

1 Robert F. Kramer (SBN 181706)
 2 rkramer@krameralberti.com
 3 David Alberti (SBN 220625)
 4 dalberti@krameralberti.com
 5 Sal Lim (SBN 211836)
 6 slim@krameralberti.com
 7 Russell Tonkovich (SBN 233280)
 8 rtonkovich@krameralberti.com
 9 Robert C. Mattson (*pro hac vice*)
 10 rmattson@krameralberti.com
 11 James P. Barabas (*pro hac vice*)
 12 jbarabas@krameralberti.com
 13 Hong S. Lin (SBN 249898)
 14 hlin@krameralberti.com
 15 Robert Y. Xie (SBN 329126)
 16 rxie@krameralberti.com
 17 **KRAMER ALBERTI LIM**
 18 **& TONKOVICH LLP**
 19 577 Airport Boulevard, Suite 250
 20 Burlingame, California 94010
 21 Telephone: (650) 825-4300
 22 Facsimile: (650) 460-8443

23 *Attorneys for Plaintiff*
 24 Semiconductor Design Technologies, LLC

25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525
 526
 527
 528
 529
 530
 531
 532
 533
 534
 535
 536
 537
 538
 539
 540
 541
 542
 543
 544
 545
 546
 547
 548
 549
 550
 551
 552
 553
 554
 555
 556
 557
 558
 559
 560
 561
 562
 563
 564
 565
 566
 567
 568
 569
 570
 571
 572
 573
 574
 575
 576
 577
 578
 579
 580
 581
 582
 583
 584
 585
 586
 587
 588
 589
 590
 591
 592
 593
 594
 595
 596
 597
 598
 599
 600
 601
 602
 603
 604
 605
 606
 607
 608
 609
 610
 611
 612
 613
 614
 615
 616
 617
 618
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 660
 661
 662
 663
 664
 665
 666
 667
 668
 669
 670
 671
 672
 673
 674
 675
 676
 677
 678
 679
 680
 681
 682
 683
 684
 685
 686
 687
 688
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 699
 700
 701
 702
 703
 704
 705
 706
 707
 708
 709
 710
 711
 712
 713
 714
 715
 716
 717
 718
 719
 720
 721
 722
 723
 724
 725
 726
 727
 728
 729
 730
 731
 732
 733
 734
 735
 736
 737
 738
 739
 740
 741
 742
 743
 744
 745
 746
 747
 748
 749
 750
 751
 752
 753
 754
 755
 756
 757
 758
 759
 760
 761
 762
 763
 764
 765
 766
 767
 768
 769
 770
 771
 772
 773
 774
 775
 776
 777
 778
 779
 780
 781
 782
 783
 784
 785
 786
 787
 788
 789
 790
 791
 792
 793
 794
 795
 796
 797
 798
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809
 810
 811
 812
 813
 814
 815
 816
 817
 818
 819
 820
 821
 822
 823
 824
 825
 826
 827
 828
 829
 830
 831
 832
 833
 834
 835
 836
 837
 838
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 850
 851
 852
 853
 854
 855
 856
 857
 858
 859
 860
 861
 862
 863
 864
 865
 866
 867
 868
 869
 870
 871
 872
 873
 874
 875
 876
 877
 878
 879
 880
 881
 882
 883
 884
 885
 886
 887
 888
 889
 890
 891
 892
 893
 894
 895
 896
 897
 898
 899
 900
 901
 902
 903
 904
 905
 906
 907
 908
 909
 910
 911
 912
 913
 914
 915
 916
 917
 918
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 1000

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

1 **FIRST AMENDED COMPLAINT FOR PATENT INFRINGEMENT**

2 1. Plaintiff Semiconductor Design Technologies, LLC (“Semiconductor Design” or
3 “Plaintiff”), by its attorneys, demands a trial by jury on all issues so triable and for its First
4 Amended Complaint against Cadence Design Systems, Inc. (“Cadence” or “Defendant”) alleges
5 the following:

6 **NATURE OF THE ACTION**

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

10 **THE PARTIES**

11 3. Semiconductor Design is a corporation organized under the laws of the State of
12 Delaware with a principal place of business at 1000 N. West Street, Suite 1200, Wilmington, DE
13 19801.

14 4. Upon information and belief, Cadence is a company organized and existing under
15 the laws of the State of Delaware, with a place of business at 2655 Seely Avenue, San Jose, CA
16 95134. Cadence may be served through its registered agent, CT Corporation System, for service
17 at 330 North Brand Blvd, #700, Glendale, CA 91203.

18 5. Upon information and belief, Cadence is a global supplier of electronic system
19 design tools, including electronic design automation and analog design environment (collectively,
20 “EDA”) software tools used to design, develop, and test semiconductor chips.

21 **JURISDICTION AND VENUE**

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

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

26 8. This Court has personal jurisdiction over Cadence because Cadence has
27 purposefully availed itself of the rights and benefits of the laws of this Judicial District. Upon
28

1 information and belief, Cadence resides in the Northern District of California by maintaining a
2 regular and established place of business at 2655 Seely Avenue, San Jose, CA 95134.

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

7 10. Venue is proper in this Judicial District under 28 U.S.C. §§1391(b) and (c) and
8 §1400(b). Cadence has committed acts of infringement through, for example, installing its Stratus
9 HLS software on computers it uses to test the functionality of its Stratus HLS software, as well as
10 providing support for its customers. Moreover, Cadence has a regular and established place of
11 business in this District. Cadence’s Principal Executive Offices are physically located in San Jose
12 in the District.

13 **INTRADISTRICT ASSIGNMENTS**

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

16 **BACKGROUND**

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

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

24 14. The process of circuit design is a series of hierarchical steps that include
25 architectural design, logic design, physical design, physical verification, and sign-off. Berg Decl.
26 at ¶14. It is common in the semiconductor industry to differentiate the engineers working at the
27 different hierarchies. In the case of architectural design, the engineers are called architects or
28

1 architectural engineers. For logic/design, they are called design engineers or logic design engineers.
2 The top-level circuit specification and behavioral design is specified by the architects, whereas the
3 register-transfer level (RTL) and RTL validation are done by the design engineers. Berg Decl. at
4 ¶20.

5 **THE ASSERTED PATENTS & PATENTED TECHNOLOGY**

6 15. Semiconductor Design is the lawful owner of all rights, title, and interests in the
7 '636 patent titled "Assertion Generating System, Program Thereof, Circuit Verifying System, and
8 Assertion Generating Method," including the right to sue and recover for infringement thereof. The
9 '636 patent was duly and legally issued on October 13, 2009, naming Mr. Takamitsu Yamada as
10 the inventor. A true and correct copy of the '636 patent is attached as Exhibit A.

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

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

25 18. The use of assertions is beneficial because it reduces verification time, allows for
26 errors to be caught earlier, focuses the design effort, and pinpoints sources of error. Berg Decl. at
27 ¶¶43-47. At the time of the '636 and '167 patents, assertion practices varied widely among
28

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

4 19. Property Specification Language (“PSL”) is a popular assertion language
5 promulgated by the Institute of Electrical and Electronics Engineers (“IEEE”). ’636 patent, col.
6 2:24-30. Manual input of assertions is disadvantageous because it creates the “possibility that a
7 mistake may be made in the assertion itself.” ’636 patent, col. 3:43-47; *see also* Berg Decl. at ¶¶49-
8 50. As a result, the designer could be provided with an incorrect assertion violation, or a mistake
9 in the design may be allowed to slip through. ’636 patent, col. 3:47-50. Manual verification also
10 negatively affects verification efficiency because the assertion description language will require
11 error checking and debugging. ’636 patent, col. 3:50-53.

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

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

23 22. The JP ’918 application describes an approach to “generate[] an assertion for
24 verifying a circuit by extracting a data transfer structure from design data of a RTL and expanding
25 the data transfer structure into a graphic structure.” ’636 patent, col. 4:12-16. But this approach is
26 also disadvantageous because the RTL is the thing to be verified, and an assertion extracted from
27 the RTL may not match the expected behavior from the circuit specification. ’636 patent, col. 4:16-
28 19; *see also* Berg Decl. at ¶52.

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

5 24. The '636 patent relates to generating and verifying design data of a semiconductor
6 integrated circuit. For example, the '636 patent discloses graphically editing a specification of an
7 integrated circuit and generating a property that verifies the specification of the semiconductor
8 integrated circuit based on design data. The property is in turn converted into an assertion
9 description language if the property is to be verified during asset verification. The '636 patent also
10 describes how to automatically generate properties from design data of a semiconductor integrated
11 circuit, using a syntax analyzer and a property extractor. *E.g.*, '636 patent, Fig. 2, col. 8:43-47,
12 9:53-10:19.

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

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

1 27. The invention recited in claim 8 is important because the use of the specification
2 as an input uses the specification generated by the architecture engineers, whereas manual input of
3 assertions is performed by the design engineers by interpreting the specification. Berg Decl. at ¶60.
4 As a result, the actual specification and the interpreted specifications may not be identical. For a
5 small chip (e.g. ~1000 transistors), the possibility of interpretation error is small, but for a large
6 chip (e.g. ~500 million transistors), the possibility of error is nearly certain. Berg Decl. at ¶60. The
7 '636 invention forces the specification and the assertions to be consistent with one another by using
8 the specification as an input into the property generation, i.e., assertion generation. Berg Decl. at
9 ¶60. This method eliminates errors in communication between the architects and the design
10 engineers. Berg Decl. at ¶60. Using the claimed framework, chip architects can create a high-level
11 behavioral circuit and later design engineers can rely on assertion verification to ensure that the
12 synthesized RTL accurately captures the desired specification of the chip design. Berg Decl. at
13 ¶60.

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

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

1 must assume the design intention of the semiconductor integrated circuit. '636 patent, col. 2:31-33.
2 As a result, hand-written assertions cannot be relied upon, either for accuracy or complete
3 functional coverage, to determine when a system no longer has to be revised. '636 patent, col. 3:43-
4 50. Further, hand-written assertions must themselves be verified and debugged. '636 patent, col.
5 3:50-53. Simply looking at a specification to make assumptions about the design intent is not the
6 same as using an automated tool to verify the specification of an integrated circuit. A human simply
7 cannot achieve complete or accurate functional coverage of an integrated circuit, and thus cannot
8 truly verify the circuit's specification from assumptions about design intent.

9 30. The solution proposed by the '636 patent also provides advantages over other
10 automated techniques that generate assertions from the RTL description of a design because the
11 RTL description itself may deviate from the design intent. Thus, if the assertions are generated
12 from RTL, the assertions are not guaranteed to match the design intent. '636 patent, col. 4:4-19,
13 4:29-31; *see also* Berg Decl. at ¶62.

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

20 32. Thus, the claims of the '636 patent do not preempt all ways of generating
21 assertions, but are rather directed to specific approaches of generating assertions based on a
22 property read from design data generated from a specification of a semiconductor integrated circuit.
23 These approaches, which are described and claimed in the '636 patent, are fundamentally different
24 from and superior to both hand-written assertions, which require one to assume which assertions
25 might cover the design intent, and other automated assertion-generating systems, which generate
26 assertions from the RTL description and not from the design data of the specification for the
27 semiconductor integrated circuit. Despite those differences, the generation of assertions from RTL
28 descriptions, for example, is still performed by some commercial software packages today and is

1 not preempted by the claims of the '636 patent.

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

4 34. Semiconductor Design is the lawful owner of all rights, title, and interests in the
5 '167 patent titled "Semiconductor Design Support Device, Semiconductor Design Support Method,
6 and Manufacturing Method for Semiconductor Integrated Circuit," including the right to sue and
7 recover for infringement thereof. The '167 patent was duly and legally issued on June 28, 2011,
8 naming Yasutaka Tsukamoto as the inventor. A true and correct copy of the '167 patent is attached
9 as Exhibit B.

10 35. The '167 patent explains that "an LSI (large scale integration) circuit has come to
11 have greater size and complexity." '167 patent, 1:27-28. The '167 patent explains that "a life cycle
12 of an electronic device including the LSI is becoming shorter." *Id.*, 1:28-30. Accordingly, the '167
13 patent explains that "a circuit design is requested to be completed in a shorter time period." Col.
14 1:30-31. However, "known design methods may not design the LSI having the required greater size
15 and complexity effectively." '167 patent, col. 1:31-33. Accordingly, a technological need existed:
16 "various EDA (electronic design automation) tools for describing a design at an increased abstract
17 level are proposed." '167 patent, col. 1:33-35. Rather than design circuits at the transistor level,
18 digital circuit designers use computer tools to design circuits at higher levels of abstraction, *e.g.*, a
19 behavioral level. Berg Decl. at ¶71. The circuit designer defines a behavioral model (or
20 description) that describes how the circuit is to function. A behavioral synthesis tool can convert
21 the behavioral description to a register transfer level (RTL) description. '167 patent, col. 1:36-39.
22 The behavioral description is not aware of timing or clock cycles. '167 patent, col. 1:40-43. The
23 RTL description is an intermediate level abstraction that includes registers and combinational logic.
24 The RTL description operates according to a clock, and as such is aware of timing. '167 patent,
25 col. 1:43-45. The RTL description can be simulated in software to determine the overall latency of
26 the RTL circuit. '167 patent, col. 1:48-55.

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

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

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

22 38. The claims of the '167 patent do not merely recite a preexisting method of
23 performance, but rather are directed to specific technological improvements to semiconductor
24 design, analysis, and simulation technology. Preexisting methods do not efficiently check the
25 latency of the blocks of the behavioral description. For example, the '167 patent describes and
26 claims the use of a correspondence table that maps the states in the RTL description to the blocks
27 in the behavioral description and the use of a latency analyzer that calculates the total latency in
28 each block based on the relationships between the blocks and the states in the RTL description in

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

5 39. The claims of the '167 patent do not preempt all ways of generating an RTL
6 description or performing a logic simulation on the RTL description, but are rather directed to
7 specific approaches of mapping states in the RTL description to blocks in the behavioral description
8 to determine the latency of each block from the RTL simulation. The '167 patent does not preempt
9 behavioral synthesis, or analysis of a resulting RTL description. The claims require much more
10 than that (i.e., latency analysis on a block-by-block basis corresponding to the behavioral
11 description), and the use of a correspondence table generator to determine the correspondence
12 between of different portions (e.g., blocks and/or states) of the behavioral and RTL descriptions.
13 As a result, the ordered combination of elements provides an improved semiconductor design
14 support tool that enables a designer to determine the latencies associated with different blocks of a
15 behavioral description. Providing an improvement to an existing device and/or process would not
16 preempt the existing device and/or process. Accordingly, each claim of the '167 patent thus recites
17 a combination of elements sufficient to ensure that the claim amounts to significantly more than a
18 patent on an ineligible concept.

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

28 41. Semiconductor Design is the owner of all right, title, and interest in and to each of

1 the Asserted Patents with full and exclusive right to bring suit to enforce the Asserted Patents,
2 including the right to recover for past damages and/or royalties prior to the expiration of the
3 Asserted Patents.

4 42. The Asserted Patents are valid and enforceable.

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

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

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

14 45. Upon information and belief, Cadence has directly infringed and continues to
15 directly infringe at least, for example, claim 8 of the ’636 patent by making, using, selling, and/or
16 offering for sale its 636 Accused Products, which are stored on computer-readable media encoded
17 with a program for a computer in an assertion generating system. Cadence’s infringing use of the
18 636 Accused Products includes its internal use and testing of those products, its demonstration of
19 the 636 Accused Products to third parties, its storage of 636 Accused Products on servers for
20 transmitting the 636 Accused Products to customers or for hosting those products, and its
21 distribution of copies of the 636 Accused Products to customers.

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

25 47. Upon information and belief, Cadence had full knowledge of the ’636 patent, for
26 example, based upon Cadence’s receipt of various letters sent by Ricoh Company, Ltd., the former
27 owner of the ’636 patent, to Cadence senior personnel including letters sent to Cadence on
28

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

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

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

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

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

24 51. Claim 1 of the Cadence '060 patent recites:

25 1. A computer-implemented method for handling assertion libraries in verification
26 of a design, the method comprising:

27 structuring and implementing at least one verification component for the verification
28 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

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

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

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

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

12 1. A computer implemented method for assertion synthesis of circuitry comprising:
13 using at least one computer system which comprises at least one processor and is
14 programmed for performing:

15 converting an assertion formula to a sequence implication form using one or more
16 semantic preserving rewrite rules, wherein

17 the sequence implication form converted by the one or more semantic preserving
18 rewrite rules is semantically equivalent to the assertion formula;

19 optimizing a resulting formula, which is converted from the assertion formula, to
20 reduce a number of states in a state machine that represents the circuitry;

21 synthesizing the resulting formula to the state machine using context aware
22 sequence synthesis, wherein

23 the context aware sequence synthesis synthesizes the resulting formula based at least
24 in part upon context of one or more sequences in the resulting formula; and
25 verifying the circuitry using the state machine.

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

28 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

1 circuit layout that is independent from the portions of the electronic circuit layout in
2 the other partitions of the plurality of partitions;

3 generating, by the processor, an empty delta view for each of the plurality of
4 partitions, wherein the empty delta view is a cell view corresponding to a record in
5 a cell database, wherein the empty delta view is configured to be checked in and
6 checked out locally;

7 concurrently rendering, by the processor, one or more design layout interfaces each
8 showing at least one partition of the plurality of partitions, wherein the at least one
9 partition is configured to be edited using the respective design layout interface;

10 while receiving one or more concurrent modification instructions to edit the
11 respective partitions configured to be edited from the one or more design layout
12 interfaces:

13 modifying, by the processor, the respective empty delta view associated with the
14 respective partition configured to be edited, according to the respective modification
15 instructions received from the respective design layout interface to generate a
16 respective modified delta view from the respective empty delta view, wherein the
17 modified delta view is configured to checked in and checked out locally; and

18 updating, by the processor, each respective partition according to the respective
19 modified delta view, thereby allowing parallel editing of the electronic circuit layout
20 by a plurality of electronic circuit designers.

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

23 Embodiments described herein provide a new layout editor tool allowing designers
24 to concurrently edit various aspects of an electronic circuit layout, even at disparate
25 hierarchical levels of the design. The new layout editor tool enables multiple
26 electronic circuit designers to concurrently edit a layout a different hierarchical
27 levels, by logically establishing editable child sub cell-level partitions within a
28 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:

1 modifying, by the processor, the second database record corresponding to the top-
2 level delta view, thereby resulting in a modified second database record containing
a modified top-level delta view;

3 in response to receiving, from the client computer, a command to modify a subcell
4 view from a top-level partition:

5 generating, by the processor, a plurality of sub-level partitions based upon the
6 predefined attribute, each respective sub-level partition comprising the predefined
7 attribute inherited from the top-level partition that corresponds to the portion of the
8 electronic circuit layout at a sub level stored as a third database record including the
9 predefined attribute;

10 generating, by the processor, a plurality of sub-level delta views corresponding
11 respectively to each sub-level partition, wherein each respective sub-level delta view
12 is a subcell view configured to store edits to the respective sub-level partition as a
13 fourth database record; and

14 modifying, by the processor, the fourth database record corresponding to the sub-
15 level delta view according to the command, thereby resulting in a modified fourth
16 database record containing a modified sub-level delta view; and

17 displaying, by the processor, a representation of the modified sub-level delta view
18 to the client computer having the access rights,

19 wherein the first database record and the third database record remain unmodified
20 by the commands.

21 58. These Cadence patents are directed to analogous subject matter as Plaintiff's '636
22 and '167 patents, and together they depict the landscape for patent-eligible improvements to EDA
23 technology. In the course of applying for its EDA patents, during prosecution of the '183 patent,
24 Cadence provided a lengthy explanation to the Patent Office under oath as to why EDA inventions
25 are eligible for patenting under Section 101. In fact, to persuade the Patent Office to issue patent
26 claims for Cadence's own EDA inventions analogous to Plaintiff's '636 and '167 patents, Cadence
27 represented to the Patent Office under oath that its EDA inventions are not simply directed to an
28 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

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

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

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

- 13
- 14 ▶ Synthesis of SystemC assertions and C++ asserts to SystemVerilog assertions (SVAs)

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

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

21 (annotated).

22 61. The Stratus HLS software is used for assertion verification of a semiconductor
23 integrated circuit.

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

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

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

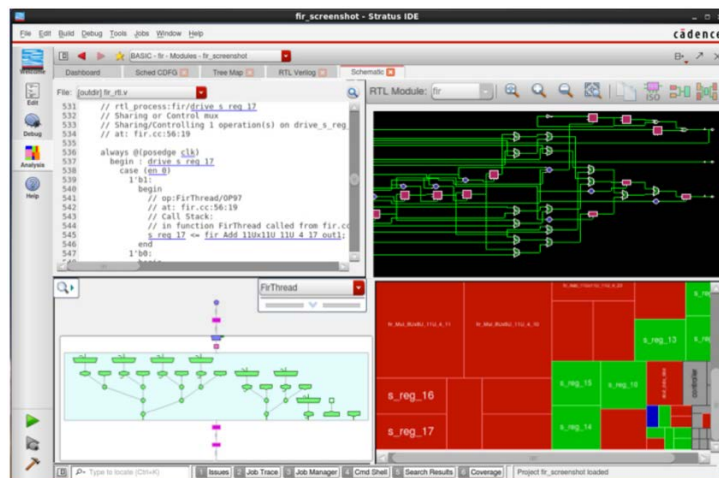
4 62. The computer upon which the Stratus HLS software is installed executes a
 5 specification inputting step that generates design data of the semiconductor integrated circuit by
 6 graphically editing a specification of the semiconductor integrated circuit based on user operations.

7 GUI

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

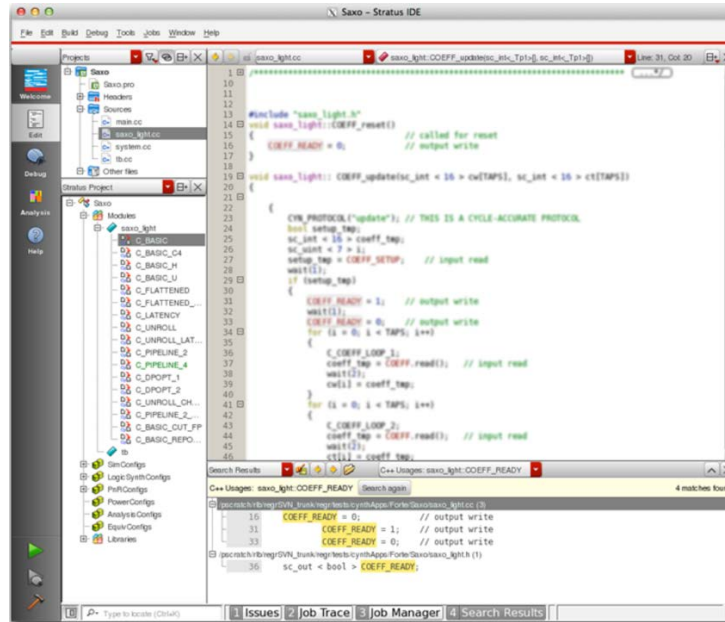
13 The Stratus analysis environment includes SystemC and
 14 RTL source linking, control and dataflow graphs, schematic
 15 viewer, and pipeline analysis, as well as QoR reporting and
 16 visualization to judge the impact of architectural optimiza-
 17 tions. Although most commonly used via the GUI,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

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

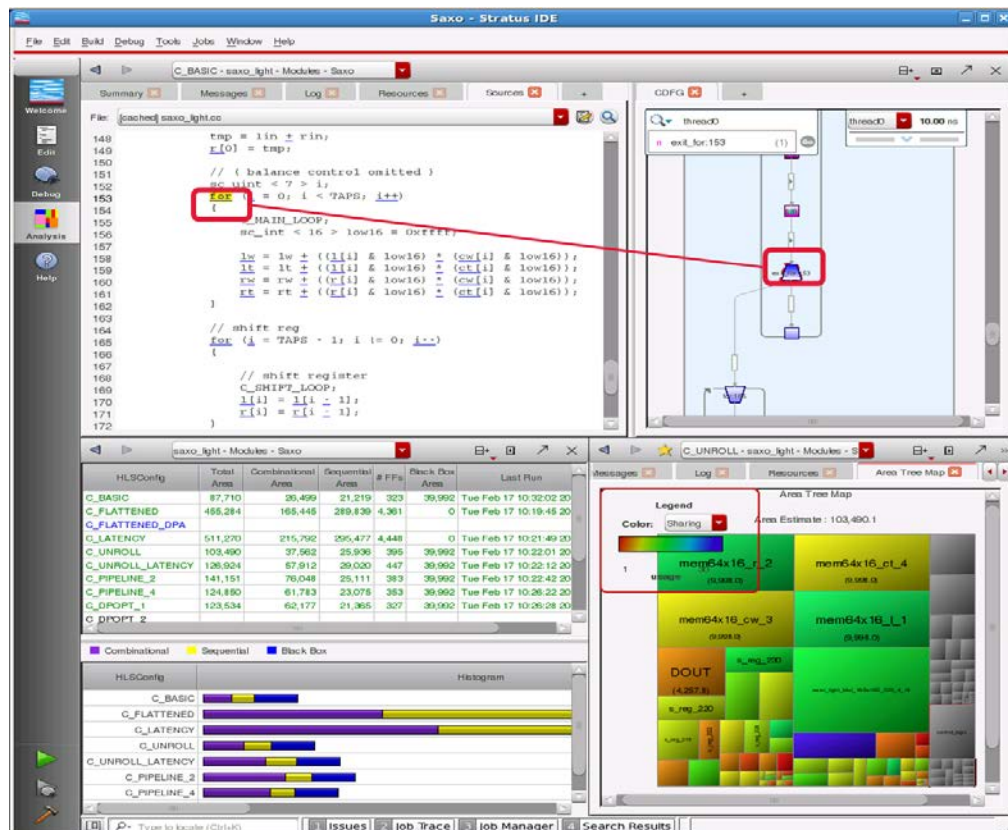


22 *Figure 2: Complete graphical analysis with links to source code*

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

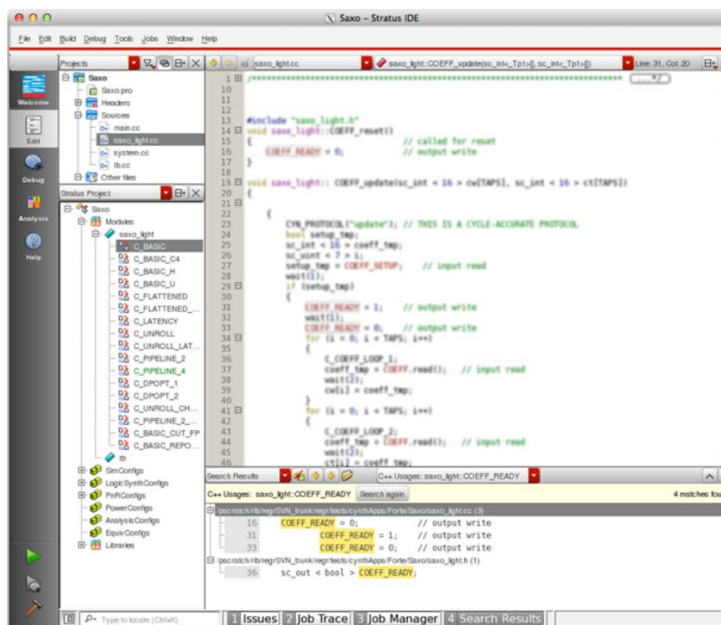


1 Source: <https://www.linkedin.com/pulse/how-dramatically-reduce-time-from-architecture-spec-tapeout-laviv/>

2
3 Integrated with the Cadence verification suite, Stratus HLS
4 supports automated mixed-language (SystemC and RTL)
5 verification and debug including assertions, debugging,
6 waveforms, and linkage back to the original SystemC design.

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

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



The screenshot shows the Saxo - Stratus IDE interface. The main window displays a C++ code file named saxo_light.cc. The code includes a header file, defines a setup function, and contains a main function with a loop that reads and writes data. The search results window at the bottom shows 4 matches for the search term 'saxo_light::COEFF_READY'.

```

1  #include "saxo_light.h"
2  #include "saxo_light.h"
3  #include "saxo_light.h"
4  #include "saxo_light.h"
5  #include "saxo_light.h"
6  #include "saxo_light.h"
7  #include "saxo_light.h"
8  #include "saxo_light.h"
9  #include "saxo_light.h"
10 #include "saxo_light.h"
11 #include "saxo_light.h"
12 #include "saxo_light.h"
13 #include "saxo_light.h"
14 #include "saxo_light.h"
15 #include "saxo_light.h"
16 #include "saxo_light.h"
17 #include "saxo_light.h"
18 #include "saxo_light.h"
19 #include "saxo_light.h"
20 #include "saxo_light.h"
21 #include "saxo_light.h"
22 #include "saxo_light.h"
23 #include "saxo_light.h"
24 #include "saxo_light.h"
25 #include "saxo_light.h"
26 #include "saxo_light.h"
27 #include "saxo_light.h"
28 #include "saxo_light.h"
29 #include "saxo_light.h"
30 #include "saxo_light.h"
31 #include "saxo_light.h"
32 #include "saxo_light.h"
33 #include "saxo_light.h"
34 #include "saxo_light.h"
35 #include "saxo_light.h"
36 #include "saxo_light.h"
37 #include "saxo_light.h"
38 #include "saxo_light.h"
39 #include "saxo_light.h"
40 #include "saxo_light.h"
41 #include "saxo_light.h"
42 #include "saxo_light.h"
43 #include "saxo_light.h"
44 #include "saxo_light.h"
45 #include "saxo_light.h"
46 #include "saxo_light.h"
47 #include "saxo_light.h"
48 #include "saxo_light.h"
49 #include "saxo_light.h"
50 #include "saxo_light.h"
51 #include "saxo_light.h"
52 #include "saxo_light.h"
53 #include "saxo_light.h"
54 #include "saxo_light.h"
55 #include "saxo_light.h"
56 #include "saxo_light.h"
57 #include "saxo_light.h"
58 #include "saxo_light.h"
59 #include "saxo_light.h"
60 #include "saxo_light.h"
61 #include "saxo_light.h"
62 #include "saxo_light.h"
63 #include "saxo_light.h"
64 #include "saxo_light.h"
65 #include "saxo_light.h"
66 #include "saxo_light.h"
67 #include "saxo_light.h"
68 #include "saxo_light.h"
69 #include "saxo_light.h"
70 #include "saxo_light.h"
71 #include "saxo_light.h"
72 #include "saxo_light.h"
73 #include "saxo_light.h"
74 #include "saxo_light.h"
75 #include "saxo_light.h"
76 #include "saxo_light.h"
77 #include "saxo_light.h"
78 #include "saxo_light.h"
79 #include "saxo_light.h"
80 #include "saxo_light.h"
81 #include "saxo_light.h"
82 #include "saxo_light.h"
83 #include "saxo_light.h"
84 #include "saxo_light.h"
85 #include "saxo_light.h"
86 #include "saxo_light.h"
87 #include "saxo_light.h"
88 #include "saxo_light.h"
89 #include "saxo_light.h"
90 #include "saxo_light.h"
91 #include "saxo_light.h"
92 #include "saxo_light.h"
93 #include "saxo_light.h"
94 #include "saxo_light.h"
95 #include "saxo_light.h"
96 #include "saxo_light.h"
97 #include "saxo_light.h"
98 #include "saxo_light.h"
99 #include "saxo_light.h"
100 #include "saxo_light.h"

```

11
12
13
14
15
16
17
18
19
20
21
22 Source: [https://login.cadence.com/content/dam/cadence-
23 www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf](https://login.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf)

24 64. The computer upon which the Stratus HLS software is installed executes a property
25 generating step that reads the design data generated at the specification inputting step from the
26 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 intent 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.

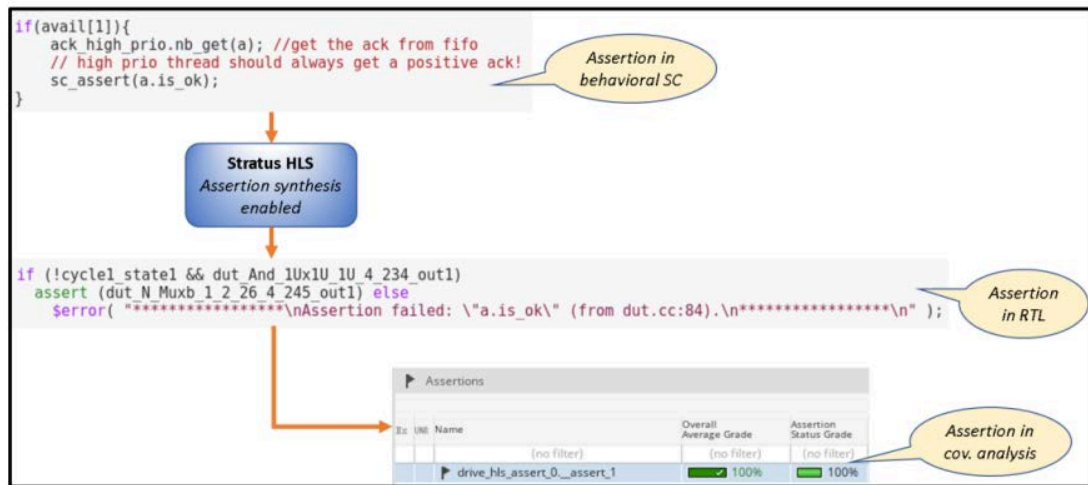


Figure 5: Assertion synthesis flow

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>.

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

4
5 **3. Exploration:** Once configured, Stratus HLS explores the solution space. Exploration is governed by a Stratus HLS Tcl control
6 file that defines how **constraints** are changed and imposed on the candidate micro-architecture designs.

7 Source: <https://www.cadence.com/content/dam/cadence->
8 [www/global/en_US/documents/tools/digital-design-signoff/cadence-stratus-hls-algorithm-wp.pdf](https://www.global/en_US/documents/tools/digital-design-signoff/cadence-stratus-hls-algorithm-wp.pdf)
9 (annotated).

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

12 Source: <https://login.cadence.com/content/dam/cadence->
13 [www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf](https://www.global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf)

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

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

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

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28



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>.

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

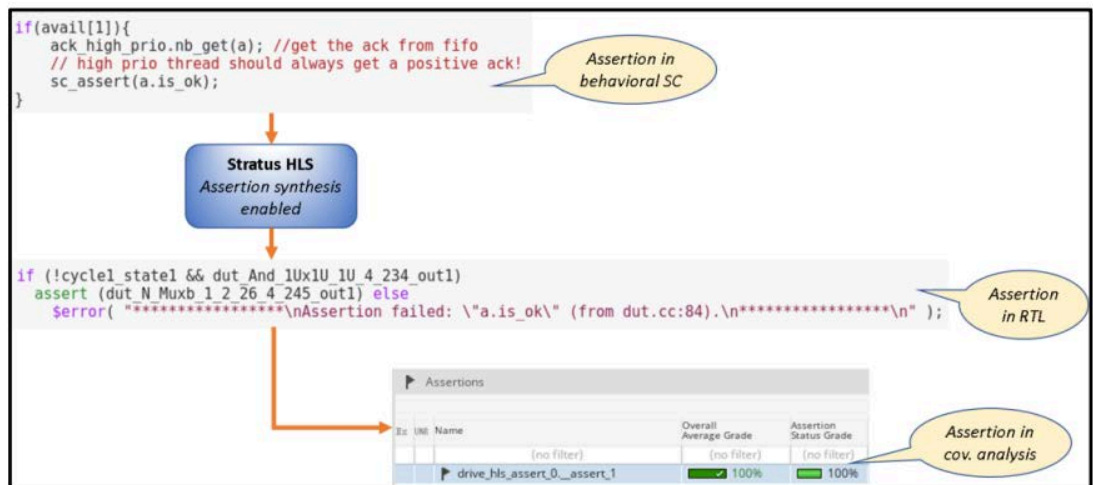
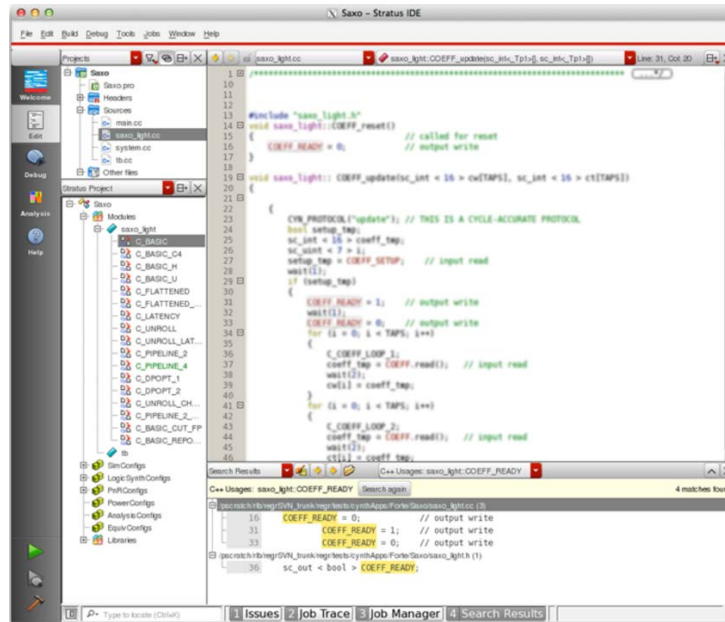


Figure 5: Assertion synthesis flow

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>.

[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.

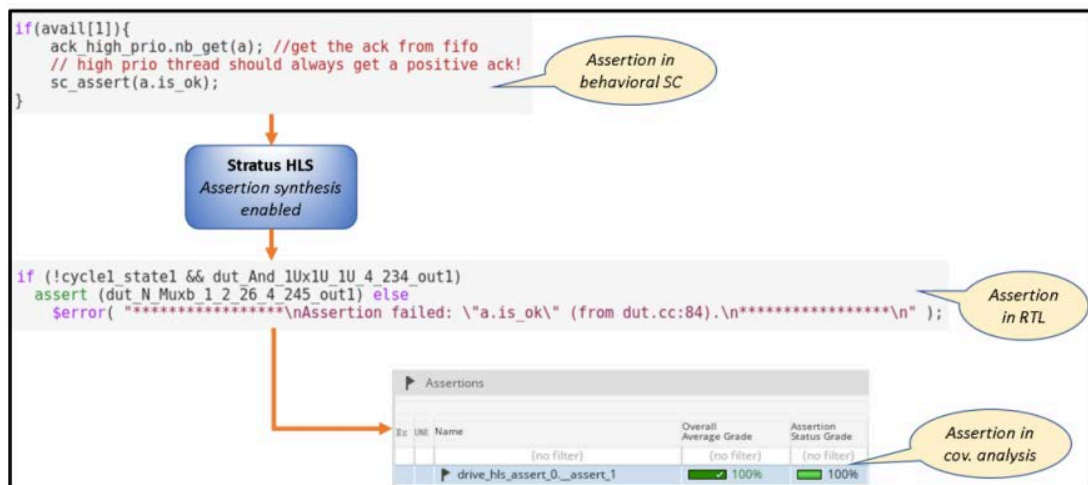


Figure 5: Assertion synthesis flow

1 Source: S. Dahir, “Using HLS to improve Design-for-Verification of multi-pipeline designs with
2 resource sharing,” DVCON 2021 at 5-6, available at [https://dvcon-proceedings.org/wp-
3 content/uploads/using-hls-to-improve-design-for-verification-of-multi-pipeline-designs-with-
4 resource-sharing.pdf.pdf](https://dvcon-proceedings.org/wp-content/uploads/using-hls-to-improve-design-for-verification-of-multi-pipeline-designs-with-resource-sharing.pdf.pdf)

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

- 8
- 9
- 10
 - ▶ Synthesis of SystemC assertions and C++ asserts to SystemVerilog assertions (SVAs)

11 Source: [https://login.cadence.com/content/dam/cadence-
13 www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf](https://login.cadence.com/content/dam/cadence-12 www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf)

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

18 [https://login.cadence.com/content/dam/cadence-
21 www/global/en_US/documents/tools/digital-
22 design-signoff/stratus-high-level-synthesis-ds.pdf](https://login.cadence.com/content/dam/cadence-19 www/global/en_US/documents/tools/digital-
20 design-signoff/stratus-high-level-synthesis-ds.pdf) (annotated).

23 Another way to improve DFV is for designers to add assertions in their code to identify illegal conditions. For
24 example, the high-priority thread should never get a NACK response on a memory access request. Using Stratus™
25 HLS, SystemC/C++ assertions can be synthesized into the generated RTL implementation by turning on the
26 *synthesize_asserts* feature. This allows the designer to communicate effectively and clearly the design intend and
27 assumptions to the verifiers. The verifiers would use these assertions as targets to be verified and covered in their
28 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.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

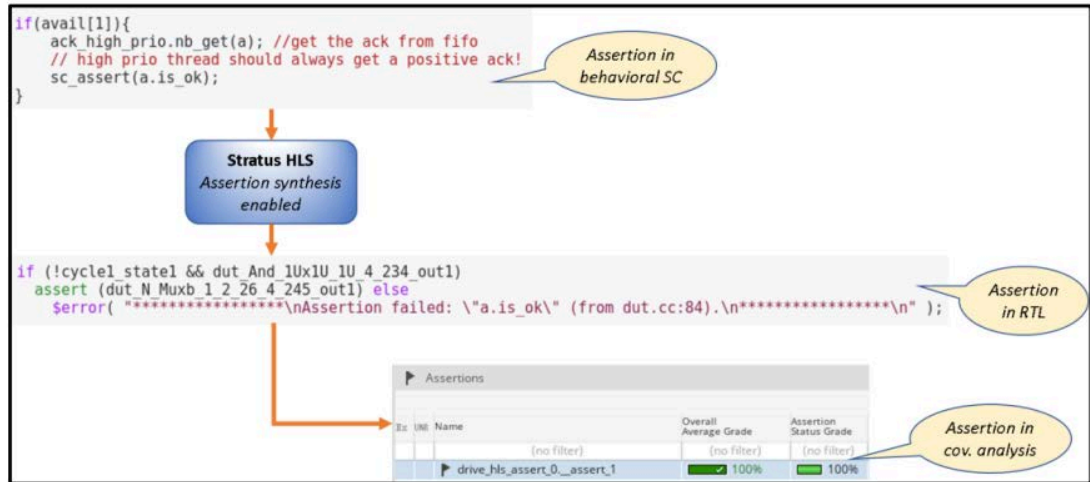


Figure 5: Assertion synthesis flow

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.

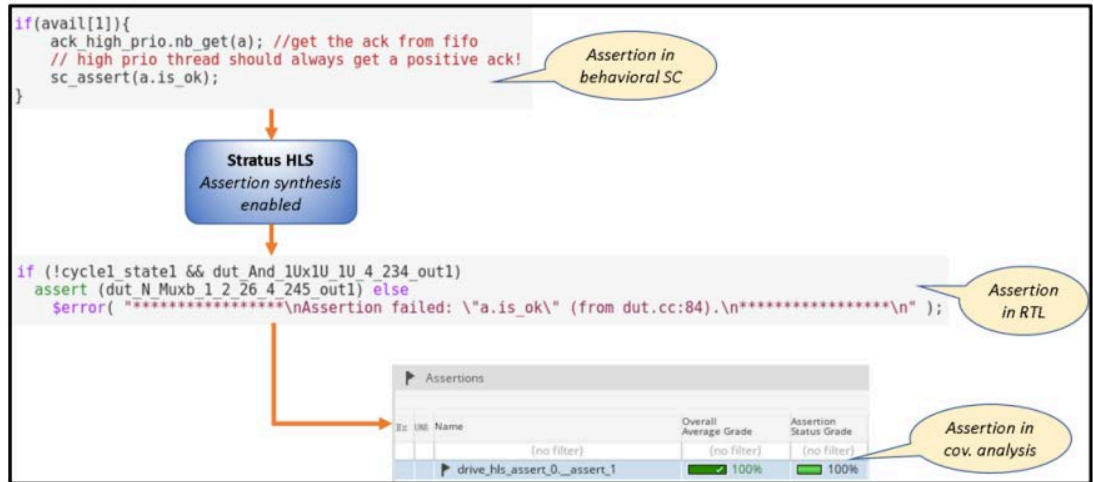


Figure 5: Assertion synthesis flow

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-

1 [level-synthesis.html](#); https://www.cadence.com/en_US/home/support.html, including all related
2 domains and subdomains. Cadence does so knowing that its customers will commit these
3 infringing acts. Despite its knowledge of the '636 patent, Cadence continues to make, use, sell,
4 and/or offer for sale the 636 Accused Products thereby specifically intending for and inducing its
5 customers to infringe the '636 patent. Those customers include Intel, Qualcomm, Socionext,
6 Syntiant, Himax, and Methods2Business. [https://www.cadence.com/en_US/home/tools/digital-
7 design-and-signoff/synthesis/stratus-high-level-synthesis.html](https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html).

8 71. Upon information and belief, Cadence has indirectly infringed and continues to
9 indirectly infringe at least claim 8 of the '636 patent in violation of 35 U.S.C. § 271(c) by
10 contributing to the infringement by its customers. Those customers include Intel, Qualcomm,
11 Socionext, Syntiant, Himax, and Methods2Business.

12 [https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-
14 level-synthesis.html](https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-
13 level-synthesis.html). Cadence sells or offers for sale in the United States the 636 Accused Products,
15 including for example the Stratus HLS software, with knowledge that they are especially designed
16 or adapted to operate in a manner that infringes that patent and despite the fact that the infringing
17 technology or aspects of the 636 Accused Products are not a staple of commerce suitable for
18 substantial non-infringing use. For example, Cadence knows that the Stratus HLS software
19 infringes when stored on a computer readable media because it enables a computer to provide an
20 assertion-generation system that generates an assertion description for assertion verification of a
21 semiconductor integrated circuit and to execute the steps recited in claim 8. Cadence is aware that
22 the Stratus HLS software operates as described above, that such functionality infringes the '636
23 patent, including claim 8, and that the Accused Products have no substantial non-infringing use.
24 Cadence continues to sell and offer for sale in the United States its infringing products after
25 receiving notice of the '636 patent and knew how it is infringed by Cadence's products. The portion
26 of the Stratus HLS software that maps to claim 8 (i.e., the infringing aspect) has no substantial non-
27 infringing uses.

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

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

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

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

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

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

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

25 78. Upon information and belief, Cadence had full knowledge of the '167 patent, for
26 example, based upon Cadence's receipt of various letters sent by Ricoh Company, Ltd., the former
27 owner of the '167 patent, to Cadence senior personnel including letters sent to Cadence on
28

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

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

6 80. A computer installed with the Stratus HLS software is a semiconductor design
7 support device for designing a semiconductor integrated device. Stratus HLS software starts with
8 a behavioral description, which is then converted into HDL.

- 9
- 10
 - ▶ Synthesis of SystemC assertions and C++ asserts to SystemVerilog assertions (SVAs)

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

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

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

21 (annotated).

22 Cadence® Stratus™ High-Level Synthesis (HLS) automatically creates high-quality register
23 transfer level (RTL) design implementations for ASIC, system-on-chip (SoC), and FPGA targets
24 from high-level IEEE 1666 SystemC™, C++, and MATLAB® descriptions. The proven successes
25 of Stratus HLS in production designs around the world are testament to its consistently high-
26 quality results, mature feature set, and complete design coverage. While most widely used for
27 image processing, wireless, and machine learning (ML) applications, products built with
28 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](https://login.cadence.com/content/dam/cadence-
www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf)

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

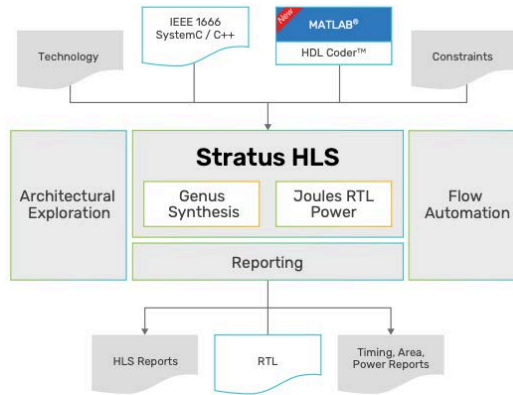


Figure 1: Stratus HLS uses the Genus synthesis and Joules power engines to create high-quality RTL targeted to your technology and design constraints

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 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

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.

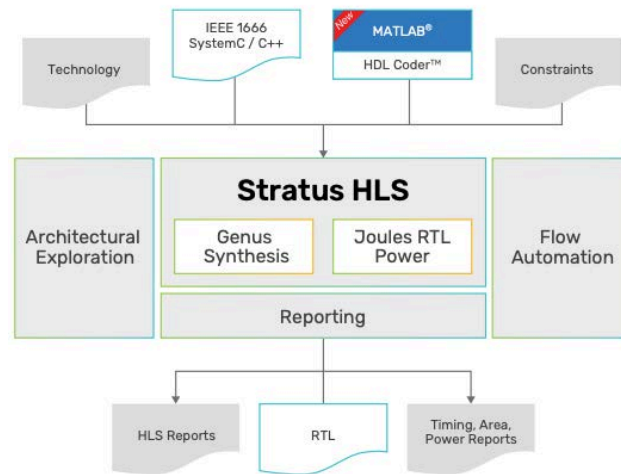


Figure 1: Stratus HLS uses the Genus synthesis and Joules power engines to create high-quality RTL targeted to your technology and design constraints

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.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

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

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

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

8 Design Closure

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

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

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

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

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

28 85. The Stratus HLS software includes the Genus Synthesis Solution engine.

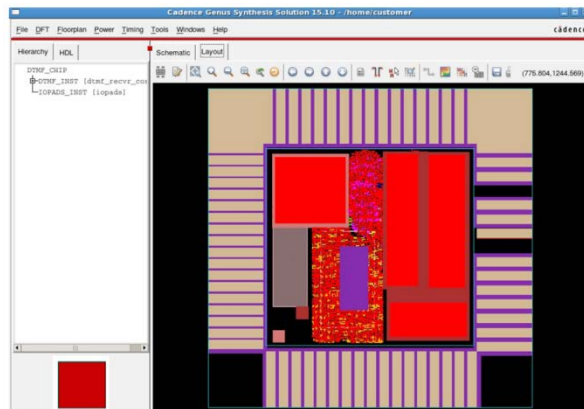
- 29 ▶ Genus logic synthesis and Joules power engines inside of
30 Stratus HLS provide accurate timing, area, and power
31 estimates

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

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

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

1 Source: <https://www.cadence.com/content/dam/cadence->
2 [www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf](https://www.cadence.com/content/dam/cadence-)



3
4
5
6
7
8
9
10 *Figure 1: The Genus Synthesis Solution enables timing debug with*
11 *physical interconnect knowledge built-in. Cross-probe to the*
12 *physical viewer to see associated wirelengths, floorplan*
13 *blockages, and estimated routing, and extract the chip-/block-*
14 *level physical context for use in unit-level RTL design.*

15 Source: <https://www.cadence.com/content/dam/cadence->
16 [www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf](https://www.cadence.com/content/dam/cadence-)

- 17 ▶ 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%

18 Source: <https://www.cadence.com/content/dam/cadence->
19 [www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf](https://www.cadence.com/content/dam/cadence-)

20 Tight Correlation to Place and Route

21 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

22
23
24 Source: <https://www.cadence.com/content/dam/cadence->

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

27 [www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf](https://www.cadence.com/content/dam/cadence-)

1 Source: <https://www.cadence.com/content/dam/cadence->
 2 [www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf](https://www.cadence.com/content/dam/cadence-)

3
 4 **Register Retiming**

5 The Genus Synthesis Solution can retime registers along
 6 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.

7 Source: <https://www.cadence.com/content/dam/cadence->
 8 [www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf](https://www.cadence.com/content/dam/cadence-)

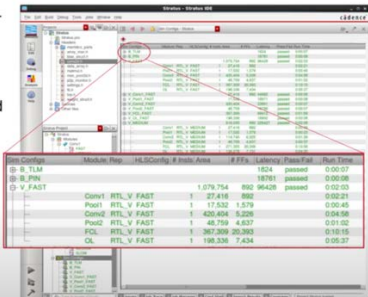
9
 10 **Parasitic extraction and delay calculation.** The Genus
 and Innovus solutions leverage unified parasitic extraction
 and delay calculation with full support for advanced-node
 waveform modeling.

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

14 Source: <https://www.cadence.com/content/dam/cadence->
 15 [www/global/en_US/documents/tools/digital-design-signoff/genus-product-brief-rebrand-v1.pdf](https://www.cadence.com/content/dam/cadence-)

18 **Synthesis and RTL Simulation**

- 19 • Stratus™ HLS synthesizes each module to RTL
 - Targets specified technology and clock period
 - Each module synthesized according to given settings and configurations
- 20 • RTL is simulated with the same testbench used for the behavioral SystemC® model
- 21 • RTL simulation...
 - Verifies network performance (latency, throughput)
 - Verifies functional equivalence between the behavior and synthesized RTL
 - Ensures blocks working together (e.g., no deadlock)
 - Generates waveforms for power analysis



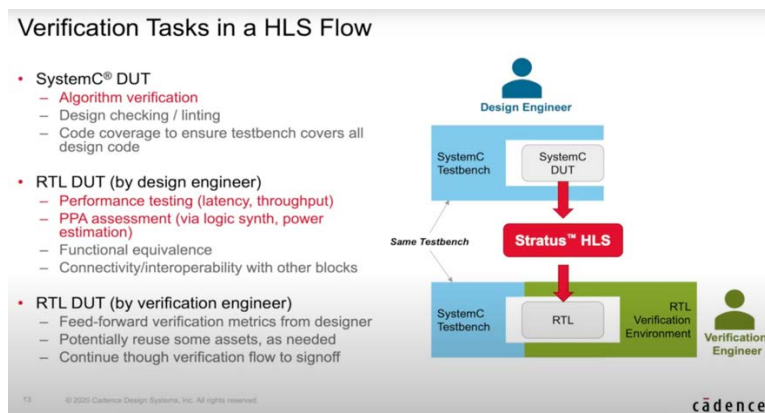
| Module | # of Blocks | # of FFs | Latency | Run Time |
|-----------------|-------------|----------|---------|----------|
| 0 - TLM | 1824 | 1824 | passed | 0:00:07 |
| 0 - PFM | 18781 | 18781 | passed | 0:00:08 |
| 0 - V FAST | 1,079,754 | 992 | 99428 | 0:02:03 |
| DmV1 RTL V FAST | 1 | 27,416 | 882 | 0:00:21 |
| PmV1 RTL V FAST | 1 | 17,432 | 1,576 | 0:00:45 |
| CmV2 RTL V FAST | 1 | 426,424 | 8,228 | 0:04:58 |
| PmV2 RTL V FAST | 1 | 48,759 | 4,837 | 0:01:02 |
| FCL RTL V FAST | 1 | 367,269 | 20,383 | 0:10:15 |
| GL RTL V FAST | 1 | 199,338 | 7,434 | 0:05:37 |

© 2020 Cadence Design Systems, Inc. All rights reserved. cadence

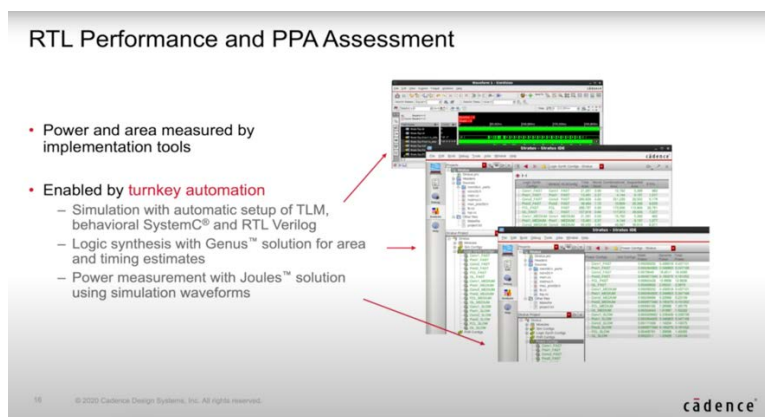
24 Source: <https://www.youtube.com/watch?v=uxhIFYZ8iC0>

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

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28



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



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

Design Space Exploration Results

- Multiple bit widths implemented for each SystemC® module
 - Larger bit widths give better accuracy
 - Smaller bit widths give smaller hardware with reduced power consumption
- Multiple synthesis configurations for each SystemC module
 - Tradeoff latency and hardware area
 - Slow and small vs fast and large
- Easy to find the best solution that meets your requirements

| Parameters | | Results | | | |
|---------------------|-----------------------|------------|-----------------|-------------|-------------------------------------|
| Bit Width | Speed Grade (latency) | Power (mW) | Images / Second | Area (Kum²) | Accuracy (%) |
| 16 | FAST | 63.0 | 5,294 | 103.5 | |
| | MED | 38.9 | 2,454 | 68.8 | 96.68% |
| | SLOW | 11.4 | 234 | 47.6 | |
| 15 | FAST | 94.7 | 5,664 | 113.4 | |
| | MED | 38 | 2,454 | 68.8 | 19% smaller for only 14% less power |
| | SLOW | 11 | 234 | 47.6 | |
| 14 | FAST | 81 | 2,454 | 68.8 | 0.64% reduction in accuracy |
| | MED | 11.3 | 234 | 47.6 | |
| | SLOW | 76.9 | 6,026 | 93.8 | |
| 13 | MED | 32.5 | 2,460 | 55.6 | 96.04% |
| | SLOW | 10.6 | 234 | 38.9 | |
| | FAST | 62.2 | 5,961 | 81.0 | |
| 12 | MED | 26.2 | 2,454 | 49.9 | 91.45% |
| | SLOW | 9.9 | 234 | 36.2 | |
| | FAST | 62.2 | 5,961 | 81.0 | |
| Range of trade-offs | | 9.6x | 25.8x | 3.1x | |

© 2020 Cadence Design Systems, Inc. All rights reserved. cadence

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

1 state in the RTL description.

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

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

8 Design Closure

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

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

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

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

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

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

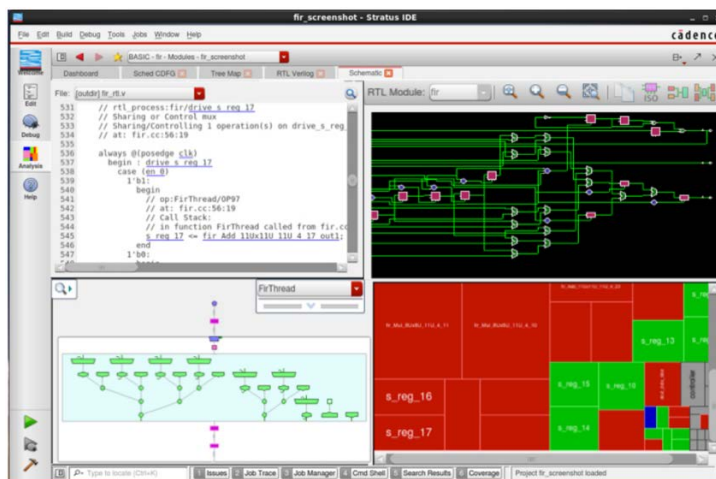
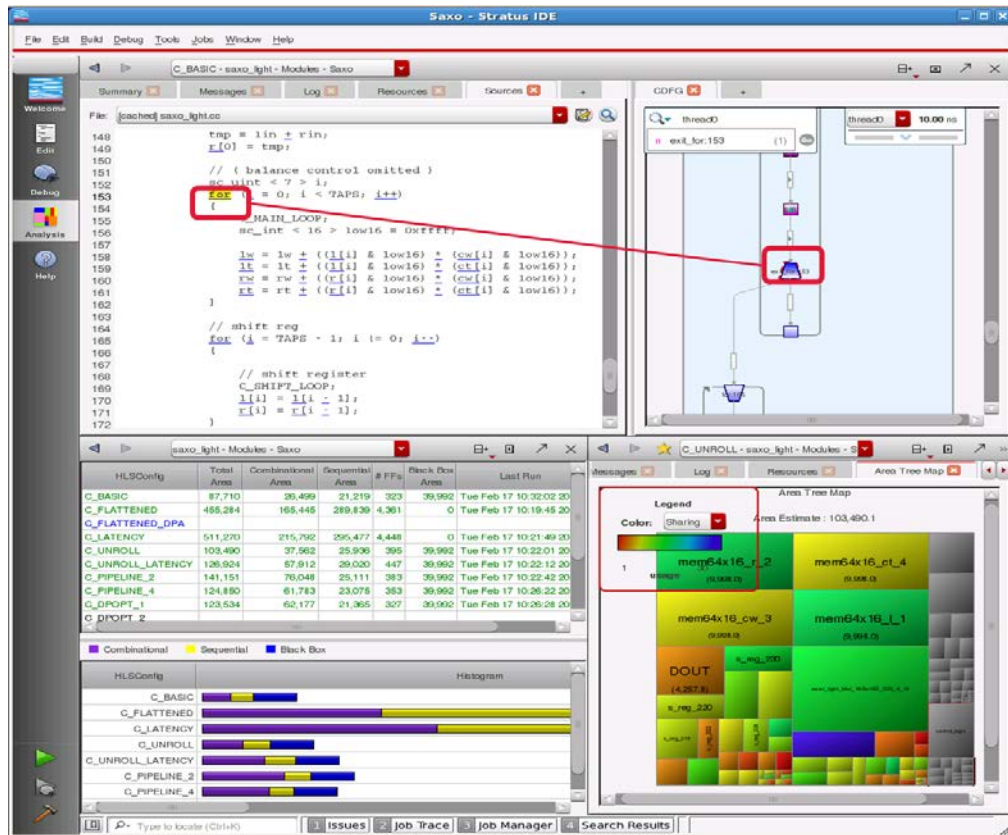


Figure 2: Complete graphical analysis with links to source code

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.

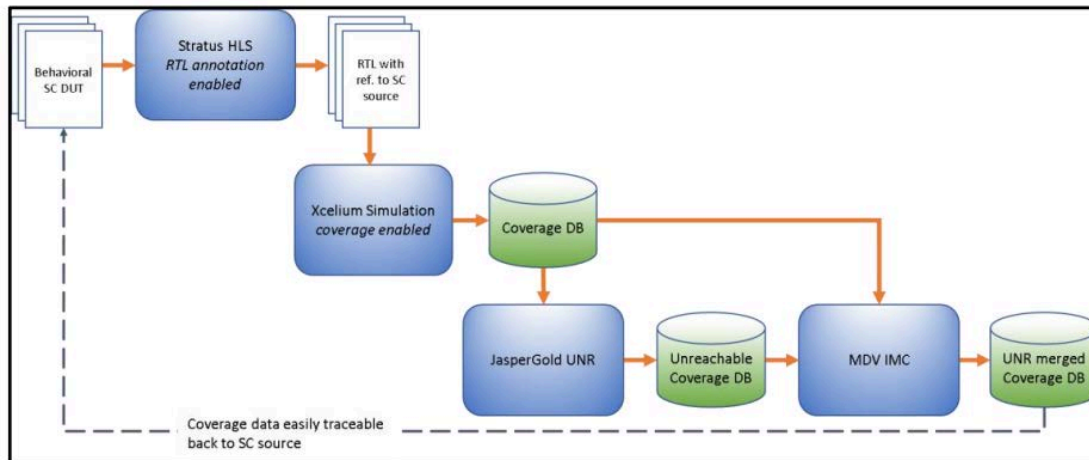


Figure 8: UNR analysis flow with traceability back to SC source

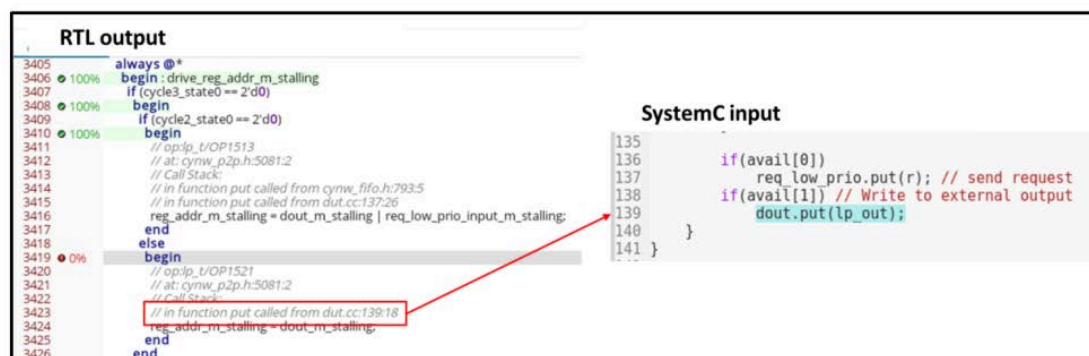
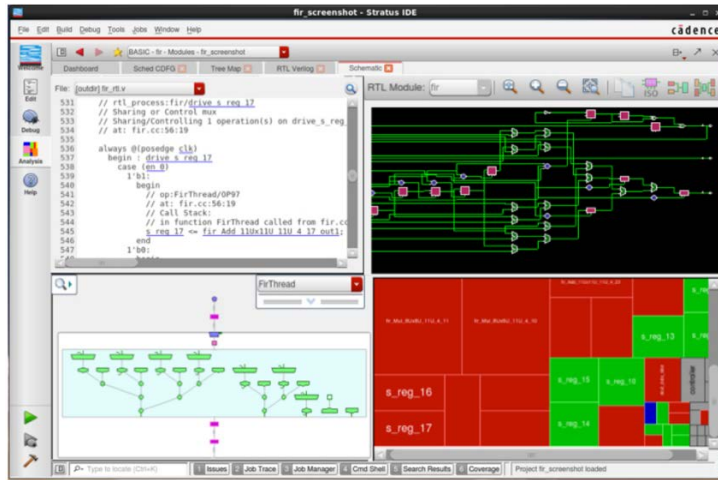


Figure 9: RTL annotation for traceability back to originating SystemC code lines

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.pdf>.

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



11 *Figure 2: Complete graphical analysis with links to source code*

12 Source: https://login.cadence.com/content/dam/cadence-13 www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf

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

17 Design Closure

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

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

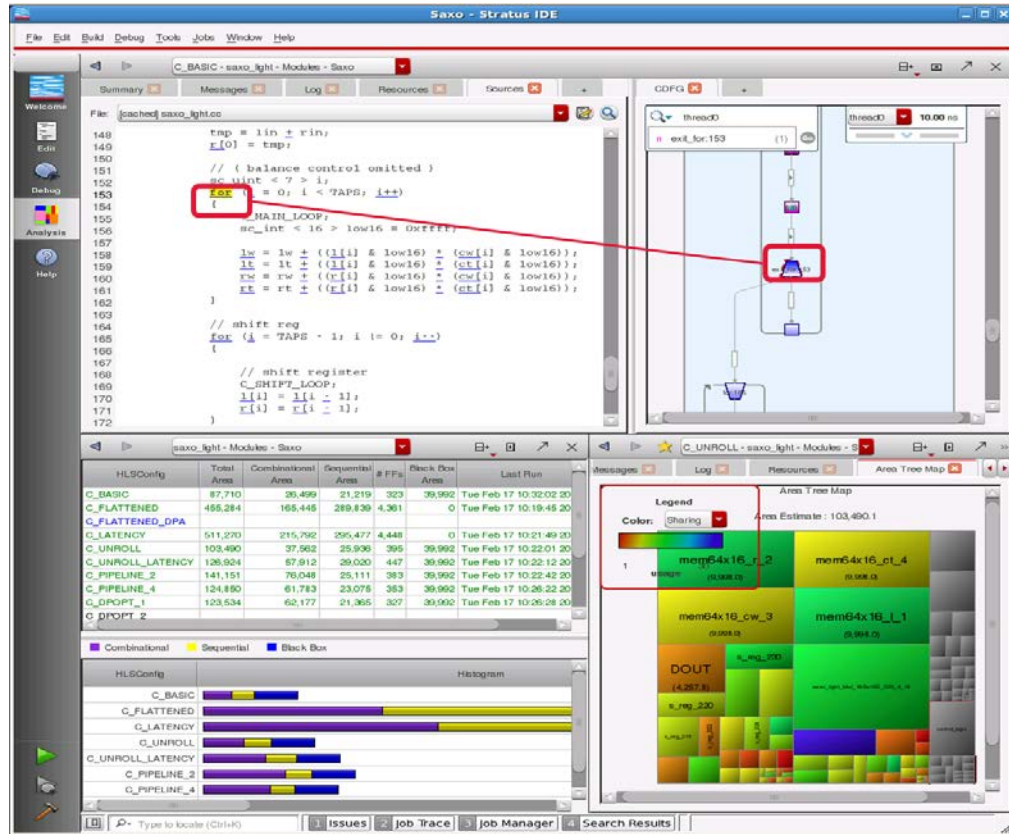
26 The user can control how aggressively Stratus HLS packs
27 these operations into each clock period. Creating designs
28 with Stratus HLS can save months of back-end effort by
preventing timing closure problems.

Source:

29
30
31
32
33
34
35
36
37
38

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.

https://login.cadence.com/content/dam/cadence-39 www/global/en_US/documents/tools/digital-40 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.

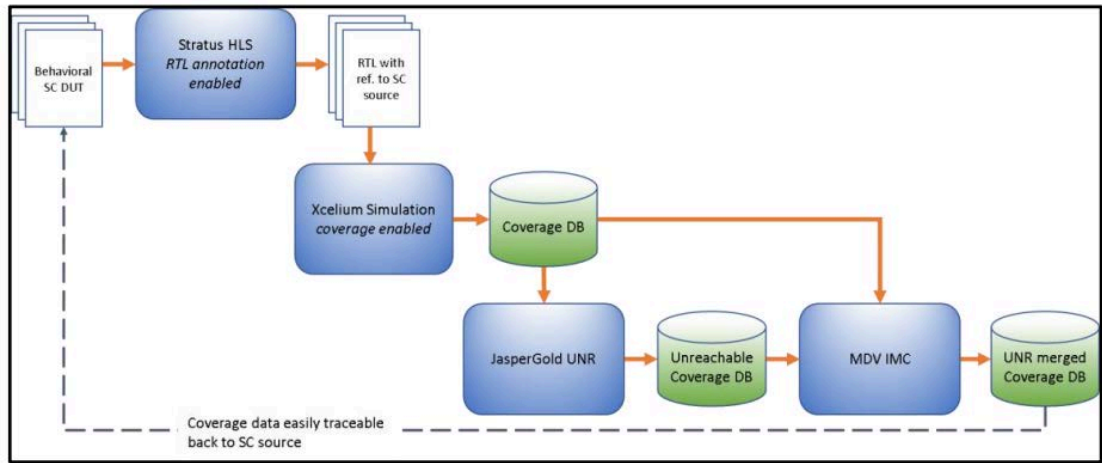


Figure 8: UNR analysis flow with traceability back to SC source

| RTL output | SystemC input |
|--|---|
| <pre> 3405 always @* 3406 begin : drive_reg_addr_m_stalling 3407 if (cycle3_state0 == 2'd0) 3408 begin 3409 if (cycle2_state0 == 2'd0) 3410 begin 3411 // op_ip_t/OP1513 3412 // at: cynw_p2p.h:5081:2 3413 // Call Stack; 3414 // in function put called from cynw_fifo.h:793:5 3415 // in function put called from dut.cc:137:26 3416 reg_addr_m_stalling = dout_m_stalling req_low_prio_input_m_stalling; 3417 end 3418 else 3419 begin 3420 // op_ip_t/OP1521 3421 // at: cynw_p2p.h:5081:2 3422 // Call Stack; 3423 // in function put called from dut.cc:139:18 3424 reg_addr_m_stalling = dout_m_stalling; 3425 end 3426 end </pre> | <pre> 135 136 if(avail[0]) 137 req_low_prio.put(r); // send request 138 if(avail[1]) // Write to external output 139 dout.put(lp_out); 140 } 141 } </pre> |

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

1 the Accused Products, providing instruction materials, support training, and services regarding the
2 Accused Products, and actively inducing its customers to acquire and/or install the infringing
3 products, including Stratus HLS software, on customer-provided computer-readable media to be
4 used in connection with a computer in an assertion-generating system for generating assertion
5 descripts for validation of a semiconductor integrated circuit. *See, e.g.,*
6 [https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-](https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html)
7 [level-synthesis.html](https://www.cadence.com/en_US/home/support.html); https://www.cadence.com/en_US/home/support.html, including all related
8 domains and subdomains. Cadence does so knowing that its customers will commit these
9 infringing acts. Despite its knowledge of the '167 patent, Cadence continues to make, use, sell,
10 and/or offer for sale its infringing products thereby specifically intending for and inducing its
11 customers to infringe the '167 patent. Those customers include Intel, Qualcomm, Socionext,
12 Syntiant, Himax, and Methods2Business. [https://www.cadence.com/en_US/home/tools/digital-](https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html)
13 [design-and-signoff/synthesis/stratus-high-level-synthesis.html](https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html).

14 94. Upon information and belief, Cadence has also indirectly infringed and continues to
15 indirectly infringe at least claim 1 of the '167 patent in violation of 35 U.S.C. §271(c) by
16 contributing to the infringement by its customers. Those customers include Intel, Qualcomm,
17 Socionext, Syntiant, Himax, and Methods2Business.

18 [https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-](https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html)
19 [level-synthesis.html](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,
20 including the Stratus HLS software, with knowledge that they are especially designed or adapted
21 to operate in a manner that infringes that patent and despite the fact that the infringing technology
22 or aspects of the 167 Accused Products are not a staple of commerce suitable for substantial non-
23 infringing use. For example, Cadence is aware that the Stratus HLS software operates as described
24 above and that such functionality infringes the '167 patent, including claim 1. Cadence is aware
25 that the Stratus HLS software infringes when installed and/or used because it enables a computer
26 to provide a semiconductor design support device for designing a semiconductor integrated circuit,
27 that such installation and/or use infringes the '167 patent, including claim 1, and that the Accused
28 Products have no substantial non-infringing use. The portion of the Stratus HLS software that maps

1 to claim 1 (i.e., the infringing aspect) has no substantial non-infringing uses.

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

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

6 **PRAYER FOR RELIEF**

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

- 9 a. That U.S. Patent No. 7,603,636 be judged valid, enforceable, and infringed by
10 Defendant.
11 b. That U.S. Patent No. 7,971,167 be judged valid enforceable, and infringed by
12 Defendant.
13 c. That Plaintiff be awarded judgment against Defendant for damages together with
14 interests and costs fixed by the Court including an accounting of all infringements
15 and/or damages not presented at trial; and
16 d. That Plaintiff be awarded such other and further relief as this Court may deem just
17 and proper.
18

19 **JURY DEMAND**

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

21
22 Date: June 29, 2023

Respectfully submitted,

23 /s/ Robert F. Kramer

24 Robert F. Kramer

25 **KRAMER ALBERTI LIM &
26 TONKOVICH LLP**