesize_asserts* feature. This allows the designer to communicate effectively and clearly the design intend and assumptions to the verifiers. The verifiers would use these assertions as targets to be verified and covered in their testbenches. Assertions are clear coverage analysis points that can be reviewed using assertion coverage features in simulation. The following figure shows an assertion in the input behavioral SystemC® design, and the equivalent Verilog assertion synthesized into the produced RTL, and also the assertion coverage report that confirms that this design feature was exercised and verified by the verification test cases.

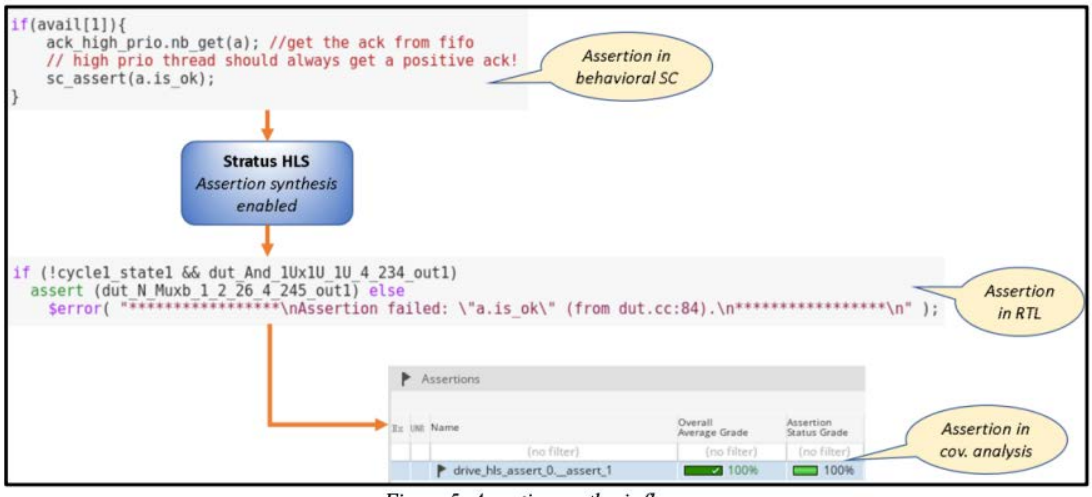


Figure 5: Assertion synthesis flow

Source: S. Dahir, "Using HLS to improve Design-for-Verification of multi-pipeline designs with resource sharing," DVCON 2021 at 5-6, available at https://dvcon-proceedings.org/wp-content/uploads/using-hls-to-improve-design-for-verification-of-multi-pipeline-designs-with-resource-sharing.pdf.pdf.

34. As part of the property generating step, the computer upon which the Stratus HLS software is installed generates a property which verifies the specification of the semiconductor integrated circuit using the read design data.

> 2. **Customize the Setup:** Stratus HLS' Behavioral Design Workbench (BDW) import function reads the design, testbench, and metadata and creates a Stratus HLS project. The user configures the Stratus HLS project by specifying the technology library, design constraints, and exploration settings.
> 3. **Exploration:** Once configured, Stratus HLS explores the solution space. Exploration is governed by a Stratus HLS Tcl control file that defines how constraints are changed and imposed on the candidate micro-architecture designs.

Source: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/cadence-stratus-hls-algorithm-wp.pdf (annotated).

> ▸ Synthesis of SystemC assertions and C++ asserts to SystemVerilog assertions (SVAs)

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

> Integrated with the Cadence verification suite, Stratus HLS supports automated mixed-language (SystemC and RTL) verification and debug including assertions, debugging, waveforms, and linkage back to the original SystemC design.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf (annotated).

> Another way to improve DFV is for designers to add assertions in their code to identify illegal conditions. For example, the high-priority thread should never get a NACK response on a memory access request. Using Stratus™ HLS, SystemC/C++ assertions can be synthesized into the generated RTL implementation by turning on the *synthesize_asserts* feature. This allows the designer to communicate effectively and clearly the design intend and assumptions to the verifiers. The verifiers would use these assertions as targets to be verified and covered in their testbenches. Assertions are clear coverage analysis points that can be reviewed using assertion coverage features in simulation. The following figure shows an assertion in the input behavioral SystemC® design, and the equivalent Verilog assertion synthesized into the produced RTL, and also the assertion coverage report that confirms that this design feature was exercised and verified by the verification test cases.

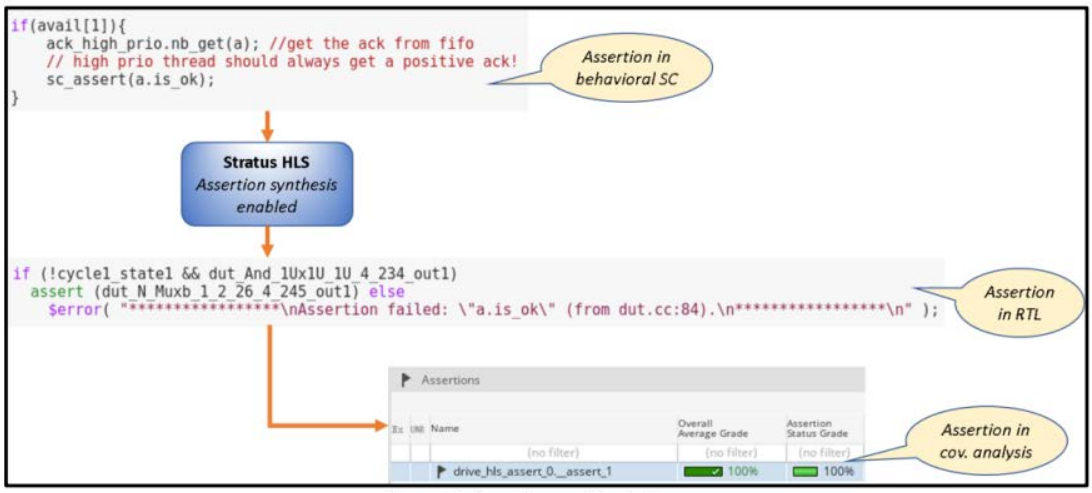


Figure 5: Assertion synthesis flow

Source: S. Dahir, "Using HLS to improve Design-for-Verification of multi-pipeline designs with resource sharing," DVCON 2021 at 5-6, available at https://dvcon-proceedings.org/wp-content/uploads/using-hls-to-improve-design-for-verification-of-multi-pipeline-designs-with-resource-sharing.pdf.pdf.

35. As part of the property generating step, the computer upon which the Stratus HLS software is installed inputs the property in the storage.

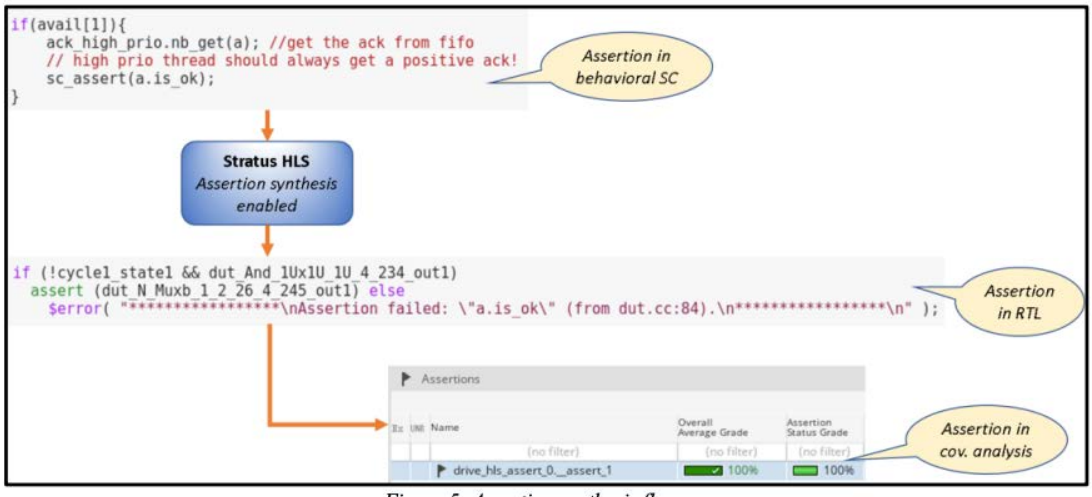
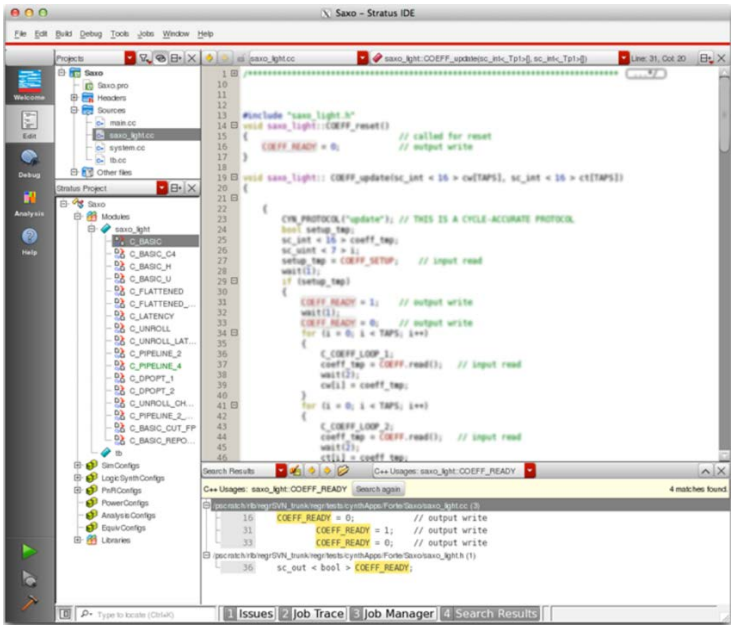


Figure 5: Assertion synthesis flow

Source: S. Dahir, "Using HLS to improve Design-for-Verification of multi-pipeline designs with resource sharing," DVCON 2021 at 5-6, available at https://dvcon-proceedings.org/wp-

content/uploads/using-hls-to-improve-design-for-verification-of-multi-pipeline-designs-with-resource-sharing.pdf.pdf



Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

36. The computer upon which the Stratus HLS software is installed executes an assertion generating step that reads the property generated at the property generating step from the storage.

Figure 5: Assertion synthesis flow

Source: S. Dahir, "Using HLS to improve Design-for-Verification of multi-pipeline designs with resource sharing," DVCON 2021 at 5-6, available at https://dvcon-proceedings.org/wp-content/uploads/using-hls-to-improve-design-for-verification-of-multi-pipeline-designs-with-resource-sharing.pdf.pdf

37. As part of the assertion generating step, the computer upon which the Stratus HLS software is installed automatically converts the property into an assertion description if the property is to be verified during assertion verification.

- Synthesis of SystemC assertions and C++ asserts to SystemVerilog assertions (SVAs)

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

Integrated with the Cadence verification suite, Stratus HLS supports automated mixed-language (SystemC and RTL) verification and debug including assertions, debugging, waveforms, and linkage back to the original SystemC design.

https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf (annotated).

Another way to improve DFV is for designers to add assertions in their code to identify illegal conditions. For example, the high-priority thread should never get a NACK response on a memory access request. Using Stratus™ HLS, SystemC/C++ assertions can be synthesized into the generated RTL implementation by turning on the *synthesize_asserts* feature. This allows the designer to communicate effectively and clearly the design intend and assumptions to the verifiers. The verifiers would use these assertions as targets to be verified and covered in their testbenches. Assertions are clear coverage analysis points that can be reviewed using assertion coverage features in simulation. The following figure shows an assertion in the input behavioral SystemC® design, and the equivalent Verilog assertion synthesized into the produced RTL, and also the assertion coverage report that confirms that this design feature was exercised and verified by the verification test cases.

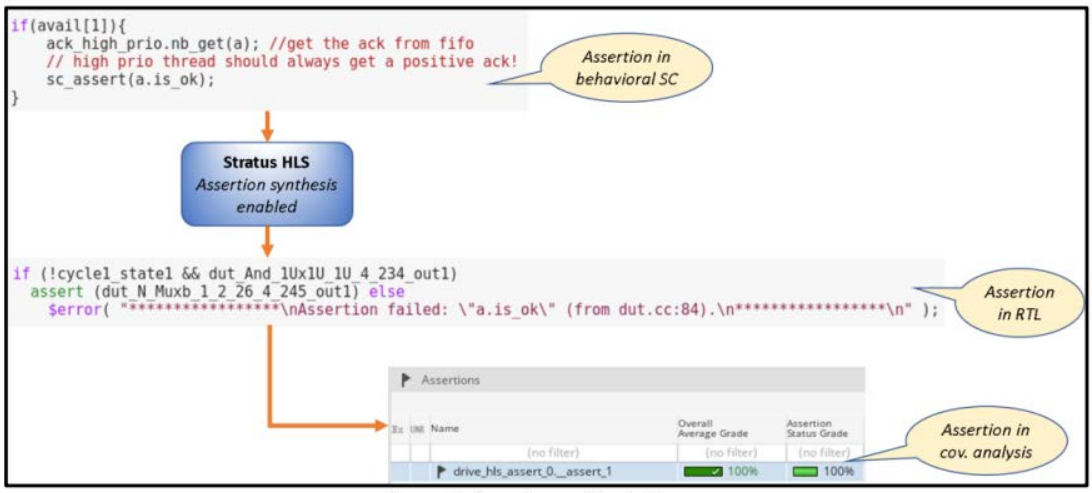


Figure 5: Assertion synthesis flow

Source: S. Dahir, "Using HLS to improve Design-for-Verification of multi-pipeline designs with resource sharing," DVCON 2021 at 5-6, available at https://dvcon-proceedings.org/wp-content/uploads/using-hls-to-improve-design-for-verification-of-multi-pipeline-designs-with-resource-sharing.pdf.pdf.

38. Properties and assertions for describing the design include selection conditions with respect to state transitions, logic values, and signals in the design data.

> Another way to improve DFV is for designers to add assertions in their code to identify illegal conditions. For example, the high-priority thread should never get a NACK response on a memory access request. Using Stratus™ HLS, SystemC/C++ assertions can be synthesized into the generated RTL implementation by turning on the *synthesize_asserts* feature. This allows the designer to communicate effectively and clearly the design intend and assumptions to the verifiers. The verifiers would use these assertions as targets to be verified and covered in their testbenches. Assertions are clear coverage analysis points that can be reviewed using assertion coverage features in simulation. The following figure shows an assertion in the input behavioral SystemC® design, and the equivalent Verilog assertion synthesized into the produced RTL, and also the assertion coverage report that confirms that this design feature was exercised and verified by the verification test cases.

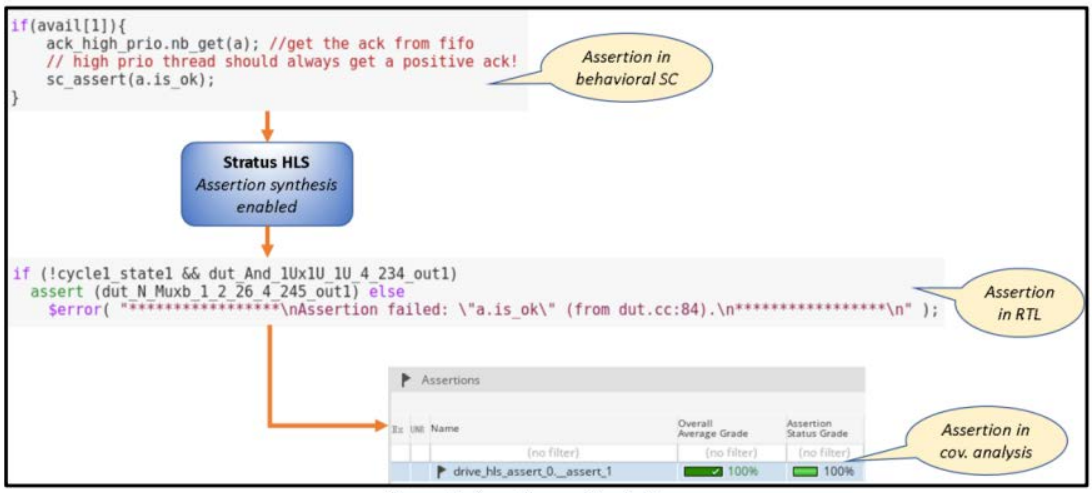


Figure 5: Assertion synthesis flow

Source: S. Dahir, "Using HLS to improve Design-for-Verification of multi-pipeline designs with resource sharing," DVCON 2021 at 5-6, available at https://dvcon-proceedings.org/wp-content/uploads/using-hls-to-improve-design-for-verification-of-multi-pipeline-designs-with-resource-sharing.pdf.pdf.

39. Upon information and belief, Cadence has indirectly infringed and continues to indirectly infringe at least claim 8 of the '636 patent in violation of 35 U.S.C. § 271(b). From at least the time Cadence received notice of its infringement, Cadence has induced others to infringe at least one claim of the '636 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Cadence's clients, customers, and end users, whose use of the Accused Products constitute direct infringement of at least one claim of the '636 patent. In particular, Cadence's actions that aided and abetted others such as customers and end users to infringe include advertising and distributing the Accused Products, providing instruction materials, support training, and services regarding the Accused Products, and actively inducing its customers to acquire and/or install the infringing products, including Stratus HLS software, on customer-provided computer-readable media to be used in connection with a computer in an assertion-generating system for generating assertion descripts for validation of a semiconductor integrated circuit. *See, e.g.*, https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-

level-synthesis.html; https://www.cadence.com/en_US/home/support.html, including all related domains and subdomains. Cadence does so knowing that its customers will commit these infringing acts. Despite its knowledge of the '636 patent, Cadence continues to make, use, sell, and/or offer for sale the 636 Accused Products thereby specifically intending for and inducing its customers to infringe the '636 patent.

40. Upon information and belief, Cadence has indirectly infringed and continues to indirectly infringe at least claim 8 of the '636 patent in violation of 35 U.S.C. § 271(c) by contributing to the infringement by its customers. Cadence sells or offers for sale in the United States the 636 Accused Products, including the Stratus HLS software, with knowledge that they are especially designed or adapted to operate in a manner that infringes that patent and despite the fact that the infringing technology or aspects of the 636 Accused Products are not a staple of commerce suitable for substantial non-infringing use. For example, Cadence knows that the Stratus HLS software infringes when stored on a computer readable media because it enables a computer to provide an assertion-generation system that generates an assertion description for assertion verification of a semiconductor integrated circuit and to execute the steps recited in claim 8. Cadence is aware that the Stratus HLS software operates as described above, that such functionality infringes the '636 patent, including claim 8, and that the Accused Products have no substantial non-infringing use. Cadence continues to sell and offer for sale in the United States its infringing products after receiving notice of the '636 patent and how it is infringed by Cadence's products. The portion of the Stratus HLS software that maps to claim 8 (i.e., the infringing aspect) has no substantial non-infringing uses.

41. Cadence's infringement has damaged and continues to damage and injure Semiconductor Design.

42. Semiconductor Design is entitled to recover the damages sustained as a result of Cadence's wrongful acts in an amount subject to proof at trial.

COUNT 2 – INFRINGEMENT OF U.S. PATENT NO. 7,971,167

43. Semiconductor Design incorporates by reference the allegations contained in paragraphs 1 to 42 above.

44. Cadence provides software products for generating an RTL description from a behavioral description and calculating the latency of blocks in the behavioral description that correspond to states in the RTL description ("the 167 Accused Products"), that, when installed or used by Cadence or its customers, infringes, either literally or under the doctrine of equivalents, one or more claims of the '167 patent in violation of 35 U.S.C. § 271(a). Stratus HLS is referenced herein as an exemplary 167 Accused Product in connection with Semiconductor Design's allegations of infringement.

45. Upon information and belief, Cadence has directly infringed and continues to directly infringe at least claim 1 of the '167 patent by making, using, selling, and/or offering for sale the 167 Accused Products to provide a semiconductor design support device for designing a semiconductor integrated circuit. Cadence's infringing use of the 167 Accused Products includes its internal installation, use, and/or testing of those products, its demonstration of the 167 Accused Products to third parties, and/or its hosting or installation of the 167 Accused Products for or on behalf of third parties.

46. Upon information and belief, by at least as early as the filing or service of this Complaint, Cadence had actual knowledge of the '167 patent and the infringing nature of its products.

47. A computer installed with the Stratus HDL is a semiconductor design support device for designing a semiconductor integrated device. Stratus HDL starts with a behavioral description, which is then converted into HDL.

> ▸ Synthesis of SystemC assertions and C++ asserts to SystemVerilog assertions (SVAs)

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

> Integrated with the Cadence verification suite, Stratus HLS supports automated mixed-language (SystemC and RTL) verification and debug including assertions, debugging, waveforms, and linkage back to the original SystemC design.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf (annotated).

> Cadence® Stratus™ High-Level Synthesis (HLS) automatically creates high-quality register transfer level (RTL) design implementations for ASIC, system-on-chip (SoC), and FPGA targets from high-level IEEE 1666 SystemC™, C++, and MATLAB® descriptions. The proven successes of Stratus HLS in production designs around the world are testament to its consistently high-quality results, mature feature set, and complete design coverage. While most widely used for image processing, wireless, and machine learning (ML) applications, products built with Stratus HLS technology can be found in your home, automobile, and pockets.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

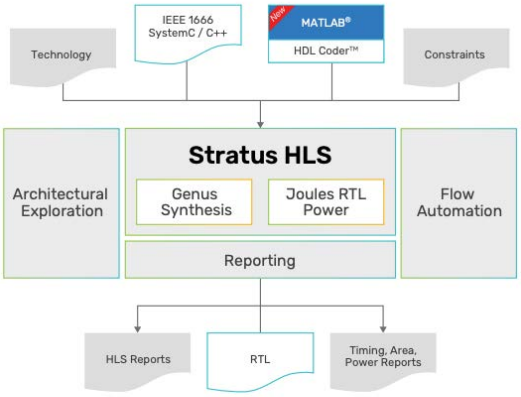


Figure 1: Stratus HLS uses the Genus synthesis and Joules power engines to create high-quality RTL targeted to your technology and design constraints

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

48. The semiconductor design support device based on the Stratus HLS software receives and stores, and thus includes, a behavioral description configured to describe an algorithm of processing performed by hardware in a motion level. For example, behavioral descriptions can be provided in SystemC and C++ models.

> Stratus HLS supports untimed and timed SystemC and C++ models, including a mix of both, providing maximum flexibility to the designer. The output can be fully pipelined (new data each cycle), pipelined at reduced throughput (new

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

> **Behavioral IP Reuse**
>
> Stratus HLS enables the creation and adaptation of behavioral IP, delivering on the promise of true IP reuse.
>
> Using Stratus HLS, the verified source code can be reused without modification for widely different process technologies, clock speeds, or PPA targets. Modifications to the algorithm, architecture, or interfaces can be made incrementally at a high level, where previously they required a complete RTL rewrite.
>
> Behavioral IP reuse with Stratus HLS significantly reduces overall design effort and maximizes return on investment (ROI).

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

49. The semiconductor design support device based on Stratus HDL generates and stores, and thus includes, an RTL description generated by reading the behavioral description.

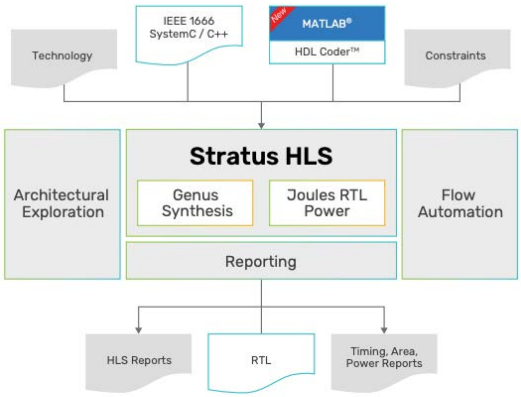
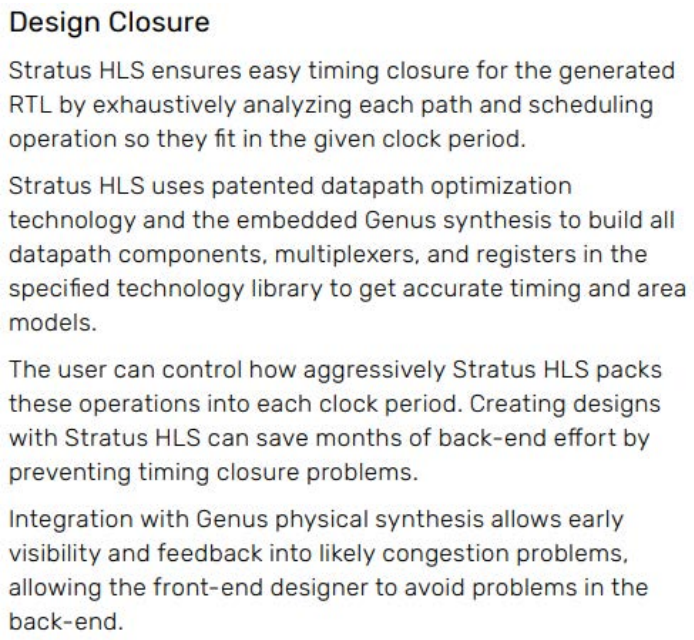


Figure 1: Stratus HLS uses the Genus synthesis and Joules power engines to create high-quality RTL targeted to your technology and design constraints

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

50. The RTL description is configured to recognize a concept including register and clock synchronism particular to the hardware.



Design Closure

Stratus HLS ensures easy timing closure for the generated RTL by exhaustively analyzing each path and scheduling operation so they fit in the given clock period.

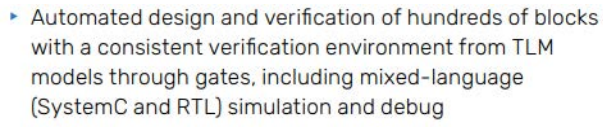
Stratus HLS uses patented datapath optimization technology and the embedded Genus synthesis to build all datapath components, multiplexers, and registers in the specified technology library to get accurate timing and area models.

The user can control how aggressively Stratus HLS packs these operations into each clock period. Creating designs with Stratus HLS can save months of back-end effort by preventing timing closure problems.

Integration with Genus physical synthesis allows early visibility and feedback into likely congestion problems, allowing the front-end designer to avoid problems in the back-end.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

51. The semiconductor design support device based on the Stratus HLS software includes a latency analyzer configured to analyze a result of a logic simulation performed on the RTL description to calculate a latency in each block representing an operation in a predetermined unit in the behavioral description.



- Automated design and verification of hundreds of blocks with a consistent verification environment from TLM models through gates, including mixed-language (SystemC and RTL) simulation and debug

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

Hierarchical Design

Stratus HLS is applicable to a single block or complex hierarchy of modules, including both HLS and RTL blocks. Stratus design and verification automation allows the designer to synthesize one, some, or all of the modules and do mixed SystemC and RTL simulation and verification.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

Moving from abstract MATLAB models to RTL descriptions has required manual conversion of MATLAB code into RTL. By definition, an RTL description expresses the cycle-accurate behavior. This step involves the design of a micro-architecture that accurately captures detailed cycle-by-cycle behavior and schedules operations among finite hardware resources to meet PPA goals in the implementation. To achieve optimal PPA, the micro-architectural solution space must be thoroughly explored—where the

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

Design Closure

Stratus HLS ensures easy timing closure for the generated RTL by exhaustively analyzing each path and scheduling operation so they fit in the given clock period.

Stratus HLS uses patented datapath optimization technology and the embedded Genus synthesis to build all datapath components, multiplexers, and registers in the specified technology library to get accurate timing and area models.

The user can control how aggressively Stratus HLS packs these operations into each clock period. Creating designs with Stratus HLS can save months of back-end effort by preventing timing closure problems.

Integration with Genus physical synthesis allows early visibility and feedback into likely congestion problems, allowing the front-end designer to avoid problems in the back-end.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

52. The Stratus HLS software includes the Genus Synthesis Solution engine.

- Genus logic synthesis and Joules power engines inside of Stratus HLS provide accurate timing, area, and power estimates

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

53. The Genus logic synthesis has timing awareness (including delay calculation) and as such is aware of the latency of operations in the behavioral description.

- Automatic extraction of full timing and physical contexts for any subset of a design. Reduces iterations between unit-level and chip-/block-level synthesis by 2X or more.

Source: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf

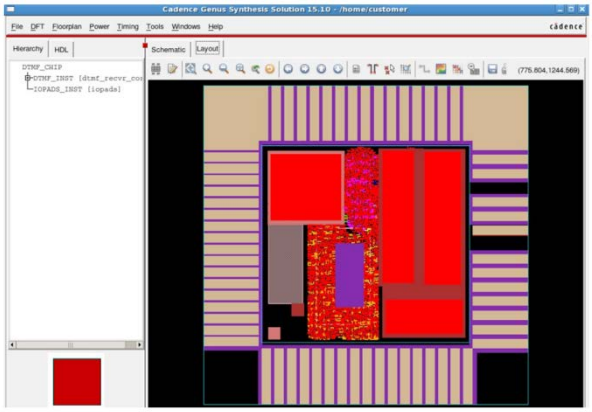
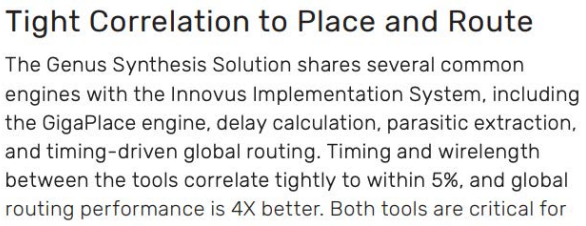


Figure 1: The Genus Synthesis Solution enables timing debug with physical interconnect knowledge built-in. Cross-probe to the physical viewer to see associated wirelengths, floorplan blockages, and estimated routing, and extract the chip-/block-level physical context for use in unit-level RTL design.

Source: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf

- Unified GigaPlace™ engine, delay calculation, parasitic extraction, and timing-driven global routing with Cadence Innovus™ Implementation System, timing and wirelength between the tools correlate to within 5%

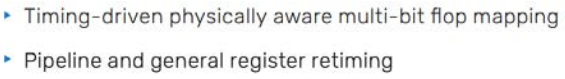
Source: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf



Tight Correlation to Place and Route

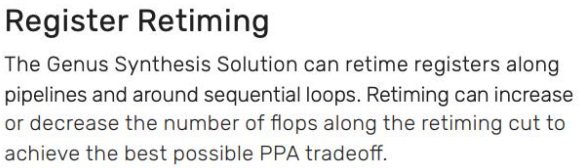
The Genus Synthesis Solution shares several common engines with the Innovus Implementation System, including the GigaPlace engine, delay calculation, parasitic extraction, and timing-driven global routing. Timing and wirelength between the tools correlate tightly to within 5%, and global routing performance is 4X better. Both tools are critical for

Source: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf



- Timing-driven physically aware multi-bit flop mapping
- Pipeline and general register retiming

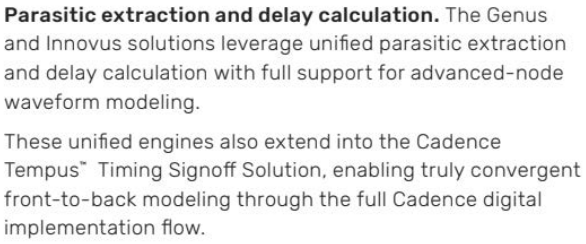
Source: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf



Register Retiming

The Genus Synthesis Solution can retime registers along pipelines and around sequential loops. Retiming can increase or decrease the number of flops along the retiming cut to achieve the best possible PPA tradeoff.

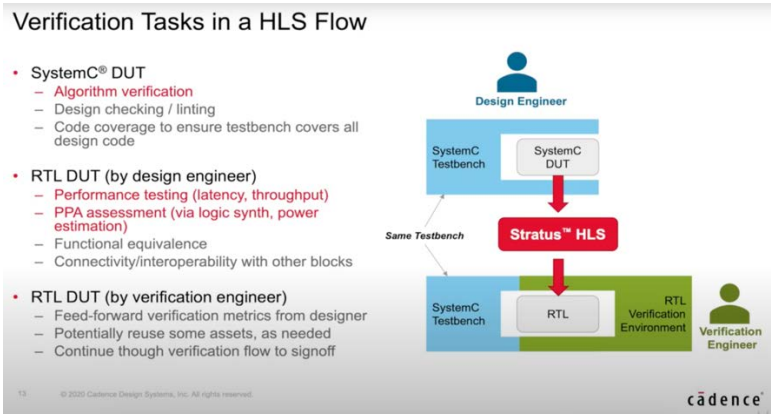
Source: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf



Parasitic extraction and delay calculation. The Genus and Innovus solutions leverage unified parasitic extraction and delay calculation with full support for advanced-node waveform modeling.

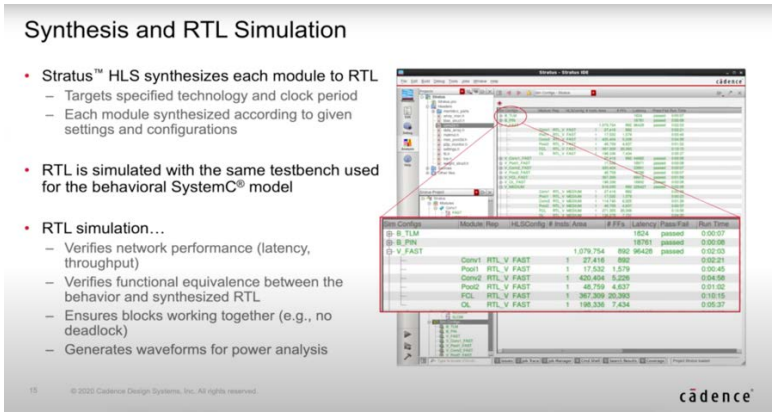
These unified engines also extend into the Cadence Tempus™ Timing Signoff Solution, enabling truly convergent front-to-back modeling through the full Cadence digital implementation flow.

Source: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/genus-product-brief-rebrand-v1.pdf

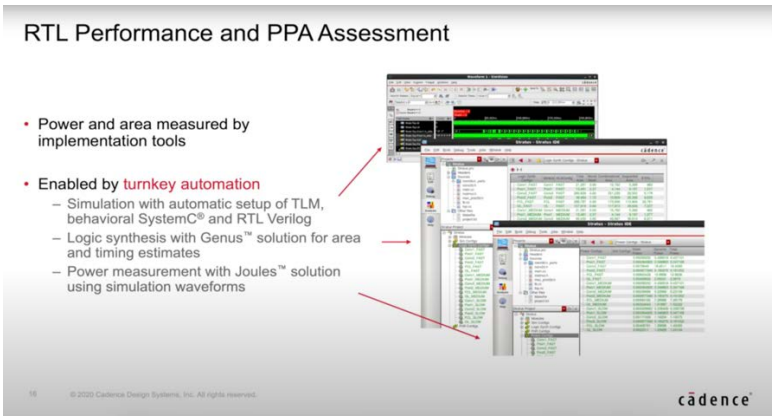


Source: https://www.youtube.com/watch?v=uxhIFYZ8iC0

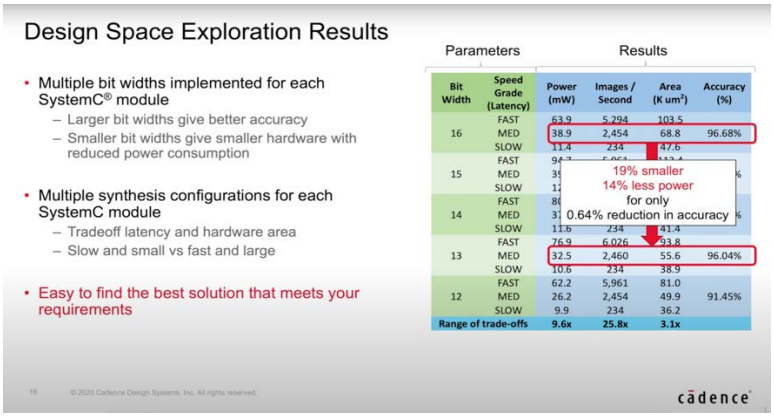
54. The Stratus HLS software GUI allows users to analyze individual modules synthesized to RTL and displays information including latency for each module.



Source: https://www.youtube.com/watch?v=uxhIFYZ8iC0



Source: https://www.youtube.com/watch?v=uxhIFYZ8iC0



Source: https://www.youtube.com/watch?v=uxhIFYZ8iC0

55. The semiconductor design support device based on the Stratus HLS software includes a correspondence table in which each block in the behavioral description corresponds to a state in the RTL description.

- Automated design and verification of hundreds of blocks with a consistent verification environment from TLM models through gates, including mixed-language (SystemC and RTL) simulation and debug

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

Design Closure

Stratus HLS ensures easy timing closure for the generated RTL by exhaustively analyzing each path and scheduling operation so they fit in the given clock period.

Stratus HLS uses patented datapath optimization technology and the embedded Genus synthesis to build all datapath components, multiplexers, and registers in the specified technology library to get accurate timing and area models.

The user can control how aggressively Stratus HLS packs these operations into each clock period. Creating designs with Stratus HLS can save months of back-end effort by preventing timing closure problems.

Integration with Genus physical synthesis allows early visibility and feedback into likely congestion problems, allowing the front-end designer to avoid problems in the back-end.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

56. The Stratus HLS software supports graphical analysis of the RTL with links to source code.

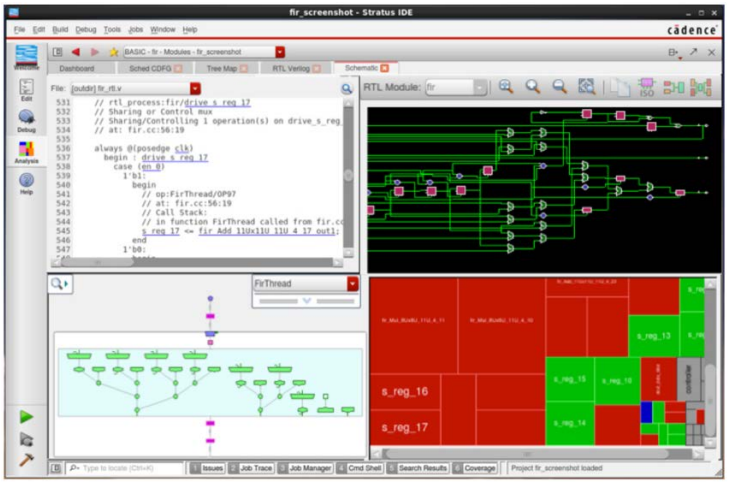
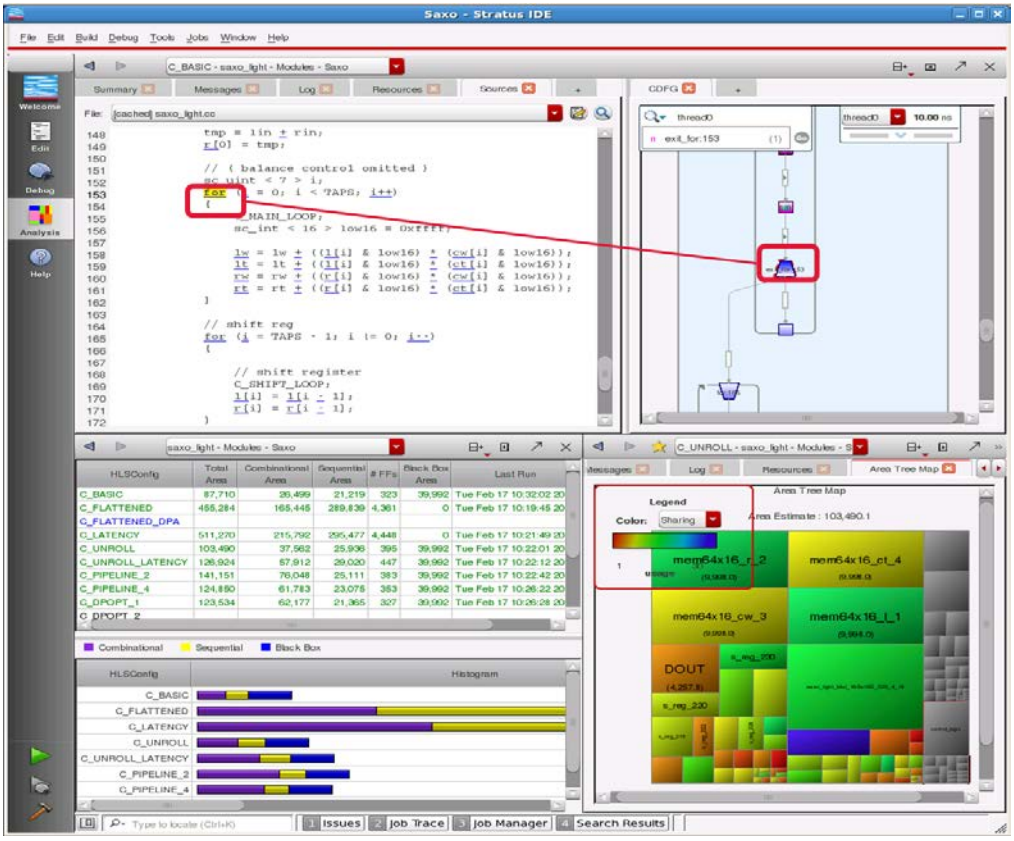


Figure 2: Complete graphical analysis with links to source code

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf



Source: https://www.linkedin.com/pulse/how-dramatically-reduce-time-from-architecture-spec-tapeout-laviv/

> Integrated with the Cadence verification suite, Stratus HLS supports automated mixed-language (SystemC and RTL) verification and debug including assertions, debugging, waveforms, and linkage back to the original SystemC design.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

57. The Stratus HLS software uses coverage databases to enable tracing RTL descriptions to the originating behavioral descriptions.

> By turning on the *rtl_annotation* feature in Stratus™ HLS, the uncovered RTL code lines are easily traced back to the originating behavioral SystemC® code lines. This saves weeks of verification effort by quickly identifying weaknesses in the TB and/or possible bugs in the DUT, and also by allowing developers to debug the involved logic on the higher abstraction behavioral/algorithmic SystemC® model.

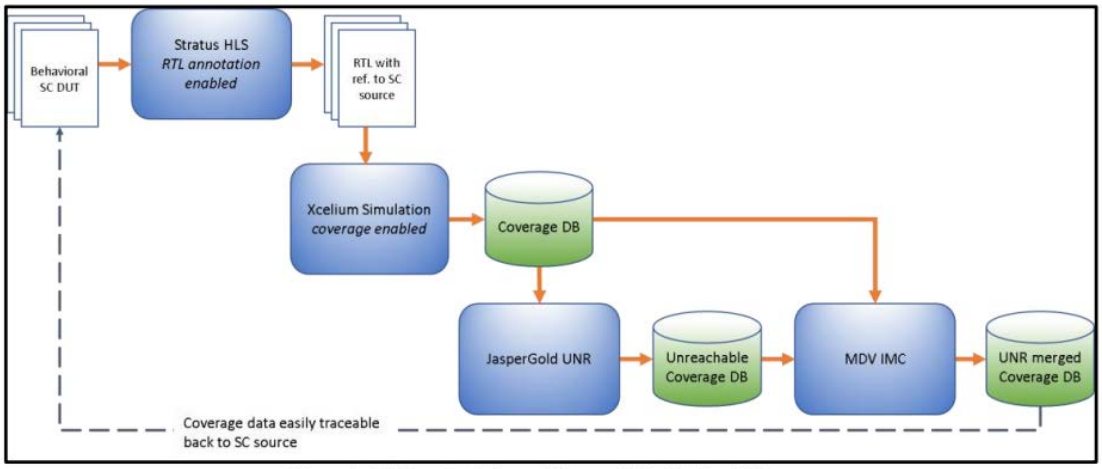


Figure 8: UNR analysis flow with traceability back to SC source

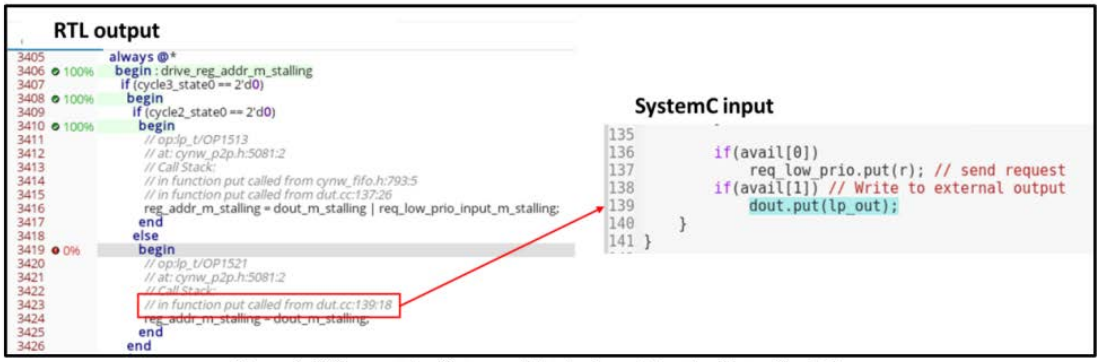


Figure 9: RTL annotation for traceability back to originating SystemC code lines

Source: S. Dahir, "Using HLS to improve Design-for-Verification of multi-pipeline designs with resource sharing," DVCON 2021 at 6-7, available at https://dvcon-proceedings.org/wp-content/uploads/using-hls-to-improve-design-for-verification-of-multi-pipeline-designs-with-resource-sharing.pdf.pdf.

58. The semiconductor design support device based on the Stratus HLS software includes a correspondence table generator configured to generate the correspondence table.

Design Closure

Stratus HLS ensures easy timing closure for the generated RTL by exhaustively analyzing each path and scheduling operation so they fit in the given clock period.

Stratus HLS uses patented datapath optimization technology and the embedded Genus synthesis to build all datapath components, multiplexers, and registers in the specified technology library to get accurate timing and area models.

The user can control how aggressively Stratus HLS packs these operations into each clock period. Creating designs with Stratus HLS can save months of back-end effort by preventing timing closure problems.

Integration with Genus physical synthesis allows early visibility and feedback into likely congestion problems, allowing the front-end designer to avoid problems in the back-end.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

59. The Stratus HLS software supports graphical analysis of the RTL with links to source code.

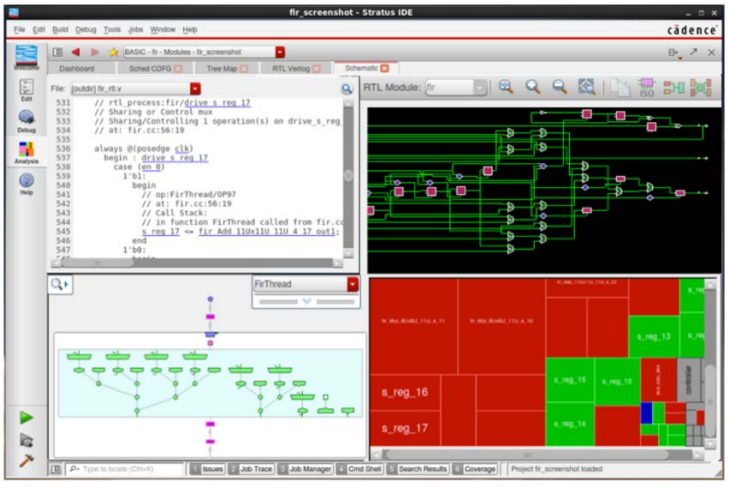
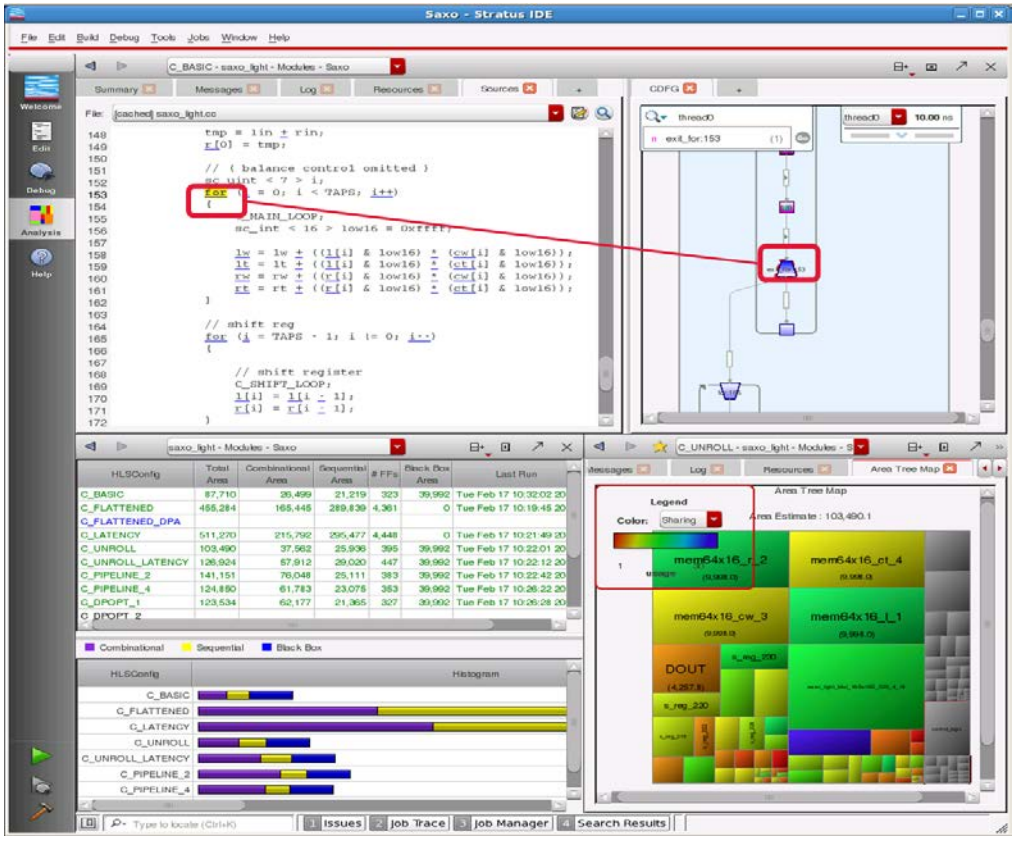


Figure 2: Complete graphical analysis with links to source code

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf



Source: https://www.linkedin.com/pulse/how-dramatically-reduce-time-from-architecture-spec-tapeout-laviv/

> Integrated with the Cadence verification suite, Stratus HLS supports automated mixed-language (SystemC and RTL) verification and debug including assertions, debugging, waveforms, and linkage back to the original SystemC design.

Source: https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

> By turning on the *rtl_annotation* feature in Stratus™ HLS, the uncovered RTL code lines are easily traced back to the originating behavioral SystemC® code lines. This saves weeks of verification effort by quickly identifying weaknesses in the TB and/or possible bugs in the DUT, and also by allowing developers to debug the involved logic on the higher abstraction behavioral/algorithmic SystemC® model.

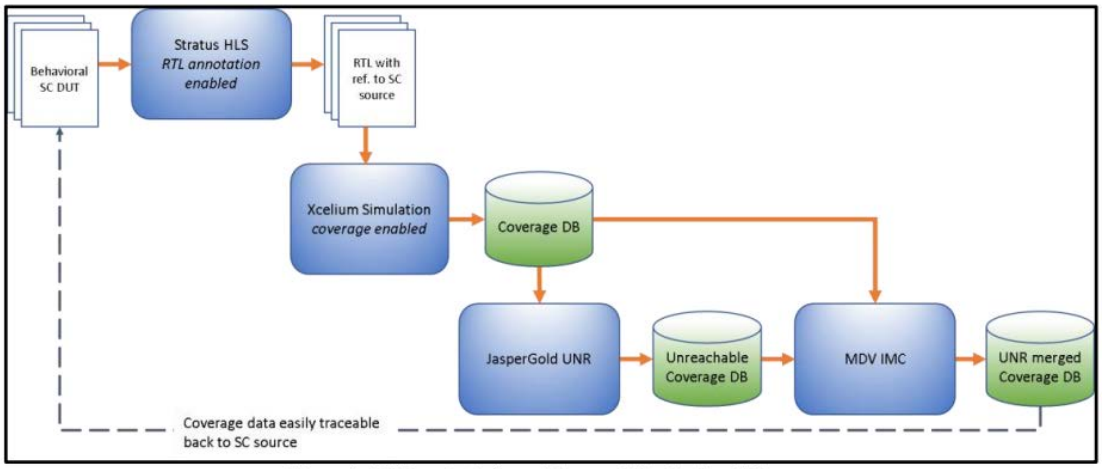


Figure 8: UNR analysis flow with traceability back to SC source

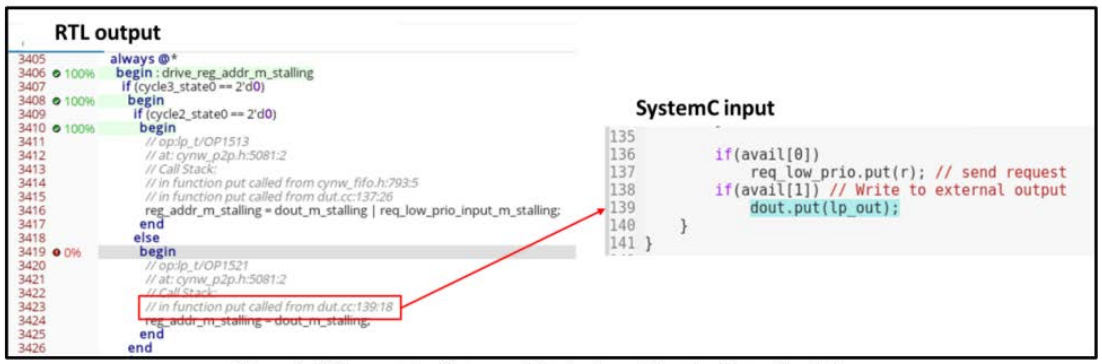


Figure 9: RTL annotation for traceability back to originating SystemC code lines

Source: S. Dahir, "Using HLS to improve Design-for-Verification of multi-pipeline designs with resource sharing," DVCON 2021 at 6-7, available at https://dvcon-proceedings.org/wp-content/uploads/using-hls-to-improve-design-for-verification-of-multi-pipeline-designs-with-resource-sharing.pdf.pdf.

60. Upon information and belief, Cadence has indirectly infringed and continues to indirectly infringe claim 1 of the '167 patent in violation of 35 U.S.C. §271(b). From at least the time Cadence received notice of its infringement, Cadence has induced others to infringe at least one claim of the '167 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Cadence's clients, customers, and end users, whose use of the Accused Products constitute direct infringement of at least one claim of the '167 patent. In particular, Cadence's actions that aided and abetted others such as customers and end users to infringe include advertising and distributing

the Accused Products, providing instruction materials, support training, and services regarding the Accused Products, and actively inducing its customers to acquire and/or install the infringing products, including Stratus HLS software, on customer-provided computer-readable media to be used in connection with a computer in an assertion-generating system for generating assertion descripts for validation of a semiconductor integrated circuit. *See, e.g.*, https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html; https://www.cadence.com/en_US/home/support.html, including all related domains and subdomains. Cadence does so knowing that its customers will commit these infringing acts. Despite its knowledge of the '167 patent, Cadence continues to make, use, sell, and/or offer for sale its infringing products thereby specifically intending for and inducing its customers to infringe the '167 patent.

61. Upon information and belief, Cadence has also indirectly infringed and continues to indirectly infringe at least claim 1 of the '167 patent in violation of 35 U.S.C. §271(c) by contributing to the infringement by its customers. Cadence sells or offers for sale in the United States the 167 Accused Products, including the Stratus HLS software, with knowledge that they are especially designed or adapted to operate in a manner that infringes that patent and despite the fact that the infringing technology or aspects of the 167 Accused Products are not a staple of commerce suitable for substantial non-infringing use. For example, Cadence is aware that the Stratus HLS software operates as described above and that such functionality infringes the '167 patent, including claim 1. Cadence is aware that the Stratus HLS software infringes when installed and/or used because it enables a computer to provide a semiconductor design support device for designing a semiconductor integrated circuit, that such installation and/or use infringes the '167 patent, including claim 1, and that the Accused Products have no substantial non-infringing use. The portion of the Stratus HLS software that maps to claim 1 (i.e., the infringing aspect) has no substantial non-infringing uses.

62. Cadence's infringement has damaged and continues to damage and injure Semiconductor Design.

63. Semiconductor Design is entitled to recover the damages sustained as a result of

Cadence's wrongful acts in an amount subject to proof at trial.

PRAYER FOR RELIEF

WHEREFORE, Plaintiff requests that the Court enter judgment for Plaintiff and against Defendant as follows:

a. That U.S. Patent No. 7,603,636 be judged valid, enforceable, and infringed by Defendant.

b. That U.S. Patent No. 7,971,167 be judged valid enforceable, and infringed by Defendant.

c. That Plaintiff be awarded judgment against Defendant for damages together with interests and costs fixed by the Court including an accounting of all infringements and/or damages not presented at trial; and

d. That Plaintiff be awarded such other and further relief as this Court may deem just and proper.

JURY DEMAND

Plaintiff respectfully requests a jury trial on all issues so triable.

Date: March 6, 2023

Respectfully submitted

/s/ *Robert F. Kramer*
Robert F. Kramer

KRAMER ALBERTI LIM & TONKOVICH LLP