Robert F. Kramer (SBN 181706)
rkramer@krameralberti.com
David Alberti (SBN 220625)
dalberti@krameralberti.com
Sal Lim (SBN 211836)
slim@krameralberti.com
Russell Tonkovich (SBN 233280)
rtonkovich@krameralberti.com
Robert C. Mattson (*pro hac vice*)
rmattson@krameralberti.com
James P. Barabas (*pro hac vice*)
jbarabas@krameralberti.com
Hong S. Lin (SBN 249898)
hlin@krameralberti.com
Robert Y. Xie (SBN 329126)
rxie@krameralberti.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 FRANCISCO DIVISION

SEMICONDUCTOR DESIGN TECHNOLOGIES, LLC, *Plaintiff,* v. CADENCE DESIGN SYSTEMS, INC., *Defendant.*	Case No. 3:23-cv-01001-AMO **FIRST AMENDED COMPLAINT FOR PATENT INFRINGEMENT** **DEMAND FOR JURY TRIAL**

FIRST AMENDED 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 First Amended 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 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 and analog design environment (collectively, "EDA") 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.

BACKGROUND

12. EDA tools are used by engineers to design electronic systems such as integrated circuits and printed circuit boards. This technology is discussed further in the Declaration of John Berg, a qualified expert in Electronic Design Automation (EDA) technology attached as Exhibit C ("Berg Decl.") and incorporated herein as though fully stated herein. Berg Decl. at ¶¶4-11.

13. The EDA software market in the United States is approximately $4 billion USD, by end-use, and growing. Cadence and other suppliers of EDA tools protect their innovations by, among other things, obtaining patents for their improvements to EDA software.

14. The process of circuit design is a series of hierarchical steps that include architectural design, logic design, physical design, physical verification, and sign-off. Berg Decl. at ¶14. It is common in the semiconductor industry to differentiate the engineers working at the different hierarchies. In the case of architectural design, the engineers are called architects or

architectural engineers. For logic/design, they are called design engineers or logic design engineers. The top-level circuit specification and behavioral design is specified by the architects, whereas the register-transfer level (RTL) and RTL validation are done by the design engineers. Berg Decl. at ¶20.

THE ASSERTED PATENTS & PATENTED TECHNOLOGY

15. 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 Mr. Takamitsu Yamada as the inventor. A true and correct copy of the ’636 patent is attached as Exhibit A.

16. Design verification, in particular register-transfer level validation, is an important step in the digital circuit design workflow as it allows a circuit designer to confirm that a circuit works as intended. Berg Decl. at ¶¶20, 22, 34. A chip design may fail verification in different ways, *e.g.*, (1) when there is a failure to identify a corner case in a large and complex circuit, (2) when there is a failure to understand shared interface specifications in a collaborative environment where different designers work on different blocks of a circuit, and (3) when errors are present in a specification for a third-party design. ’636 patent, col. 1:33-45.

17. One way to verify a circuit design is to utilize “assertion verification technology” in which assertions are used to test the behavior of the circuit. An assertion can be described as “a note in which intention (in some cases, this shows property) of designing is written and is generally written by a comment sentence in a RTL.” ’636 patent, col. 1:55-57; *see also* Berg Decl. at ¶¶36-40. “The intention is interpreted by a simulator during the verification, for example, an error log is generated in a case where a circuit to be verified operates differently from the intention.” ’636 patent, col. 1:57-60; *see also* Berg Decl. at ¶¶35, 38.

18. The use of assertions is beneficial because it reduces verification time, allows for errors to be caught earlier, focuses the design effort, and pinpoints sources of error. Berg Decl. at ¶¶43-47. At the time of the ’636 and ’167 patents, assertion practices varied widely among

organizations. Berg Decl. at ¶48. Those organizations who took assertions seriously, typically CPU companies, would have more than hundreds of thousands of assertions in a full design, including blocks. *Id.*

19. Property Specification Language ("PSL") is a popular assertion language promulgated by the Institute of Electrical and Electronics Engineers ("IEEE"). '636 patent, col. 2:24-30. Manual input of assertions is disadvantageous because it creates the "possibility that a mistake may be made in the assertion itself." '636 patent, col. 3:43-47; *see also* Berg Decl. at ¶¶49-50. As a result, the designer could be provided with an incorrect assertion violation, or a mistake in the design may be allowed to slip through. '636 patent, col. 3:47-50. Manual verification also negatively affects verification efficiency because the assertion description language will require error checking and debugging. '636 patent, col. 3:50-53.

20. The '636 patent also describes other known approaches of assertion verification as explained in Japanese Laid-Open Patent Application Nos. 2000-181933 and 2000-142918. '636 patent, col. 3:64-4:24. But these other approaches are disadvantageous. Berg Decl. at ¶¶51-52.

21. The JP '933 application describes an approach in which "a state transition machine is modeled by a graph and a state transition path that satisfies the assertion is searched." In this approach, "a model of the state transition machine … is formed manually, or the state transition machine is extracted from design data formed by a description language of a RTL." '636 patent, col. 4:4-11. A manually-input state machine model can include errors. Nor can the RTL description be relied upon to determine state transitions because it is the thing that is being verified. Berg Decl. at ¶51. The RTL may include an error about a state transition, and this error would be propagated to subsequent verification of the RTL.

22. The JP '918 application describes an approach to "generate[] an assertion for verifying a circuit by extracting a data transfer structure from design data of a RTL and expanding the data transfer structure into a graphic structure." '636 patent, col. 4:12-16. But this approach is also disadvantageous because the RTL is the thing to be verified, and an assertion extracted from the RTL may not match the expected behavior from the circuit specification. '636 patent, col. 4:16-19; *see also* Berg Decl. at ¶52.

23. The ’636 patent inventions improve upon these existing approaches, including manual input of assertions, generating assertions based on manual input of a state transition diagram, and generating assertions based on the RTL that is to be verified, and doing so using a technological approach to generating assertions that match the initial circuit specification.

24. 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. The ’636 patent also describes how to automatically generate properties from design data of a semiconductor integrated circuit, using a syntax analyzer and a property extractor. *E.g.*, ’636 patent, Fig. 2, col. 8:43-47, 9:53-10:19.

25. The assertion generating method is automated and contains more information to ensure, including: (1) the specification inputting step that generates design data of the semiconductor integrated circuit by graphically editing a specification of the semiconductor integrated circuit, (2) a property generating step that reads the design data generated at the specification inputting step from the storage and generates a property which verifies the specification of the semiconductor integrated circuit using the read design data and inputs the property in the storage, (3) the property generated by the property generating step is a selection condition with respect to a state transition, a logic value of at least one or more signals, or at least one or more signals in the design data, and (4) an assertion generating step that reads the property generated at the property generating step from the storage and converts the property into an assertion description. Berg Decl. at ¶59.

26. 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.

27. The invention recited in claim 8 is important because the use of the specification as an input uses the specification generated by the architecture engineers, whereas manual input of assertions is performed by the design engineers by interpreting the specification. Berg Decl. at ¶60. As a result, the actual specification and the interpreted specifications may not be identical. For a small chip (e.g. ~1000 transistors), the possibility of interpretation error is small, but for a large chip (e.g. ~500 million transistors), the possibility of error is nearly certain. Berg Decl. at ¶60. The ’636 invention forces the specification and the assertions to be consistent with one another by using the specification as an input into the property generation, i.e., assertion generation. Berg Decl. at ¶60. This method eliminates errors in communication between the architects and the design engineers. Berg Decl. at ¶60. Using the claimed framework, chip architects can create a high-level behavioral circuit and later design engineers can rely on assertion verification to ensure that the synthesized RTL accurately captures the desired specification of the chip design. Berg Decl. at ¶60.

28. Specifically, the ’636 patent describes assertion verification technology in which assertions are written directly by hand or generated by RTL, as well as the resulting problems of these techniques. ’636 patent, col. 1:46-4:31. The novel solution to those problems, among other elements, includes a property generating unit/step that automatically reads design data to generate properties which verify the specification of a semiconductor integrated circuit. The ability to read properties from a specification for purposes of automatically generating assertions was not known or conventional, especially when it was performed by a computer program stored on a computer-readable medium, which can then accurately determine properties for verification. Berg Decl. at ¶63. The ’636 patent enables a circuit designer to graphically edit the design data to generate properties and assertions in an automated and accurate manner, i.e., on-the-fly, which provides a technological benefit and a time-to-market benefit to companies whose engineers employ it. Berg Decl. at ¶68. Consequently, the ’636 patent avoids the mistakes and reduces the inefficiency of prior art approaches and enables computers to intelligently perform the bulk of the work in generating assertions for circuit design verification. Berg Decl. at ¶69.

29. This solution differs from hand-written assertions because hand-written assertions

must assume the design intention of the semiconductor integrated circuit. ’636 patent, col. 2:31-33. As a result, hand-written assertions cannot be relied upon, either for accuracy or complete functional coverage, to determine when a system no longer has to be revised. ’636 patent, col. 3:43-50. Further, hand-written assertions must themselves be verified and debugged. ’636 patent, col. 3:50-53. Simply looking at a specification to make assumptions about the design intent is not the same as using an automated tool to verify the specification of an integrated circuit. A human simply cannot achieve complete or accurate functional coverage of an integrated circuit, and thus cannot truly verify the circuit’s specification from assumptions about design intent.

30. The solution proposed by the ’636 patent also provides advantages over other automated techniques that generate assertions from the RTL description of a design because the RTL description itself may deviate from the design intent. Thus, if the assertions are generated from RTL, the assertions are not guaranteed to match the design intent. ’636 patent, col. 4:4-19, 4:29-31; *see also* Berg Decl. at ¶62.

31. In addition, the claims of the ’636 patent recite an unconventional and non-generic use of circuit specifications, properties, and computers to generate assertions, especially as compared to existing approaches at the time, i.e., the error-prone approaches to generate assertions noted in the specification. Berg Decl. at ¶69. The claim elements when considered together enable a streamlined framework to accurately generate assertions, and avoid the problems associated with existing approaches. Berg Decl. at ¶69.

32. Thus, 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. These approaches, which are described and claimed in the ’636 patent, are fundamentally different from and superior to both hand-written assertions, which require one to assume which assertions might cover the design intent, and other automated assertion-generating systems, which generate assertions from the RTL description and not from the design data of the specification for the semiconductor integrated circuit. Despite those differences, the generation of assertions from RTL descriptions, for example, is still performed by some commercial software packages today and is

not preempted by the claims of the ’636 patent.

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

34. 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.

35. The ’167 patent explains that “an LSI (large scale integration) circuit has come to have greater size and complexity.” ’167 patent, 1:27-28. The ’167 patent explains that “a life cycle of an electronic device including the LSI is becoming shorter.” *Id.*, 1:28-30. Accordingly, the ’167 patent explains that “a circuit design is requested to be completed in a shorter time period.” Col. 1:30-31. However, “known design methods may not design the LSI having the required greater size and complexity effectively.” ’167 patent, col. 1:31-33. Accordingly, a technological need existed: “various EDA (electronic design automation) tools for describing a design at an increased abstract level are proposed.” ’167 patent, col. 1:33-35. Rather than design circuits at the transistor level, digital circuit designers use computer tools to design circuits at higher levels of abstraction, *e.g.*, a behavioral level. Berg Decl. at ¶71. The circuit designer defines a behavioral model (or description) that describes how the circuit is to function. A behavioral synthesis tool can convert the behavioral description to a register transfer level (RTL) description. ’167 patent, col. 1:36-39. The behavioral description is not aware of timing or clock cycles. ’167 patent, col. 1:40-43. The RTL description is an intermediate level abstraction that includes registers and combinational logic. The RTL description operates according to a clock, and as such is aware of timing. ’167 patent, col. 1:43-45. The RTL description can be simulated in software to determine the overall latency of the RTL circuit. ’167 patent, col. 1:48-55.

36. But prior to the ’167 invention, there was no effective way to determine the portion(s) of the corresponding behavioral description responsible for high latency in the RTL

description. For example, if a simulation of an RTL description reveals high latency, the designer cannot easily determine which part(s) of the corresponding behavioral description was responsible for the high latency. Berg Decl. at ¶72. The designer must manually revisit the behavioral description, only knowing that the synthesized RTL had high latency resulting from some unknown portion of the behavioral description. Berg Decl. at ¶72. Alternatively, the designer can attempt to diagnose the latency directly in RTL, but then the behavioral and RTL descriptions become untethered, which means the designer cannot guarantee the RTL description accurately reflects the behavioral description. Berg Decl. at ¶72.

37. The '167 patent proposes a specific improvement to a semiconductor design support device that allows a circuit designer to determine the latency of specific blocks of the behavioral description, e.g., to pinpoint specific portions of a behavioral description associated with high latency. Berg Decl. at ¶78. At the heart of claim 1 is a "latency analyzer" to check the latency based on an RTL simulation and a "correspondence table generator" that maps blocks in the behavioral description to states in the RTL description. Berg Decl. at ¶¶76-78. Relying on the correspondence table generator, the latency analyzer can determine the latency of behavioral blocks that are associated with states in the RTL description, providing new and improved functionality to EDA tools that were previously unavailable. Berg Decl. at ¶83. The specific use of the latency analyzer with the correspondence table generator amounts to a non-conventional and non-generic combination of elements that grants circuit designers greater insight into the delays caused by their behavioral designs to enable them to easily focus their efforts on improving the high-latency portions of their designs. Berg Decl. at ¶¶72-74.

38. 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. For example, the '167 patent describes and claims the use of a correspondence table that maps the states in the RTL description to the blocks in the behavioral description and the use of a latency analyzer that calculates the total latency in each block based on the relationships between the blocks and the states in the RTL description in

the correspondence table. ’167 patent, col. 5:26-32. During prosecution, the Patent Office acknowledged the novelty and non-obviousness of the correspondence table over the prior art when it indicated that pending claims 2 and 8 (which issued as claims 1 and 3, respectively), among others, contained allowable subject matter.

39. 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 to determine the latency of each block from the RTL simulation. The ’167 patent does not preempt behavioral synthesis, or analysis of a resulting RTL description. The claims require much more than that (i.e., latency analysis on a block-by-block basis corresponding to the behavioral description), and the use of a correspondence table generator to determine the correspondence between of different portions (e.g., blocks and/or states) of the behavioral and RTL descriptions. As a result, the ordered combination of elements provides an improved semiconductor design support tool that enables a designer to determine the latencies associated with different blocks of a behavioral description. Providing an improvement to an existing device and/or process would not preempt the existing device and/or process. 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.

40. In addition, the claims of the ’167 patent are directed to non-conventional and non-generic uses of elements (e.g., a correspondence table generator and latency analyzer). The combination of a correspondence table generator and a latency analyzer to allow granular analysis of latency corresponding to blocks within a behavioral description was unconventional at the time of the ’167 patent because it is a departure from the approach used at the time (e.g., manually identifying high latency blocks in the behavioral description). Berg Decl. at ¶84. The combination of these two elements was also non-generic because when they are paired together in the manner claimed in claim 1, they allow designers to have greater insight into timing issues associated with the behavioral description than was available at the time. Berg Decl. at ¶84.

41. 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.

42. The Asserted Patents are valid and enforceable.

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

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

44. 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.

45. Upon information and belief, Cadence has directly infringed and continues to directly infringe at least, for example, claim 8 of the '636 patent by making, using, selling, and/or offering for sale 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.

46. 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.

47. Upon information and belief, Cadence had full knowledge of the '636 patent, for example, based upon Cadence's receipt of various letters sent by Ricoh Company, Ltd., the former owner of the '636 patent, to Cadence senior personnel including letters sent to Cadence on

November 2, 2020 and December 11, 2020 specifically informing Cadence of the ’636 patent and its applicability to Cadence’s EDA business and products.

48. In addition, Cadence had knowledge of the ’636 patent, for example by way of this patent having been cited in Cadence’s own U.S. patents, including for example the following Cadence patents: U.S. Patent Nos. 7,712,060, 7,810,056, 9,842,183, and 10,922,469, which are attached as Exhibits D, E, F, and G, respectively. Accordingly, upon information and belief, prior to the filing of this lawsuit, Cadence had actual knowledge of the ’636 patent and the infringing nature of its products.

49. Cadence’s own U.S. patents are instructive both with respect to its sworn representations to the United States Patent Office as to the patentability of EDA technologies and as to Cadence’s knowledge of Semiconductor Design’s ’636 patent.

50. The Cadence ’060 patent is titled “Method and system for handling assertion libraries in functional verification” and issued on May 4, 2010. Its Abstract states:

> A method and system for handling assertion libraries in verification of a design are disclosed. The method and system include structuring and implementing at least one verification component in at least one of the assertion libraries with at least one standard assertion language supported by at least one verification tool, creating an assertion library element for a specific requirement for verification of the design without dependence on the at least one verification tool for the assertion library element, and resolving assertion status. With the disclosed method and system, visualization of assertion status at various levels of design hierarchy and at verification component level may be achieved, and implementing verification techniques may include optimization techniques during and/or after verification.

51. Claim 1 of the Cadence ’060 patent recites:

> 1. A computer-implemented method for handling assertion libraries in verification of a design, the method comprising:
>
> structuring and implementing at least one verification component for the verification of the design capable of being stored in at least one of the assertion libraries with at least one assertion language supported by at least one verification tool;
>
> creating an assertion library element for a specific requirement for the verification of the design without dependence on the at least one verification tool for the assertion library element;
>
> resolving assertion status by consolidating a set of assertion statuses during or after verification when doing assertion based verification (ABV) using the verification component and/or assertion, wherein the act of resolving assertion status is performed by using a processor; and

> storing the assertion status in a volatile or non-volatile computer readable medium or displaying the assertion status on a display device.

52. The Cadence '056 patent is titled "Method and system for implementing context aware synthesis of assertions" and issued October 5, 2010. Its Abstract states:

> A method and system for implementing context aware synthesis of assertions is disclosed. The method and system for assertion synthesis includes converting an assertion formula to sequence implication form using semantic preserving rewrite rules, performing optimizations on the resulting formula to reduce the number of state-bits in a final FSM (Finite State Machine), and synthesizing the resulting formula to the final FSM using context aware sequence synthesis.

53. Claim 1 of the Cadence '056 patent recites:

> 1. A computer implemented method for assertion synthesis of circuitry comprising:
>
> using at least one computer system which comprises at least one processor and is programmed for performing:
>
> converting an assertion formula to a sequence implication form using one or more semantic preserving rewrite rules, wherein
>
> the sequence implication form converted by the one or more semantic preserving rewrite rules is semantically equivalent to the assertion formula;
>
> optimizing a resulting formula, which is converted from the assertion formula, to reduce a number of states in a state machine that represents the circuitry;
>
> synthesizing the resulting formula to the state machine using context aware sequence synthesis, wherein
>
> the context aware sequence synthesis synthesizes the resulting formula based at least in part upon context of one or more sequences in the resulting formula; and verifying the circuitry using the state machine.

54. The Cadence '183 patent is titled "Methods and systems for enabling concurrent editing of electronic circuit layouts" and issued December 12, 2017. Its Abstract states:

> Methods and systems of an electronic circuit design system described herein provide a new layout editor tool to make edits in an electronic circuit layout. A plurality of partitions is created in the electronic circuit layout. The new layout editor tool enables multiple electronic circuit designers to edit a different partition of the plurality of partitions of the same electronic circuit layout at the same time and save the edited partition locally.

55. Claim 1 of the Cadence '183 patent recites:

> 1. A processor-implemented method for modifying an electronic circuit layout, comprising:
>
> generating, by a processor, a plurality of partitions in an electronic circuit layout, wherein each of the plurality of partitions contains at least a portion of the electronic

> circuit layout that is independent from the portions of the electronic circuit layout in the other partitions of the plurality of partitions;
>
> generating, by the processor, an empty delta view for each of the plurality of partitions, wherein the empty delta view is a cell view corresponding to a record in a cell database, wherein the empty delta view is configured to be checked in and checked out locally;
>
> concurrently rendering, by the processor, one or more design layout interfaces each showing at least one partition of the plurality of partitions, wherein the at least one partition is configured to be edited using the respective design layout interface;
>
> while receiving one or more concurrent modification instructions to edit the respective partitions configured to be edited from the one or more design layout interfaces:
>
> modifying, by the processor, the respective empty delta view associated with the respective partition configured to be edited, according to the respective modification instructions received from the respective design layout interface to generate a respective modified delta view from the respective empty delta view, wherein the modified delta view is configured to checked in and checked out locally; and
>
> updating, by the processor, each respective partition according to the respective modified delta view, thereby allowing parallel editing of the electronic circuit layout by a plurality of electronic circuit designers.

56. The Cadence '469 patent is titled "Methods and systems of enabling concurrent editing of hierarchical electronic circuit layouts" and issued February 16, 2021. Its Abstract states:

> Embodiments described herein provide a new layout editor tool allowing designers to concurrently edit various aspects of an electronic circuit layout, even at disparate hierarchical levels of the design. The new layout editor tool enables multiple electronic circuit designers to concurrently edit a layout a different hierarchical levels, by logically establishing editable child sub cell-level partitions within a parent layout-level partition, each of which representing various components of the same electronic circuit layout.

57. Claim 1 of the Cadence '469 patent recites:

> 1. A processor-implemented method for at least two computers to concurrently create an electronic circuit layout, the method comprising:
>
> generating, by a processor, a plurality of top-level partitions from a circuit layout according to a predefined attribute, each respective top-level partition corresponding to a portion of the electronic circuit layout at a top level stored as a first database record including the predefined attribute;
>
> generating, by the processor, a plurality of top-level delta views corresponding respectively to each top-level partition, wherein each respective top-level delta view is a cell view configured to store edits to the respective top-level partition as a second database record;
>
> in response to receiving, from a client computer having access rights to the top-level partition, a command to modify the cell view from the top-level partition:

> modifying, by the processor, the second database record corresponding to the top-level delta view, thereby resulting in a modified second database record containing a modified top-level delta view;
>
> in response to receiving, from the client computer, a command to modify a subcell view from a top-level partition:
>
> generating, by the processor, a plurality of sub-level partitions based upon the predefined attribute, each respective sub-level partition comprising the predefined attribute inherited from the top-level partition that corresponds to the portion of the electronic circuit layout at a sub level stored as a third database record including the predefined attribute;
>
> generating, by the processor, a plurality of sub-level delta views corresponding respectively to each sub-level partition, wherein each respective sub-level delta view is a subcell view configured to store edits to the respective sub-level partition as a fourth database record; and
>
> modifying, by the processor, the fourth database record corresponding to the sub-level delta view according to the command, thereby resulting in a modified fourth database record containing a modified sub-level delta view; and
>
> displaying, by the processor, a representation of the modified sub-level delta view to the client computer having the access rights,
>
> wherein the first database record and the third database record remain unmodified by the commands.

58. These Cadence patents are directed to analogous subject matter as Plaintiff's '636 and '167 patents, and together they depict the landscape for patent-eligible improvements to EDA technology. In the course of applying for its EDA patents, during prosecution of the '183 patent, Cadence provided a lengthy explanation to the Patent Office under oath as to why EDA inventions are eligible for patenting under Section 101. In fact, to persuade the Patent Office to issue patent claims for Cadence's own EDA inventions analogous to Plaintiff's '636 and '167 patents, Cadence represented to the Patent Office under oath that its EDA inventions are not simply directed to an abstract idea of "employing steps of manipulating with data" and why its claims are directed to "a specific solution to the technological problem of concurrently editing and modifying layouts of highly complex integrated circuits (ICs) using an electronic design automation (EDA) tool." *See* Office Action dated March 15, 2017 in Appl. No. 14/869,505; Response to Non-Final Action dated June 15, 2017 in Appl. No. 14/869,505 at 7-12, copies of which are attached as Exhibits H and I, respectively. The PTO agreed with these arguments. *See* Notice of Allowability in Appl. No. 14/869,505 at 2, a copy of which is attached as Exhibit J. Similarly, Cadence also overcame a

Section 101 rejection during prosecution of the '060 patent. *See* Office Action dated May 29, 2009 in Appl. 11/712,003 at 2-3; Response to Office Action dated August 31, 2009 in Appl. 11/712,003 at 2, 7; Notice of Allowance dated December 18, 2009 in Appl. 11/712,003, copies of which are attached as Exhibits K, L, and M. The same representations and arguments proffered by Cadence under oath to the Patent Office in support of its own inventions being eligible for patenting under Section 101 likewise support the patent eligibility of other patents in the same technological area, such as Plaintiff's '636 and '167 patents.

59. The following paragraphs demonstrate how the Stratus HLS software infringes at least claim 8 of the '636 patent. Thus, when Cadence or its customers make, use, sell, or offer to sell the Stratus HLS software, they directly infringe at least claim 8 of the '636 patent.

60. 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).

61. 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).

62. 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).

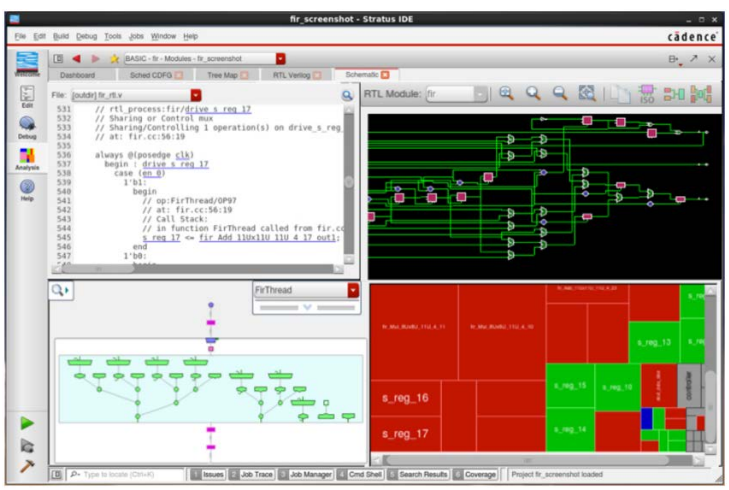
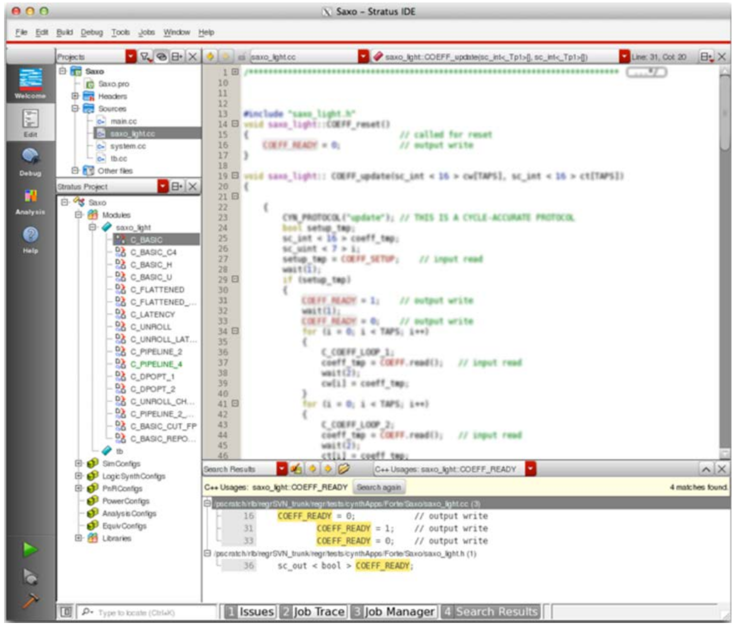
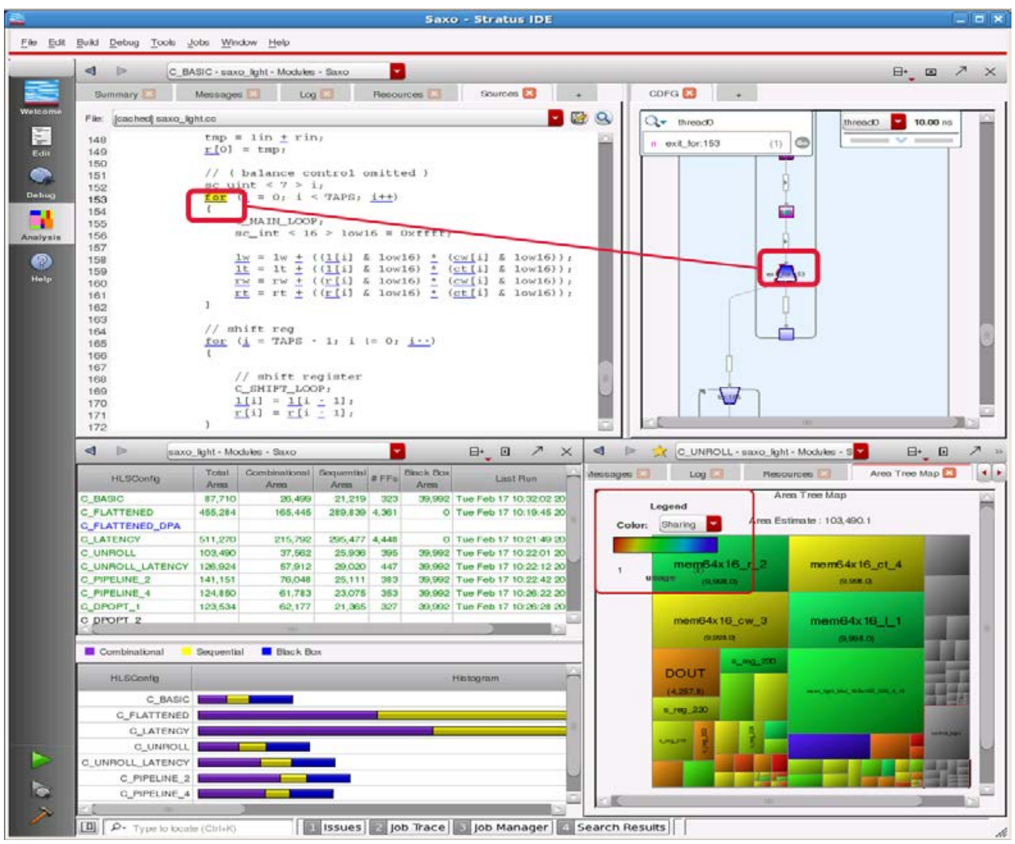


Figure 2: Complete graphical analysis with links to source code

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

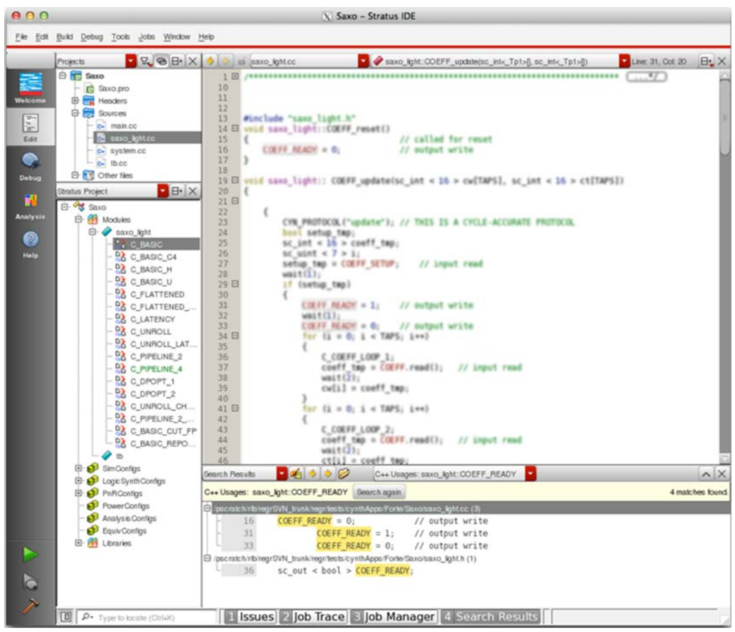


Source: https://www.linkedin.com/pulse/how-dramatically-reduce-time-from-architecture-spec-tapeout-laviv/

> 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

63. 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

64. 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.

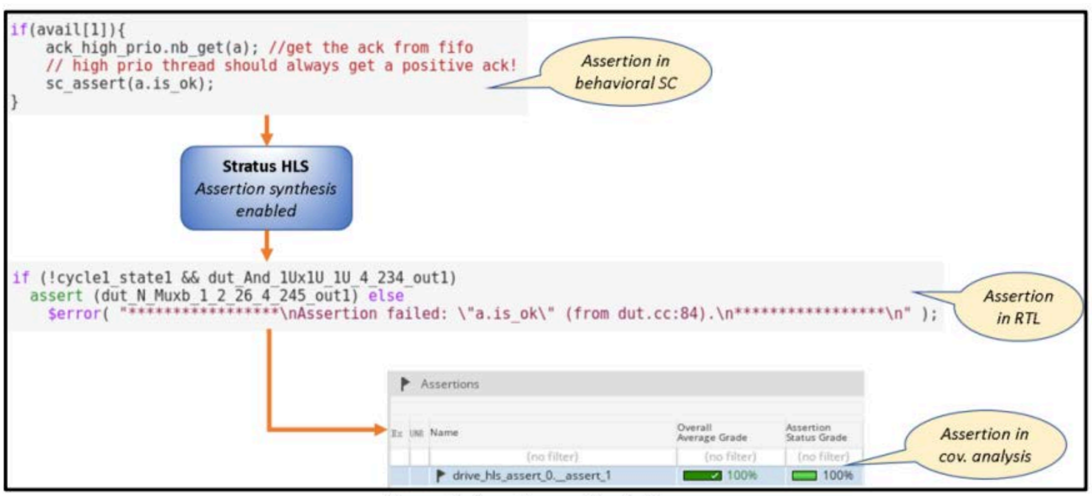


Figure 5: Assertion synthesis flow

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.

65. 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.

> **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.

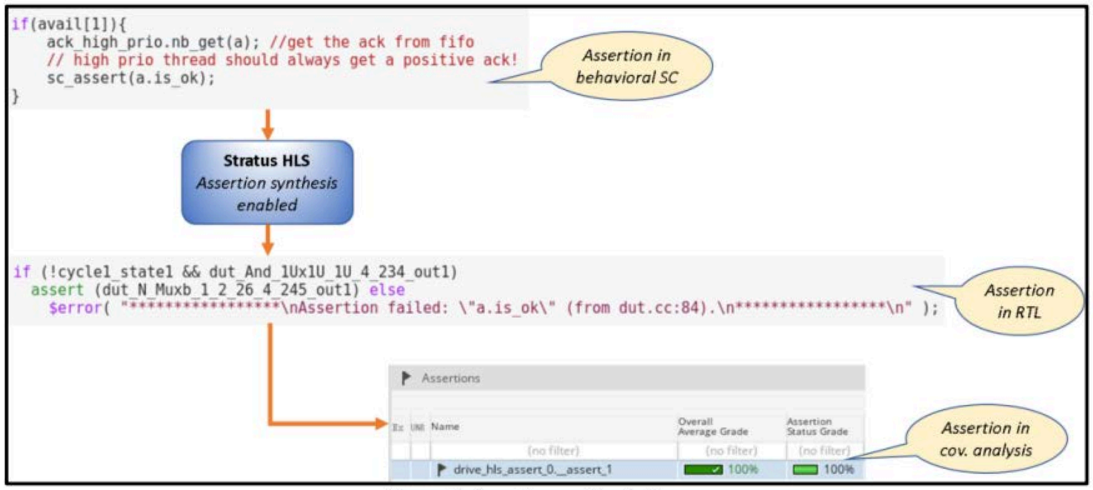


Figure 5: Assertion synthesis flow

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.

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

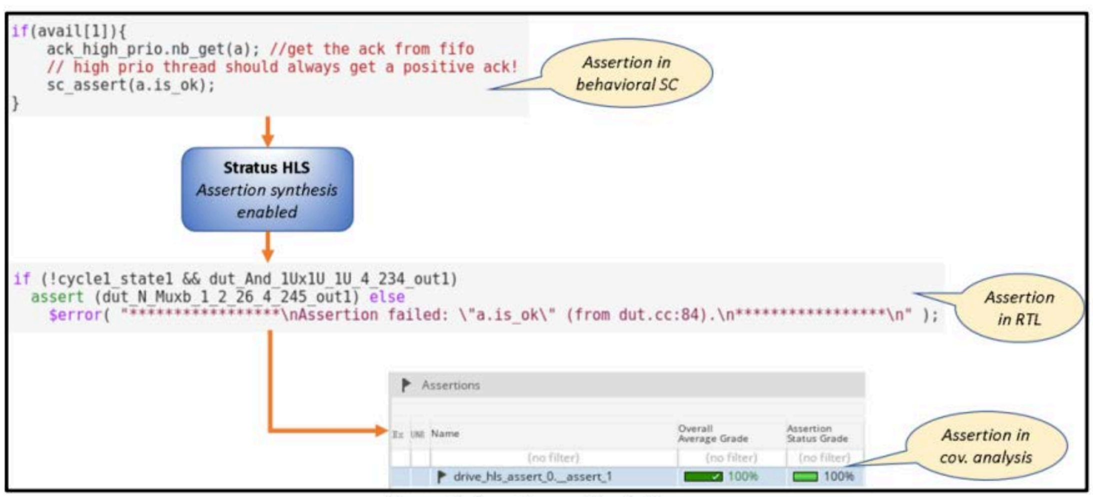
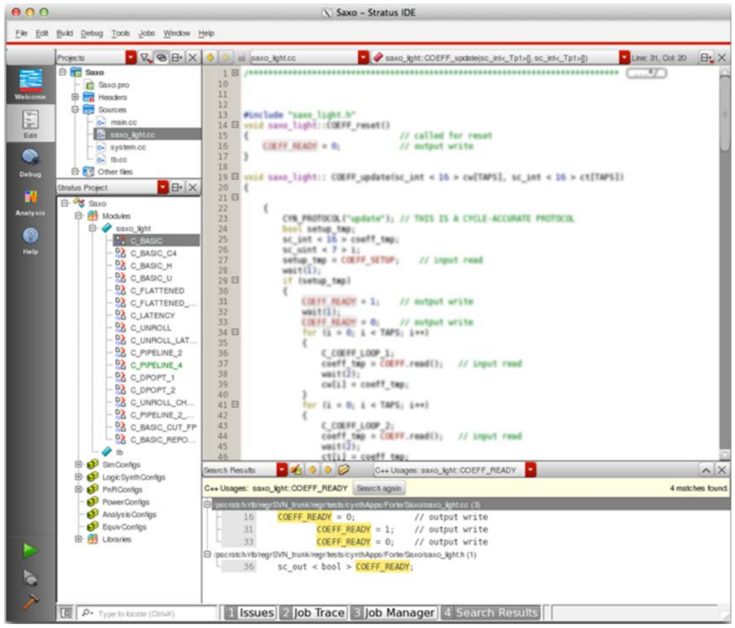


Figure 5: Assertion synthesis flow

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

67. 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.

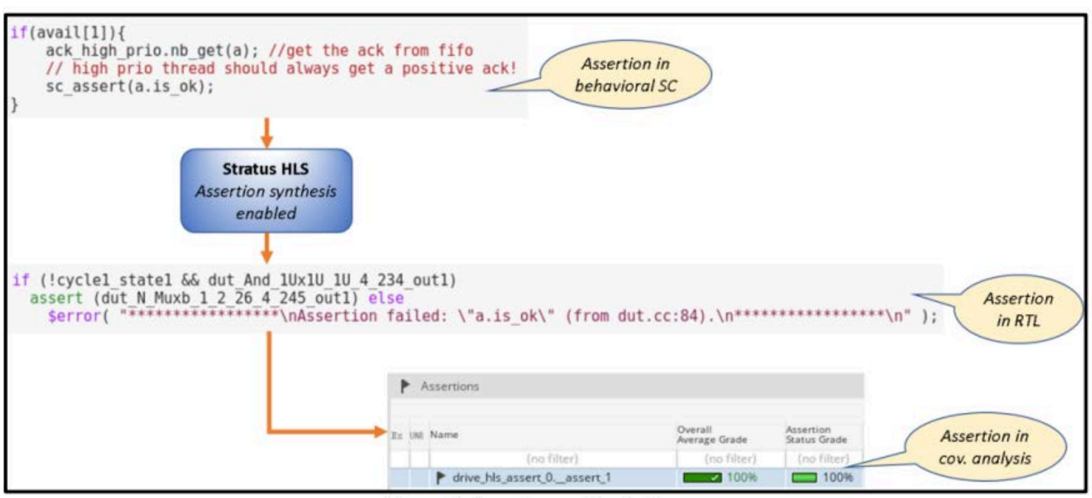


Figure 5: Assertion synthesis flow

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

68. 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.

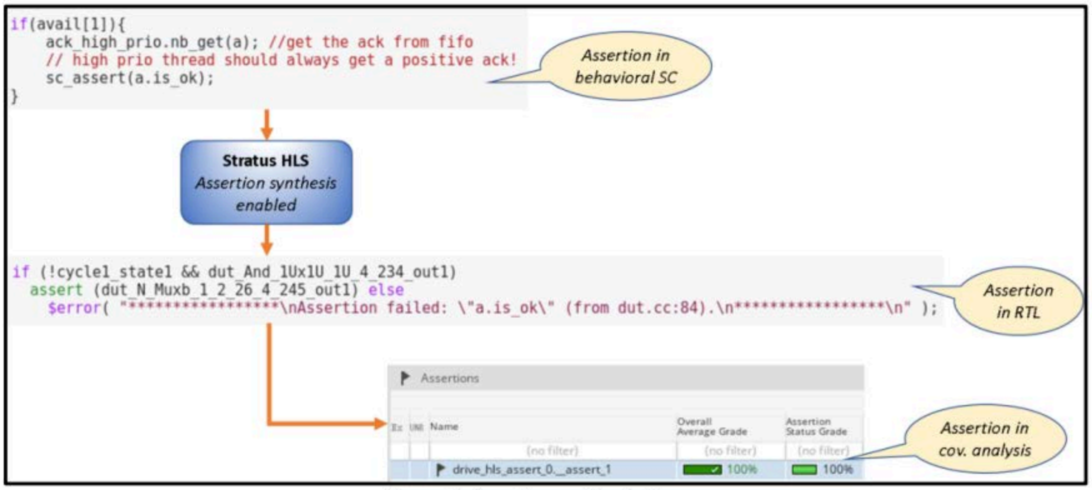


Figure 5: Assertion synthesis flow

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.

69. 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.

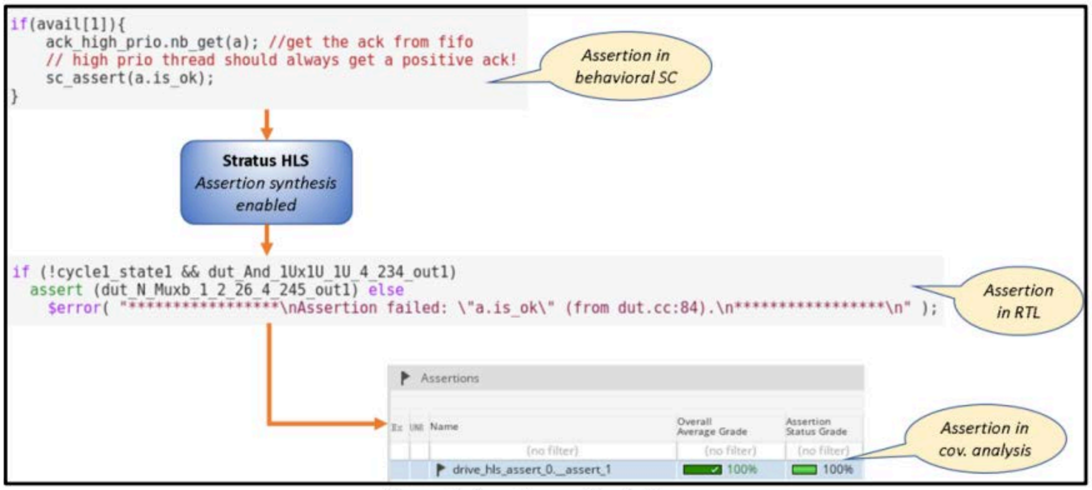


Figure 5: Assertion synthesis flow

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.

70. 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 the '636 patent , Cadence has induced others to infringe at least claim 8 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. Those customers include Intel, Qualcomm, Socionext, Syntiant, Himax, and Methods2Business. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html.

71. 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. Those customers include Intel, Qualcomm, Socionext, Syntiant, Himax, and Methods2Business.
https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html. Cadence sells or offers for sale in the United States the 636 Accused Products, including for example 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 knew 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.

72. Cadence’s infringement has damaged and continues to damage and injure Semiconductor Design.

73. 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

74. Semiconductor Design incorporates by reference the allegations contained in paragraphs 1 to 74 above.

75. 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.

76. Upon information and belief, Cadence has directly infringed and continues to directly infringe at least, for example, 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.

77. Upon information and belief, Ricoh Company, Ltd. put Cadence on notice that it infringed or potentially infringed the '167 patent prior to the filing of the original Complaint and/or invited Cadence to enter into a license under the '167 patent prior to the filing of the original Complaint. Thus, prior to the filing of this lawsuit, Cadence had actual knowledge of the '167 patent and the infringing nature of its products.

78. Upon information and belief, Cadence had full knowledge of the '167 patent, for example, based upon Cadence's receipt of various letters sent by Ricoh Company, Ltd., the former owner of the '167 patent, to Cadence senior personnel including letters sent to Cadence on

November 2, 2020 and December 11, 2020 specifically informing Cadence of the ’167 patent and its applicability to Cadence’s EDA business and products.

79. The following paragraphs demonstrate how the Stratus HLS software infringes at least claim 1 of the ’167 patent. Thus, when Cadence or its customers make, use, sell, or offer to sell the Stratus HLS software, they directly infringe at least claim 1 of the ’167 patent.

80. A computer installed with the Stratus HLS software is a semiconductor design support device for designing a semiconductor integrated device. Stratus HLS software 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

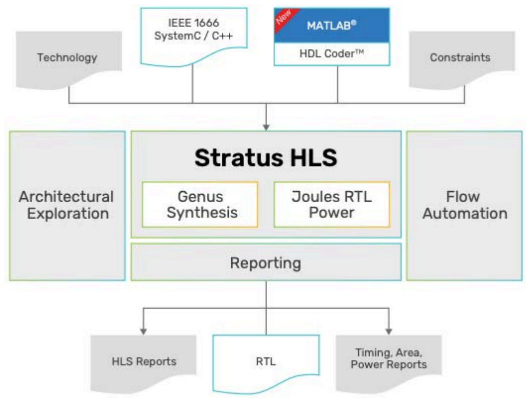


Figure 1: Stratus HLS uses the Genus synthesis and Joules power engines to create high-quality RTL targeted to your technology and design constraints

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

81. 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 flexi-bility 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

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

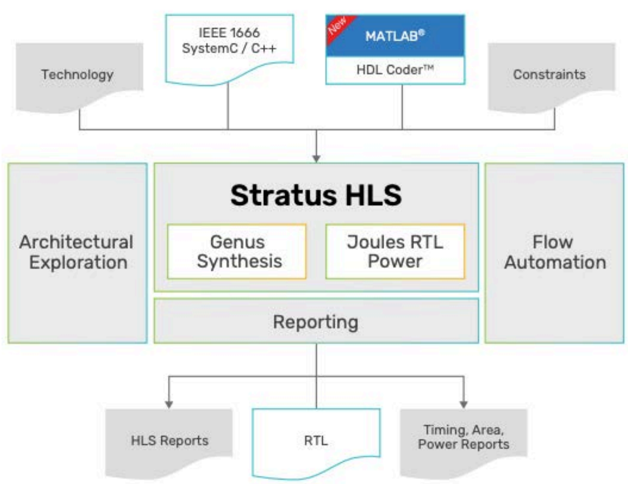


Figure 1: Stratus HLS uses the Genus synthesis and Joules power engines to create high-quality RTL targeted to your technology and design constraints

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

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

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.

84. 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

85. 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

86. 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

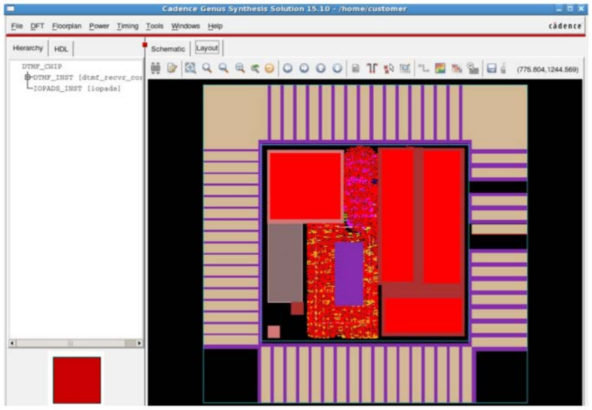


Figure 1: The Genus Synthesis Solution enables timing debug with physical interconnect knowledge built-in. Cross-probe to the physical viewer to see associated wirelengths, floorplan blockages, and estimated routing, and extract the chip-/block-level physical context for use in unit-level RTL design.

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-

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

www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf

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

Register Retiming

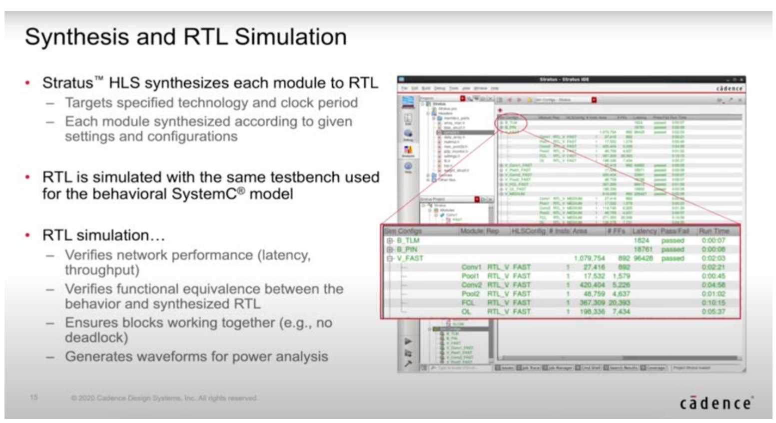
The Genus Synthesis Solution can retime registers along pipelines and around sequential loops. Retiming can increase or decrease the number of flops along the retiming cut to achieve the best possible PPA tradeoff.

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.

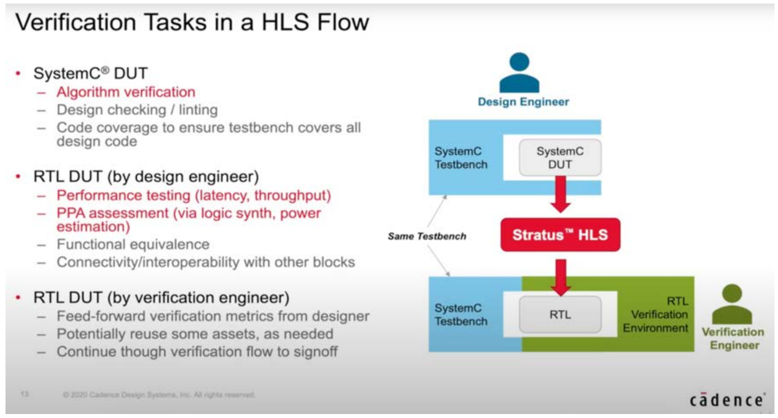
These unified engines also extend into the Cadence Tempus™ Timing Signoff Solution, enabling truly convergent front-to-back modeling through the full Cadence digital implementation flow.

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

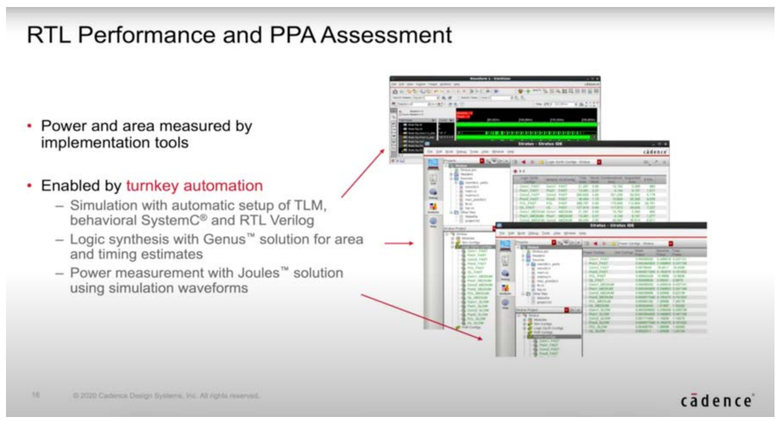


Source: https://www.youtube.com/watch?v=uxhIFYZ8iC0

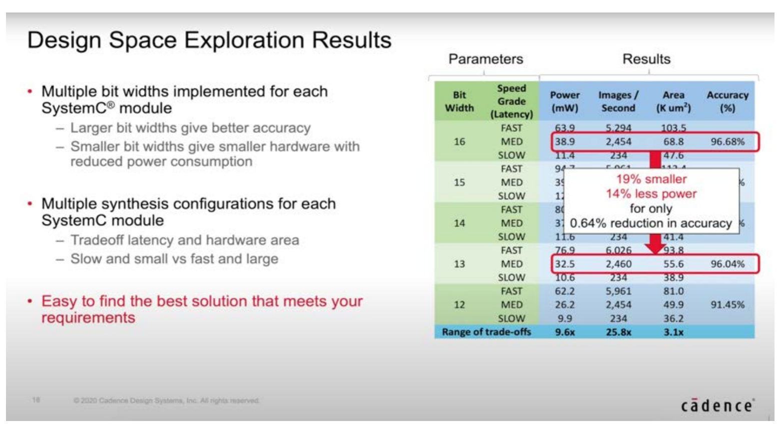
87. The Stratus HLS software GUI allows users to analyze individual modules synthesized to RTL and displays information including latency for each module.



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



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



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

88. 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

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

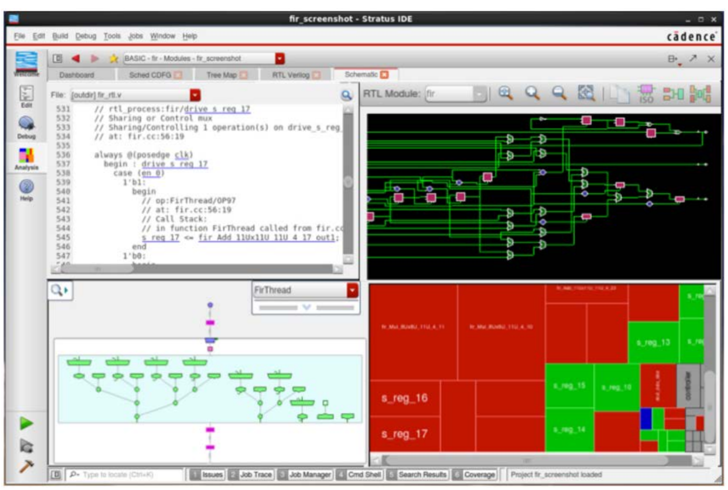
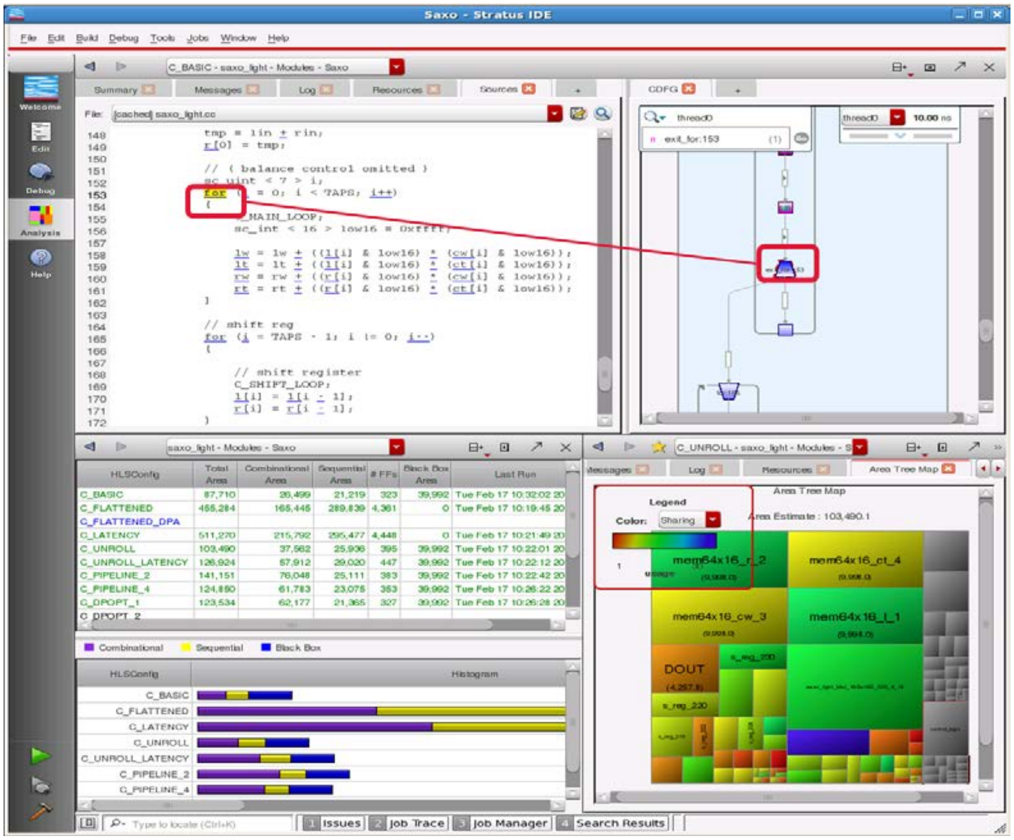


Figure 2: Complete graphical analysis with links to source code

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

90. 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.

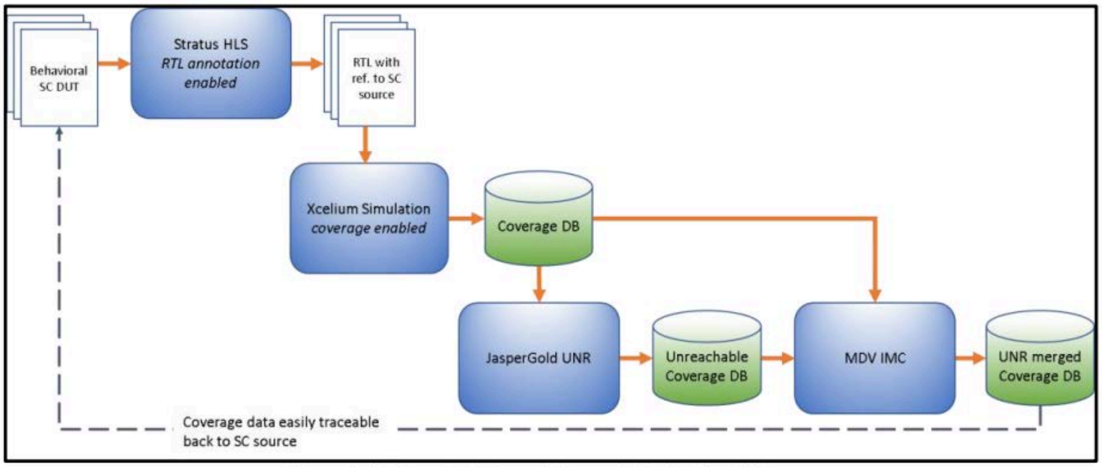


Figure 8: UNR analysis flow with traceability back to SC source

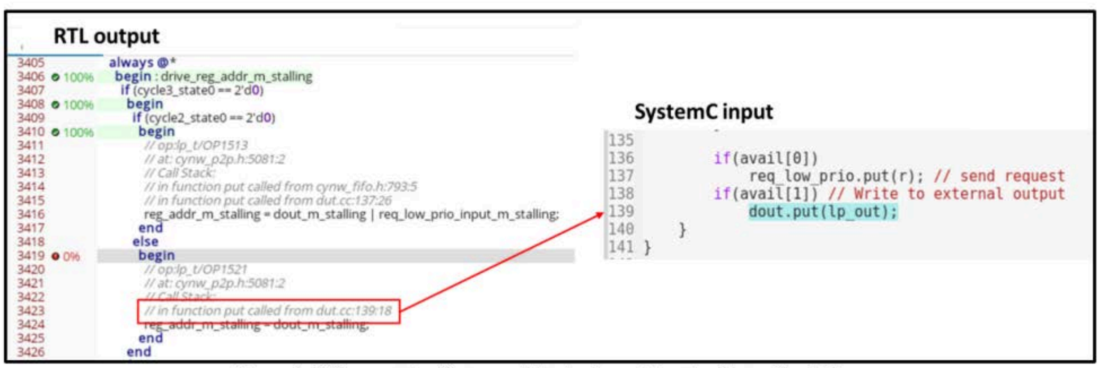


Figure 9: RTL annotation for traceability back to originating SystemC code lines

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.

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

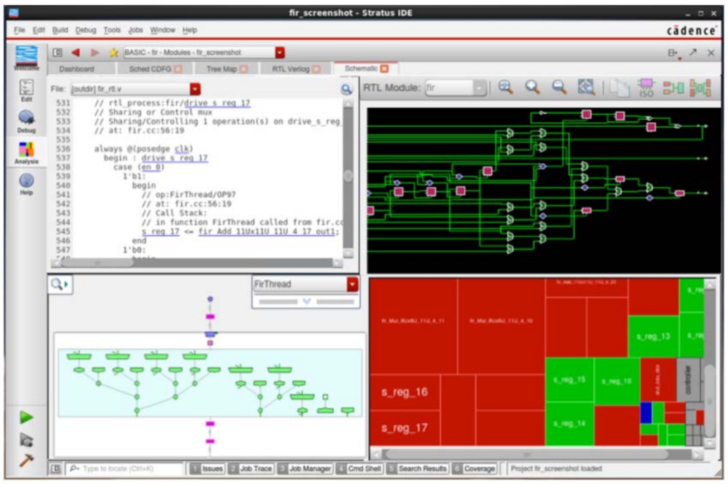
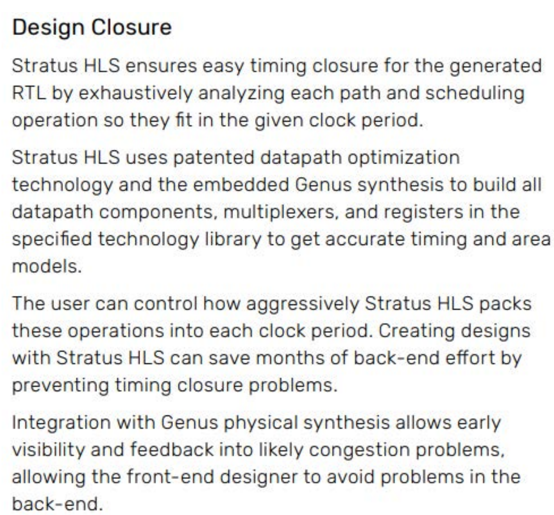


Figure 2: Complete graphical analysis with links to source code

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

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



Design Closure

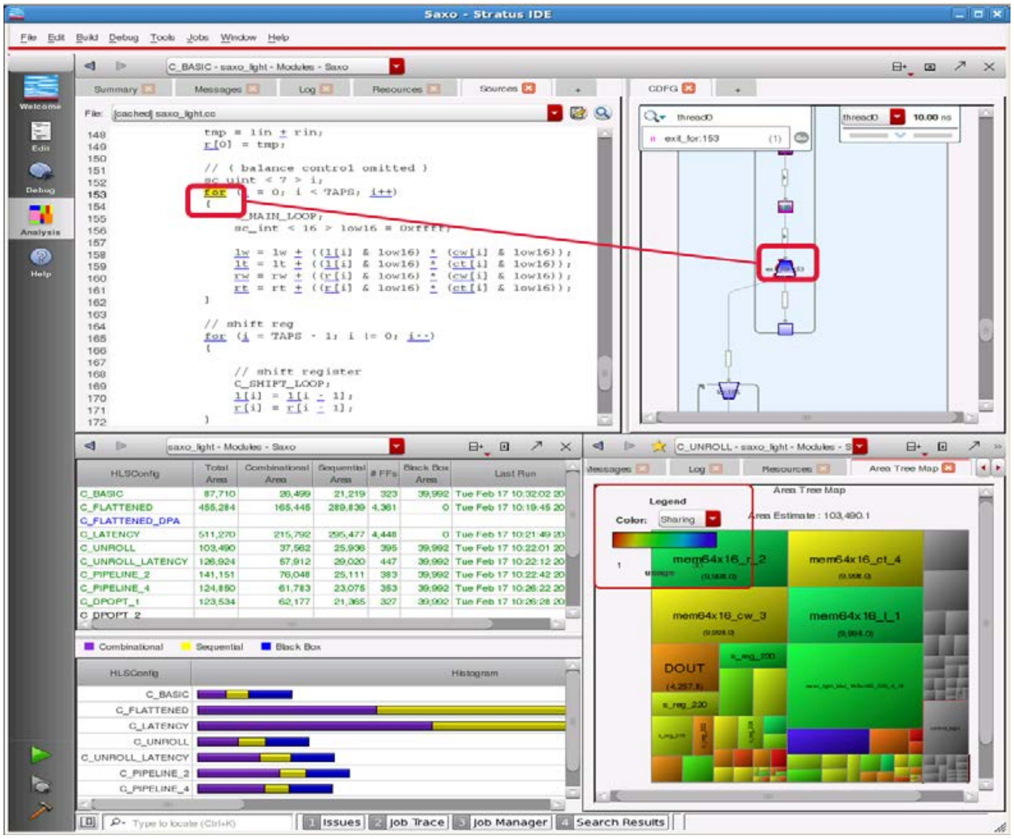
Stratus HLS ensures easy timing closure for the generated RTL by exhaustively analyzing each path and scheduling operation so they fit in the given clock period.

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



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.

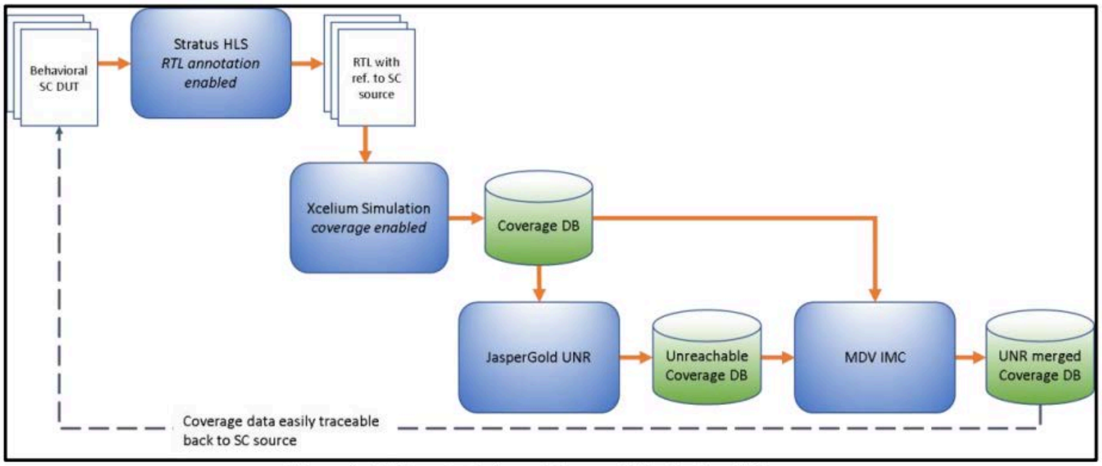


Figure 8: UNR analysis flow with traceability back to SC source

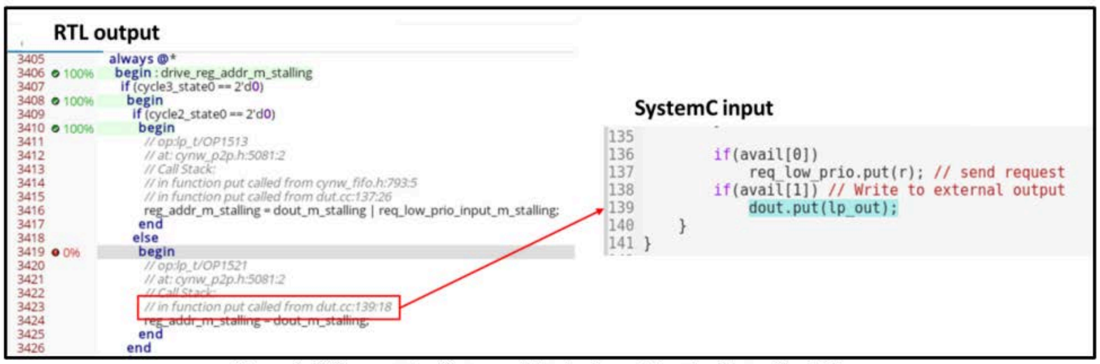


Figure 9: RTL annotation for traceability back to originating SystemC code lines

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.

93. 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 claim 1 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. Those customers include Intel, Qualcomm, Socionext, Syntiant, Himax, and Methods2Business. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html.

94. 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. Those customers include Intel, Qualcomm, Socionext, Syntiant, Himax, and Methods2Business. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html. 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.

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

96. 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: June 29, 2023

Respectfully submitted,

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

KRAMER ALBERTI LIM & TONKOVICH LLP