IC Validator
Functional Safety Manual

March 2018, Revision 1.4

SYNOPSYS®

Copyright and Proprietary Information Notice

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA, 94043
www.synopsys.com

Document Control

Revision history

Version	Description	Date
1.0	First release of the document submitted for review.	15-Jan-2018
1.1	Added revision history, fixed template issues.	06-Feb-2018
1.2	Fixed boilerplate changes from general feedback.	01-Mar-2018
1.3	Added Appendix B and Master STAR list link	02-Mar-2018
1.4	Content updated based on certification review	09-Mar-2018

Contents

1
Customer Support

This section describes the customer support that is available through the Synopsys SolvNet® customer support website or by contacting the Synopsys support center.

Accessing SolvNet

The SolvNet support site includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The site also gives you access to a wide range of Synopsys online services, which include downloading software, viewing documentation, and entering a call to the Support Center.

To access the SolvNet site:

1. Go to the web page at https://solvnet.synopsys.com.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register.)

If you need help using the site, click **Help** on the menu bar.

Contacting Synopsys Support

If you have problems, questions, or suggestions, you can contact the Synopsys support center in the following ways:

- Go to the Synopsys Global Support Centers site on synopsys.com. There you can find e-mail addresses and telephone numbers for Synopsys support centers throughout the world.
- Go to either the Synopsys SolvNet site or the Synopsys Global Support Centers site and open a case online (Synopsys user name and password required).

2 Scope of This Document

This section describes the scope of this document and defines terms used in this document.

Using This Document

The *IC Validator Functional Safety Manual* describes the proper use of the IC Validator tool in safety-related applications according to the ISO 26262 standard, and is intended to confirm the compliance of the IC Validator tool to the standard when used in the context of a tool chain.

The IC Validator physical verification tool provides signoff solutions for design rule checking, connectivity verification, layout versus schematic, and metal fill insertion. Design rule checking (DRC) covers the ability of the foundry to physically fabricate the chip, based on geometric interactions. Electrical rule checking (ERC) addresses electrically charged build-up during fabrication so that it does not discharge in a way that harms the design, and it covers certain electrical state circuitry conditions. Layout versus schematic (LVS) provides a mechanism for creating a netlist based on the design's polygons, and comparing it to the original netlist to ensure the design is as intended. Metal fill provides the insertion of shapes on layers to create the desired density of polygons to ensure planar creation of layers during fabrication.

Section 3 describes an overview of the ISO 26262-8, clause 11 and the approach adopted by Synopsys to comply with the requirements of the standard. Section 4 defines the general information such as where to find the latest documentation and installation requirements regarding the use of the IC Validator tool as a software tool in the development of safety-related applications. Section 5 shows the high-level overview of the tool chain that this product belongs to. Section 6 details the safety-related requirements for safety-qualified use cases of the IC Validator tool. Section 7 lists the known limitations of the use cases.

Specific documentation for performing design and analysis as part of an ISO 26262 compliant flow is provided in Section 3, Section 5, Section 6, Appendix A , and Appendix B of this document, the *IC Validator Functional Safety Manual*.

Terms and Definitions

Term	Definition
AoU	Assumption of Use. An action that is assumed and required to be taken by the user of a software tool.

Term	Definition
ASIL	Automotive Safety Integrity Level. This is a risk classification scheme defined by the standard ISO 26262. The standard identifies four levels: ASIL A, ASIL B, ASIL C, and ASIL D. ASIL D dictates the highest integrity requirements on a product and ASIL A dictates the lowest.
Component	A part of an electronic system that implements a function in a vehicle. See also Part 1 of the standard ISO 26262 for the definition. The standard also refers to elements and items, but for the *IC Validator Functional Safety Manual*, there is no difference.
CoU	Condition of Use. A condition of the design, software tool, design environment, or situation that is assumed and required to be fulfilled by the user.
CRM	Customer Relationship Management. Internal Synopsys database that manages customer STARs.
Defect	Product nonconformance.
DRC	Design Rule Checks
ERC	Electrical Rule Checks
Fault analysis	An analysis that determines the behavior of a system when a fault is introduced.
FMEA	Failure Mode and Effects Analysis. An analysis that looks at different parts of a system, identifies ways the parts could fail, and determines the causes and effects of these potential failures.
GDSII	Input layout file format
ICV	IC Validator
LVS	Layout Versus Schematic
Milkyway	Input layout library format
NDM	Input layout library format
OASIS	Input layout file format
OpenAccess	Input layout library format

Term	Definition
Software / software tool	The IC Validator tool
Software tool criteria evaluation	Analysis according to ISO 26262 to determine the required TCL of a software tool.
Software tool qualification	Means to create evidence, that a software tool with low or medium TCL is suitable to be used in the development of safety related products according to ISO 26262.
SolvNet	Synopsys customer support site.
Standard	In this document, refers to *ISO 26262 Road Vehicles – Functional Safety*, 2011 and 2018 versions.
STAR	Synopsys Technical Action Request. A STAR documents and tracks a product Bug or Enhancement request (called a B-STAR or an E-STAR, respectively). It is stored in the Synopsys CRM database. Only Synopsys employees can access the CRM database. However, limited STAR information is available from SolvNet for customers who are associated with the user site of a STAR. Customer contacts are notified automatically when a STAR is filed or when its status changes.
TCL	Tool confidence level, as defined by ISO 26262-8, clause 11. Note: The TCL of a software tool does not necessarily indicate whether the tool may malfunction or not. The TCL defines the confidence level that an error in the safety-related design, which is introduced or left undetected by the software tool, can be prevented or detected in subsequent steps of the development flow, before the erroneous safety-related design is released.
TD	Tool error detection, as defined in ISO 26262-8, clause 11.
TI	Tool impact, as defined in ISO 26262-8, clause 11.
Use case	A use case is a specific way of using a software tool, that can be characterized by: - a limited set of tool functions and features that are used; - a set of restrictions and constraints that are regarded while using the tool; and - a specific goal to be achieved or output to be generated by using the software tool

Term	Definition
	Use cases may be associated with different steps or phases in the design process, or they may describe alternative ways of using the tool for a specific design step.

3 Confidence in the Use of Software Tools According to ISO 26262-8, Clause 11

This section provides an overview of the ISO 26262-8, clause 11. It then describes the approach adopted by Synopsys to comply with the requirements of the standard, and how this is mapped to activities performed by Synopsys and the end user of the Synopsys tools.

Overview of ISO 26262-8, Clause 11

Synopsys EDA software tools contribute significantly to the design specification, implementation, integration, verification and validation of electrical and electronic (E/E) systems and components. If these E/E systems and components are used as part of a safety-related automotive product, an error in these systems or components could have severe consequences on functional safety. Such an error may arise as a result of unforeseen operating conditions or due to a fault introduced during product development, which in turn may be caused by a software tool malfunction. ISO 26262-8, clause 11 (Confidence in the Use of Software Tools) addresses this issue and specifies requirements and methods which aim to minimize the risk of faults in the developed product due to malfunctions of a software tool affecting the product's functional safety.

According to ISO 26262, to determine the required level of confidence in a software tool that is used in the development of a safety-related automotive product, the following criteria are evaluated:

- The possibility that the malfunctioning software tool and its corresponding erroneous output can introduce or fail to detect errors in a safety-related element being developed.
- The confidence in preventing or detecting such errors in its corresponding output.

This procedure is called Software Tool Criteria Evaluation, and it must be performed for all software tools that are involved in the development of a safety-related element, resulting in a required Tool Confidence Level (TCL) for each software tool.

If the software tool criteria evaluation determines that a medium or high TCL is required, then appropriate Software Qualification methods must be applied, effectively reducing the risk of a critical software tool error. The choice of software qualification methods depends on the required TCL and the maximum ASIL of all the safety requirements allocated to the element developed using the software tool. However, if the software tool criteria evaluation determines that only a low TCL is required, then there is no need to apply such software qualification methods.

The software tool criteria evaluation and software tool qualification flow is summarized in Figure 1.

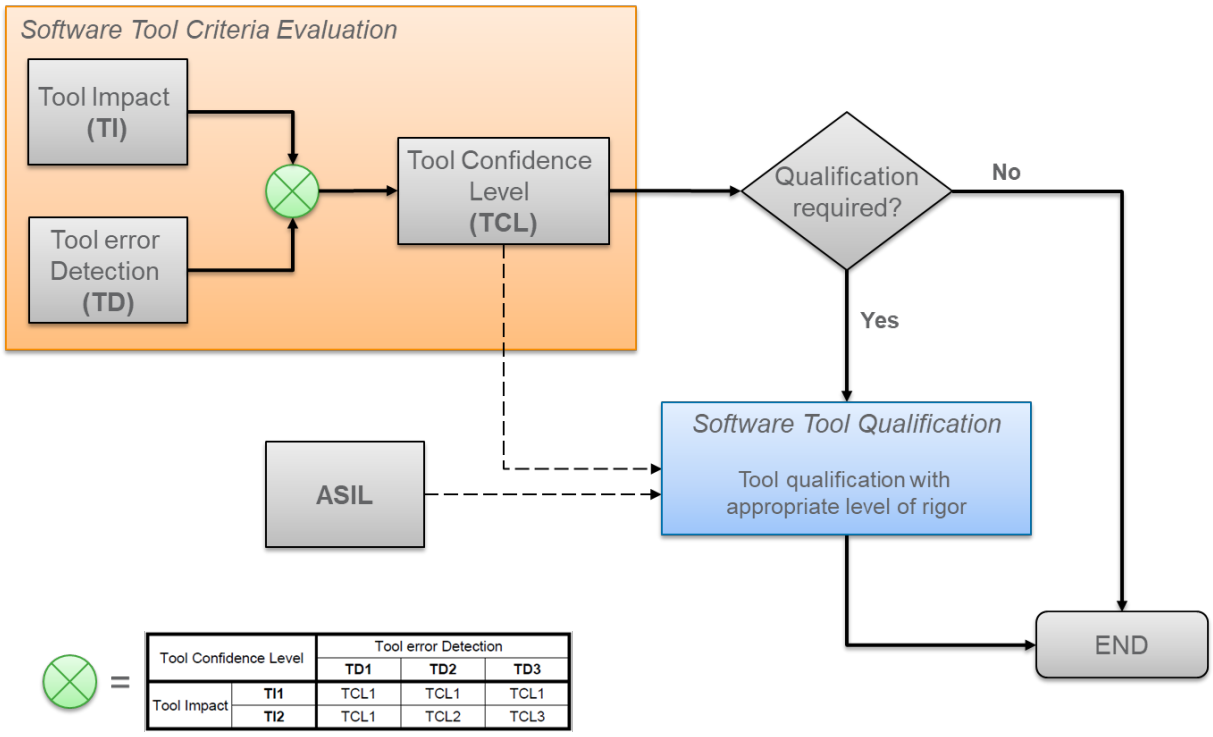


Figure 1: Software tool criteria evaluation and software tool qualification flow

Work Split Between Synopsys and Tool Users

A software tool criteria evaluation must always be performed in the development environment of the final tool user, and in the context of the actual product development. It is in this context, where potential tool malfunctions, their effect on the safety-related product, and the effectiveness of prevention and detection measures must be analyzed.

However, the tool vendor can support the tool user by performing a software tool criteria evaluation (and, if required, a software tool qualification) on their own, based on assumed tool use cases and an assumed development environment. If the assumptions made by the tool vendor match the actual situation at the tool user, then the user can take over the evaluation (and qualification) results from the tool vendor. Besides significantly reducing the effort for the tool user, this approach can also result in a better quality for the software tool criteria evaluation and qualification, since the tool vendor typically has a more detailed understanding of the inner working and possible malfunctions of the software tool.

Synopsys has adopted exactly this approach, which is summarized in Figure 2.

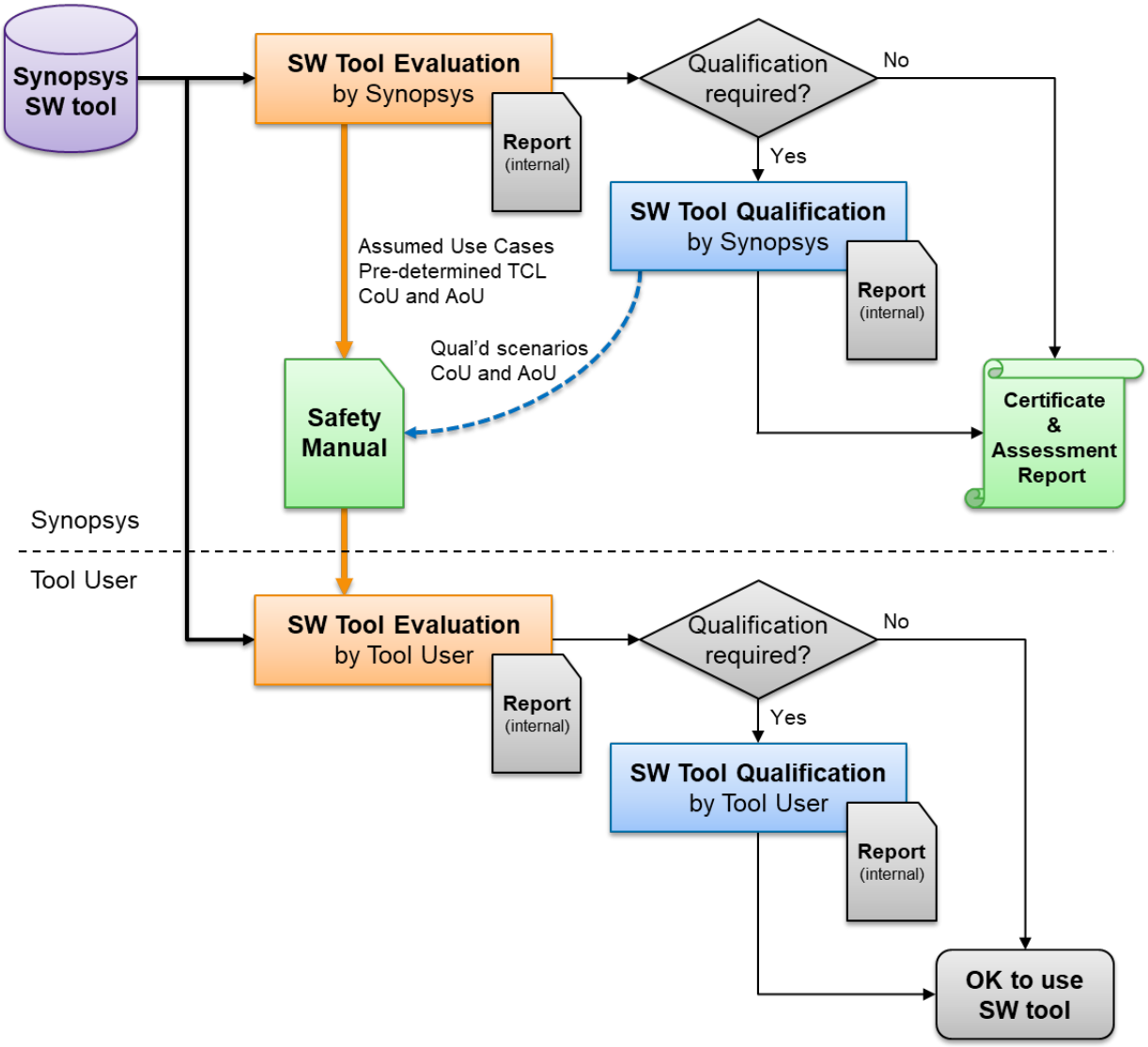


Figure 2: Work Split between Synopsys and Tool Users

Synopsys performs the following activities:

1. Software tool criteria evaluation
   - Identification of possible **use cases** for the software tool, together with required **inputs** and expected **outputs**
   - Specification of **conditions of use (CoU)** for each use case, related to the development environment in which the tool is assumed to be deployed, including tool usage procedures and constraints
   - Analysis of potential software tool **malfunctions**, and their effect on a safety-related product that is developed with this tool
   - Analysis of **prevention** and **detection measures** internal to the software tool, to avoid tool malfunctions, or to control and mitigate their effects
   - Specification of **assumptions of use (AoU)**, which are additional prevention and detection measures assumed to be performed by the end user of the tool

- Estimation of the **Tool Impact (TI)** for each malfunction, and the probability of **Tool error Detection (TD)** by the prevention and detection mechanisms (including assumptions of use)
- Determination of the required **Tool Confidence Level (TCL)** for each software tool malfunction, based on TI and TD
- Determination of the maximum TCL from all software tool malfunctions related to a use case. This is called the **pre-determined TCL** for the software tool use case
- Summary of the results in a software tool criteria evaluation report

2. Software tool qualification
   - If the pre-determined TCL indicates, that a medium (TCL2) or high (TCL3) tool confidence level is required for the software tool, then Synopsys may decide to perform a software tool qualification
   - The specific methods applied for tool qualification can vary for different tools and use cases, and they may include an evaluation of the software tool development process, the validation of the complete software tool, the validation of critical tool malfunctions with insufficient prevention and detection measures, or other methods
   - Summary of the qualification methods, procedures and results in a software tool qualification report

3. Safety manual for the software tool
   - The *IC Validator Functional Safety Manual* (this document) is an important deliverable to the tool users, as it includes all end user-relevant information from the Synopsys software tool criteria evaluation and qualification
   - Software tool criteria evaluation related information, documented in Section 6, includes:
     - Description of software tool use cases
     - Description of the required inputs and expected outputs for each use case
     - Specification of conditions of use (CoU – conditions of the design, software tool, design environment, or situation that are assumed and required to be fulfilled by the user) for each use case
     - Specification of assumptions of use (AoU – actions that are assumed and required to be taken by the user of a software tool) for each use case
     - Pre-determined TCL for each use case
   - Software tool qualification related information (not required for the IC Validator tool and therefore not included in this safety manual)
     - Description of the scope of the software tool qualification, including malfunctions and scenarios covered by the qualification
     - Specification of additional conditions of use (CoU) derived from the software tool qualification
     - Specification of additional assumptions of use (AoU) derived from the software tool qualification
   - Other information included in this safety manual
     - General information about the software tool needed by the tool user (see Appendix A)
     - Known limitations of the software tool, related to the described use cases as documented in Section 7

4. Certification and assessment report
   - Synopsys may decide to perform a functional safety assessment, to confirm the correctness, completeness and ISO 26262 conformance of the performed software tool criteria evaluation and qualification
   - Synopsys may also decide to achieve certification from an accredited third-party certification body, in addition to the functional safety assessment
   - The results of these activities are summarized in a functional safety assessment report and a certificate which can be viewed at exida Certificate for ISO 26262 Compliance

If the tool user wants to benefit from the work done by Synopsys, then according to the Figure 2 above, the user shall perform the following activities for each software tool:

1. Software tool criteria evaluation
   - Review and verify that the software tool criteria evaluation (and qualification) performed by Synopsys, as documented in the tool‘s Functional Safety Manual, matches the actual situation of the user’s product development process
     - Verify whether the actual use case(s) of the software tool match those evaluated by Synopsys
     - Verify whether the actual inputs and outputs are identical to or a sub-set of those as evaluated by Synopsys
     - Verify that all conditions of use (CoU) specified by Synopsys are met, or whether the development process can be adjusted to meet these CoU(s)
     - Verify that all assumptions of use (AoU) specified by Synopsys are met, or whether the development process can be adjusted to meet these AoU(s)
     - Verify that the pre-determined Tool Confidence Level (TCL) for the relevant use case(s) are TCL1, *or*
     - Verify that Synopsys has successfully performed an additional software tool qualification for all TCL2 and TCL3 scenarios to conclude that the tool is suitable to be used for the development of a safety-related element of the same or higher ASIL than required by the user
   - If all the verification steps described above are successful, then the results of the Synopsys software tool criterial evaluation (and qualification) are applicable to the tool user, which means:
     - The required TCL pre-determined by Synopsys can be taken over by the tool user for actual product development
     - If the pre-determined TCL is TCL1, then the tool can be used without the need to perform any additional software tool qualification
     - If the pre-determined TCL is TCL2 or TCL3, then the software tool qualification performed by Synopsys is sufficient, and the tool can be used without the need for further software tool qualification by the end user
   - All of the steps above must be documented in a software tool criteria evaluation report, including evidence for the successful conclusion of all verification steps, which may include reference to the Synopsys Functional Safety Manual, and optionally, to the Synopsys certification and assessment report

2. Software tool qualification
    - If any of the verification steps described above as part of the tool user's software tool criteria evaluation fails (e.g. different use case, CoU or AoU cannot be met, pre-determined TCL is not TCL1 and Synopsys has not performed a software tool qualification), then the user must perform his/her own software tool qualification
    - The specific methods applied for tool qualification are decided and planned by the tool user -- Synopsys does not recommend any specific methods or procedures
    - The summary of the qualification methods, procedures and results shall be documented in a software tool qualification report

4 IC Validator Description

This section provides a general description regarding the use of the IC Validator tool as a software tool in the development of safety-related applications and describes where to get the latest product documentation and the runtime environment required to use the IC Validator tool.

Coverage

The *IC Validator Functional Safety Manual* is intended to be used starting with the N-2017.12 and later versions of the IC Validator tool per the use cases presented in this document. In general, unless otherwise noted, the failure modes and detection mechanisms noted in the use cases presented in Section 6 are tool version independent.

Compliance with ISO 26262

The IC Validator tool can be used in the development of safety-related elements according to ISO 26262, with allocated safety requirements up to a maximum Automotive Safety Integrity Level D (ASIL D), if the tool is used in the context of a tool chain and in compliance with this document, the *IC Validator Functional Safety Manual*.

See the exida Certificate for ISO 26262 Compliance of Synopsys IC Validator when used in a tool chain flow.

Product Documentation and Support

Comprehensive documentation for using the IC Validator tool is provided on SolvNet. The latest documentation for the IC Validator tool can be accessed at IC Validator Online Help on SolvNet.

Specific documentation for performing design and analysis as part of an ISO 26262 compliant flow is provided in Section 3, Section 5, Section 6 and Appendix A of this document, the *IC Validator Functional Safety Manual*.

Synopsys provides online customer support for the IC Validator tool. See Section 1 for more information.

Installation and Supported Platforms

The installation of the IC Validator tool must follow the guidelines in the *Synopsys® Installation Guide* as well as the specific *IC Validator Installation Notes* document.

Users are required to download the tool executable and INSTALL_README from the SolvNet site at https://solvnet.synopsys.com/DownloadCenter/dc/product.jsp.

Supported platforms and operating systems requirements:

- For installation instructions, see the *Synopsys® Installation Guide* at https://www.synopsys.com/install.
- For the latest supported binary-compatible hardware platform or operating system, including required operating system patches, see https://www.synopsys.com/qsc.
- If updates (including security patches) to computing environments (including operating systems) are backward compatible with previous versions of the computing environment used to test the IC Validator tool, the results of the testing performed by Synopsys using such previous versions are applicable.

Additional information:

- For information about the compute platforms roadmap, go to https://www.synopsys.com/support/licensing-installation-computeplatforms/compute-platforms/compute-platforms-roadmap.html.
- For platform notices, go to https://www.synopsys.com/support/licensing-installation-computeplatforms/compute-platforms/platform-notice.html.
- For information regarding the license key retrieval process, go to https://solvnet.synopsys.com/smartkeys/smartkeys.cgi.

User Competence

To properly use the IC Validator tool, a user must have a good understanding and working knowledge of the following:

- Electrical engineering and circuit design
- The ISO 26262 standard
- Documentation of the IC Validator tool, such as the User Guide, at https://solvnet.synopsys.com/dow_retrieve/latest/ni/ dg/icvolh/Default.htm on SolvNet.
- This Functional Safety Manual
- The published list of safety-related defects for the IC Validator tool available at IC Validator Safety-Related Issues Master List. https://solvnet.synopsys.com/retrieve/2806205.html
- Applicability of the IC Validator tool in the overall tool chain

Managing Known Safety-Related Defects

Synopsys maintains current information for every reported defect through STARs. The IC Validator team evaluates each reported issue for potential impact on functional safety.

A list of all known safety-related defects for each release of IC Validator is available on a SolvNet knowledge base article and is referenced from the *IC Validator Release Notes*.

IC Validator users must assess, as part of their own software tool criteria evaluation, the potential impact of the known safety-related defects in their design and must ensure mitigation of any relevant safety-related defects.

Managing New Releases

Synopsys can release new versions of the IC Validator tool at any time to extend its functionality or to fix defects. When a new version is available, notification is posted on the SolvNet site. A subscription service is available for users to be notified of any new product releases.

When installing a new version of the IC Validator tool, users must evaluate the impact of any known safety-related defects in their design by checking the accompanying *IC Validator Release Notes* for the following:

- Any changes that apply to safety-related use cases
- List of known safety-related defects in the new version of the IC Validator tool

In addition, users must refer to the latest version of this document, the *IC Validator Functional Safety Manual*, available with the product release contents.

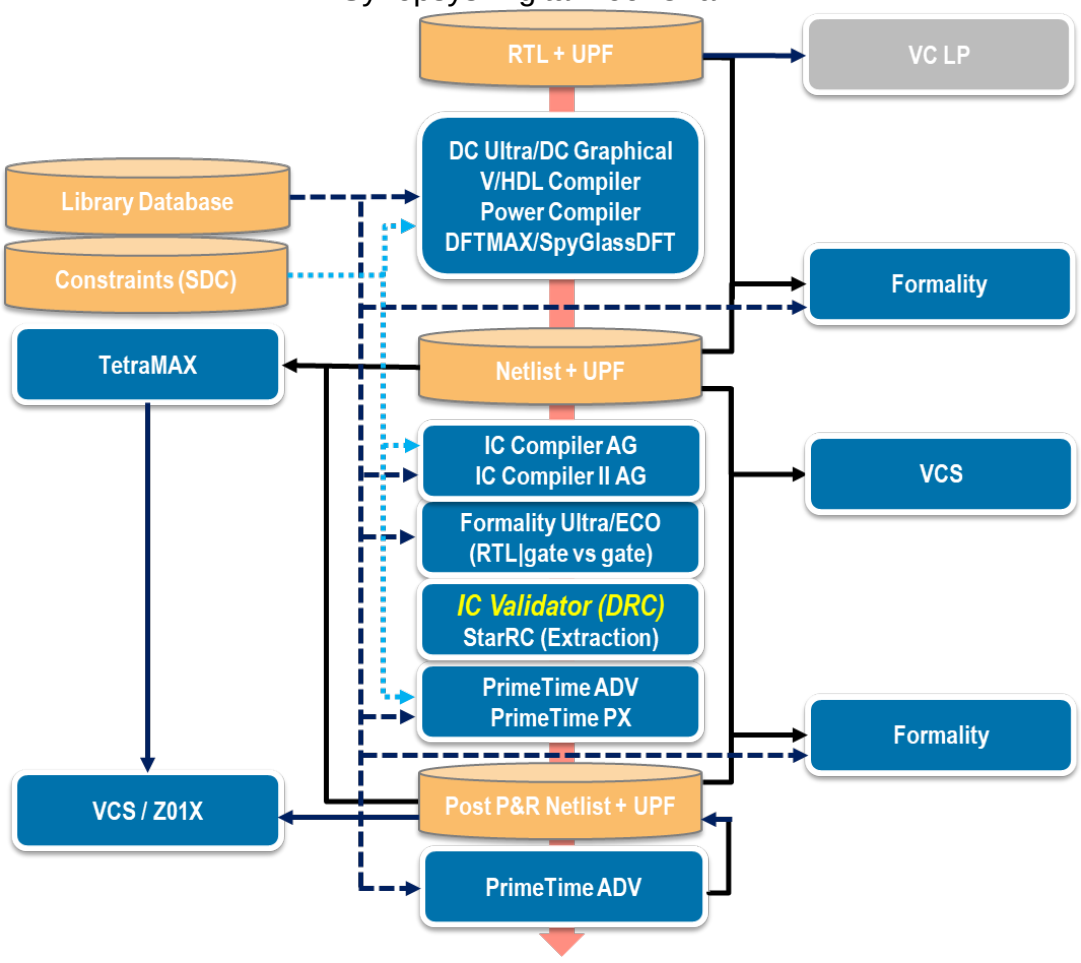
5 Synopsys Digital and Analog Tool Chains

This section provides an overview of where the IC Validator is used in the tool chain.

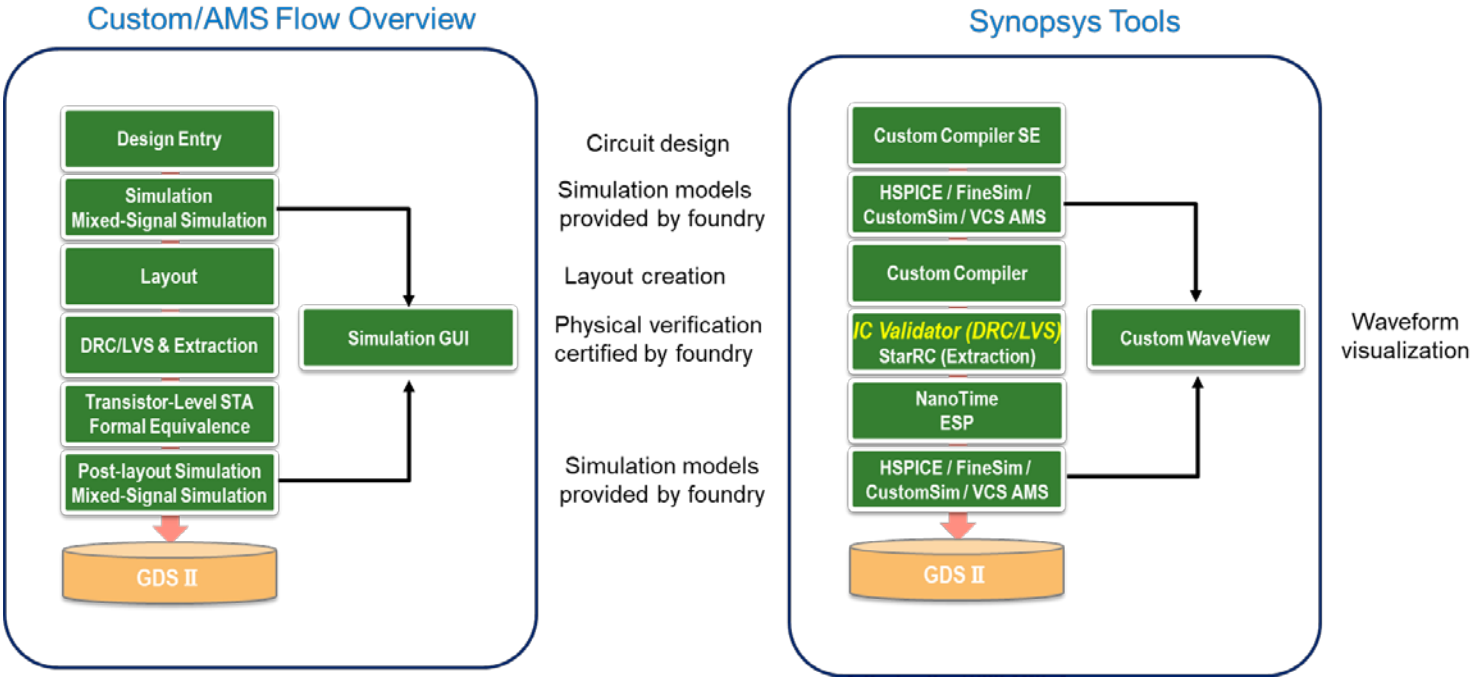
The ISO 26262 standard provides a methodology and requirements for software tool criteria evaluation and qualification (see ISO 26262-8, clause 11). It applies to software tools used for the development of safety-related designs where it is essential that the tool operates correctly without introducing or failing to detect errors in the safety-related design.

The suitability of a software tool to be used in the development of a safety-related design is determined in the software tool criteria evaluation, which results in a Tool Confidence Level (TCL): a level of confidence that the software tool does not introduce or fail to detect an error in the design without being noticed, and mitigated before the design is released as a safety-related product. This evaluation is best performed in the context of the overall software tool chain and development flow, in which the individual software tool is used. The following high-level diagrams reflects the tool chains for which the IC Validator tool is applicable.

Synopsys Digital Tool Chain



Synopsys Analog/Mixed-Signal Tool Chain



6
Use Cases

This section describes the safety-qualified use cases of the IC Validator tool. Users should also perform TCL determination based on their specific Use Cases.

The IC Validator physical verification tool provides signoff solutions for design rule checking, connectivity verification, layout versus schematic, and metal fill insertion.

The following use cases cover different aspects of physical verification. While the exact operations performed are based on how the foundry codes the corresponding runsets, the basic functions are as follows:

- DRC covers the ability of the foundry to physically fabricate the chip based on geometric interactions.
- ERC addresses electrically charged build-up during fabrication so that it does not discharge in a way that harms the design, and it covers certain electrical state circuitry conditions.
- LVS provides a mechanism for creating a netlist based on the design's polygons, and comparing it to the original schematic netlist to ensure the design is as intended.
- Metal fill provides the insertion of shapes on layers to create the desired density of polygons to ensure planar creation of layers during fabrication.

Each of these use cases can be performed in any order, iterating as the design evolves. However, as metal fill alters the polygons in the design, the user shall perform DRC, ERC, and LVS at least one final time after metal fill is performed.

Use Case 1: DRC (Design Rule Check) and ERC (Electrical Rule Check)

In this use case, the goal is to verify that the chip is able to be physically fabricated by the foundry. To accomplish this, rules are coded by the foundry to check for different physical orientations, proximities, size, and connections. Two inputs, the design and runset, are provided for this task. The rules coded in the runset are executed on the corresponding layers in the design, and the violations of those rules are written to the output files. Output files are to be reviewed and the design addressed as needed to correct such violations. Iterations are needed until no violations are reported.

In this use case, the IC Validator tool uses and generates the following main inputs and outputs.

- Inputs:
    - Layout (GDSII, OASIS, OpenAccess, NDM, Milkyway)

    - DRC and ERC runsets
- Expected outputs:
    - .RESULTS file
    - .sum (summary) file
    - Detailed DRC and ERC violations (*topcell*.LAYOUT_ERRORS)
    - .tree<###> (design hierarchy)

For this use case of *IC Validator*, the following conditions of use (constraints for the design and design environment, recommended procedures for the tool usage, etc.) shall be met:

- CoU-ICV-001: User shall review all error and warning messages and take appropriate action.
- CoU-ICV-002: User shall follow the *IC Validator User Guide* or use equivalent scripts.
- CoU-ICV-003: User shall perform DRC run on final layout (after metal fill).
- CoU-ICV-004: User shall not consider In-Design DRC as a final signoff DRC run.
- CoU-ICV-005: User shall not use selectable rules and selectable windows.

For this use case of *IC Validator*, the following assumptions of use (required actions to be taken by the tool user to prevent or detect design errors due to possible tool malfunctions) shall be met:

- AoU-ICV-001: User shall review .RESULTS and .sum files for error, warning, and abort messages. User shall confirm that correct layout and runset are used.
- AoU-ICV-002: User shall review .RESULTS file to determine whether it contains the expected steps, including design hierarchy identification, processing of polygons, and that all DRC rules are applied and executed.
- AoU-ICV-003: User shall review output file timestamps to ensure they are generated for the current run and dated more recently than the run initiated, as listed in the .RESULTS file.
- AoU-ICV-009: User shall compare and check for consistency in DRC output files between the "ERROR SUMMARY" section of the .LAYOUT_ERRORS file and the "WARNING ... violation(s) found" entries in the .sum file to ensure complete output. In case of any discrepancies, user shall rerun DRC.
- AoU-ICV-010: User shall check for consistency between the .LAYOUT_ERRORS ASCII violation report and the corresponding "DRC Errors" violation report in the VUE GUI. In case of any discrepancies, user shall rerun DRC.
- AoU-ICV-015: User shall review .RESULTS and .sum files for error, warning, and abort messages. User shall review tree files to ensure entire design hierarchy was processed.
- AoU-ICV-016: User shall ensure that the foundry has qualified the IC Validator tool and runset.

- AoU-ICV-017: User shall perform DRC checking in another tool, such as the IC Compiler or IC Compiler II tool.
- AoU-ICV-018: User shall perform Mask Rule Checking (MRC) on individual masks to catch any layout errors that may have escaped.
- AoU-ICV-019: User shall review steps in the .RESULTS file, including design hierarchy identification and polygon processing to ensure that all layers are included, no subset windows are selected, and all DRC rules are executed.

All analyzed failure modes and prevention, detection and mitigation measures (including conditions and assumptions of use listed above) are independent of the exact IC Validator tool version.

A software tool criteria evaluation performed by Synopsys according to ISO 26262-8, clause 11, which assumes the fulfillment of all conditions of use (CoU) and assumptions of use (AoU) as described above, results in a required tool confidence level:

TCL1 for *IC Validator* Use Case 1: DRC (Design Rule Check) and ERC (Electrical Rule Check)

In this case no further activities for software tool qualification are required.

Use Case 2: LVS (Layout Versus Schematic)

In this use case, the goal is to ensure that the circuitry created in the design's physical polygon layout matches the circuitry in the original schematic netlist. The foundry codes the runset to perform these tasks by way of two main sub-tasks, device extraction and compare.

During device extraction, devices are identified by recipes coded in the runset that are created through specific interaction of polygon layers. The connections between the devices are established by way of polygon layers that are identified as electrically or physically connected. Text is then applied. Upon completion, a netlist can be extracted reflecting this physical arrangement of polygons in the design.

During compare, this layout-extracted netlist can be compared to the original schematic netlist to ensure the design is as intended. Design errors can be identified during the device extraction and compare stages. Violations are written to the corresponding output files. Output files are to be reviewed and the design addressed as needed to correct such violations. Iterations are needed until no violations are reported.

In this use case, the IC Validator tool uses and generates the following main inputs and outputs.

- Inputs:
  - Layout (GDSII, OASIS, OpenAccess, NDM, Milkyway)
  - Schematic Netlist
  - LVS runset
- Expected outputs:
  - .RESULTS file
  - Device Extraction summary file (.sum)
  - Detailed LVS Layout and Device Extraction violations (*topcell*.LAYOUT_ERRORS)
  - Summary of LVS compare violations (*topcell*.LVS_ERRORS)
  - Compare summary file (*topcell*_lvs.log)

For this use case of *IC Validator*, the following conditions of use (constraints for the design and design environment, recommended procedures for the tool usage, etc.) shall be met:

- CoU-ICV-001: User shall review all error and warning messages and take appropriate action.
- CoU-ICV-002: User shall follow the *IC Validator User Guide* or use equivalent scripts.
- CoU-ICV-005: User shall not use selectable rules and selectable windows.
- CoU-ICV-006: User shall do LVS run on final layout (after metal fill).

For this use case of *IC Validator*, the following assumptions of use (required actions to be taken by the tool user to prevent or detect design errors due to possible tool malfunctions) shall be met:

- AoU-ICV-003: User shall review output file timestamps to ensure they are generated for the current run and dated more recently than the run initiated, as listed in the .RESULTS file.
- AoU-ICV-006: The design shall include design-for-test (DFT) functionality and ATPG testing shall be performed on manufactured devices.
- AoU-ICV-008: User shall review .RESULTS and .sum files for error, warning, and abort messages. User shall confirm that correct layout, schematic, and runset are used.
- AoU-ICV-011: User shall compare and check for consistency between the "FAILED equivalence point(s)" section of the .LVS_ERRORS file and the "ICV_Compare Summary" section in the lvs.log file to ensure complete output. In case of any discrepancies, user shall rerun LVS.
- AoU-ICV-012: User shall check for consistency between the .LAYOUT_ERRORS ASCII violation report and the corresponding LVS "Extraction Errors" violation report in the VUE GUI. In case of any discrepancies, user shall rerun LVS.

- AoU-ICV-013: User shall check for consistency between the .LVS_ERRORS ASCII violation report and the corresponding "LVS Errors" violation report in the VUE GUI. In case of any discrepancies, user shall rerun LVS.
- AoU-ICV-014: User shall review .RESULTS file to determine whether it contains the expected steps, including design hierarchy identification, processing of polygons, and that all Layout/LVS rules are applied and executed.
- AoU-ICV-015: User shall review .RESULTS and .sum files for error, warning, and abort messages. User shall review tree files to ensure entire design hierarchy was processed.
- AoU-ICV-020: User shall review steps in the .RESULTS file, including design hierarchy identification and polygon processing to ensure that all layers are included, no subset windows are selected, and all Layout/LVS rules are executed.
- AoU-ICV-021: User shall compare and check for consistency in LVS output files between the "ERROR SUMMARY" section of the .LAYOUT_ERRORS file and the "WARNING ... violation(s) found" entries in the .sum file to ensure complete output. In case of any discrepancies, user shall rerun LVS.

All analyzed failure modes and prevention, detection and mitigation measures (including conditions and assumptions of use listed above) are independent of the exact IC Validator tool version.

A software tool criteria evaluation performed by Synopsys according to ISO 26262-8, clause 11, which assumes the fulfillment of all conditions of use (CoU) and assumptions of use (AoU) as described above, results in a required tool confidence level:

TCL1 for *IC Validator* Use Case 2: LVS (Layout Versus Schematic)

In this case, no further activities for software tool qualification are required.

Use Case 3: Metal Fill

In this use case, the goal is to create layers of consistent height across the entire area of the chip. This is accomplished through runset coding that generates polygons to be inserted into low-density open areas on each layer. The insertion of such fill shapes is checked in the DRC use case to ensure that the density of polygons within certain windows is within the allowable range. Typically, rules that verify the resulting metal fill within the design are performed separately in the DRC runset, however, it is possible that rules can be coded in the metal fill runset. Furthermore, there are some forms of default checking that are performed (for example, to ensure a design is on the grid), so output files must be reviewed for violations and corrected as necessary.

In this use case, the IC Validator tool uses and generates the following main inputs and outputs.

- Inputs:
    - Layout (GDSII, OASIS, OpenAccess, NDM, Milkyway)
    - Metal fill runset
- Expected outputs:
    - Layout (metal fill polygons)
    - .RESULTS file
    - .sum (summary) file
    - Detailed layout violations (*topcell*.LAYOUT_ERRORS)

For this use case of *IC Validator*, the following conditions of use (constraints for the design and design environment, recommended procedures for the tool usage, etc.) shall be met:

- CoU-ICV-001: User shall review all error and warning messages and take appropriate action.
- CoU-ICV-005: User shall not use selectable rules and selectable windows.
- CoU-ICV-007: User shall only run Metal Fill on final layout (after DRC clean).

For this use case of *IC Validator*, the following assumptions of use (required actions to be taken by the tool user to prevent or detect design errors due to possible tool malfunctions) shall be met:

- AoU-ICV-001: User shall review .RESULTS and .sum files for error, warning, and abort messages. User shall confirm that correct layout and runset are used.
- AoU-ICV-003: User shall review output file timestamps to ensure they are generated for the current run and dated more recently than the run initiated, as listed in the .RESULTS file.
- AoU-ICV-004: User shall perform subsequent DRC and LVS jobs to ensure metal fill generation was created correctly, and that it did not leave low-density regions unfilled due to missing metal fill polygons.
- AoU-ICV-005: User shall perform subsequent DRC and LVS jobs to ensure metal fill generation was created correctly, and that it did not create shorts, spacing violations, or overly-dense regions from additional metal fill polygons.
- AoU-ICV-007: User shall review .RESULTS file to determine whether it contains the expected steps, including design hierarchy identification and processing of polygons.

All analyzed failure modes and prevention, detection and mitigation measures (including conditions and assumptions of use listed above) are independent of the exact IC Validator tool version.

A software tool criteria evaluation performed by Synopsys according to ISO 26262-8, clause 11, which assumes the fulfillment of all conditions of use (CoU) and assumptions of use (AoU) as described above, results in a required tool confidence level:

TCL1 for *IC Validator* Use Case 3: Metal Fill

In this case, no further activities for software tool qualification are required.

7
Limitations of Use Cases

This section describes all known limitations of the use cases mentioned in the previous section.

All known safety-related issues for the IC Validator tool are listed in the IC Validator Safety-Related Issues Master List available on SolvNet.

Appendix A
Software Tool Information

This section provides general information about the IC Validator software tool, which is needed by the tool user for performing his/her software tool criteria evaluation.

The following information about IC Validator is required according to ISO 26262-8, for the planning of the usage of a software tool (clause 11.4.4) and the preparation of the own software tool criteria evaluation (clause 11.4.5).

Please note that some of the information below provided by Synopsys simply needs to be confirmed by the tool user and can be used without modification. Other information must be completed or updated by the tool user to reflect his/her actual situation.

Required Info	**Tool Information**	**Reference / Comment**
Tool vendor	Synopsys, Inc.	ISO 26262-8, 11.4.4.1.a
Tool name and version	IC Validator	ISO 26262-8, 11.4.4.1.a To determine tool version, use: `icv -V`
Tool use cases		ISO 26262-8, 11.4.4.1.c ISO 26262-8, 11.4.5.1.a To be completed by the tool user. Align with / verify against use cases described in Section 6 of this document.
Tool inputs and expected outputs		ISO 26262-8, 11.4.5.1.b To be completed by the tool user. Align with / verify against inputs and outputs described in Section 6 of this document.

Required Info	Tool Information	Reference / Comment
Tool configuration and constraints		ISO 26262-8, 11.4.4.1.b ISO 26262-8, 11.4.5.1.c To be completed by the tool user. Align with / verify against CoU for the use cases described in Section 6 of this document.
Tool environment (OS)	Refer to the IC Validator Installation Notes at https://solvnet.synopsys.com/DownloadCenter. Click the IC Validator tool name, release number, and then "View installation guide" for tool version-specific OS support.	ISO 26262-8, 11.4.4.1.d To be completed by the tool user. Align with / verify against the OS version evaluated by Synopsys. To determine Linux version, use: `uname -osr`
Tool environment (CAD tool chain)		ISO 26262-8, 11.4.4.1.d To be completed by the tool user. To determine name and version of your tool chain, please consult your CAD department.
Maximum ASIL	ASIL D	ISO 26262-8, 11.4.4.1.e
Tool qualification methods	Not applicable	ISO 26262-8, 11.4.4.1.f Software tool qualification is not required for IC Validator
User manual and other usage guide documents	See IC Validator Online Help on SolvNet.	ISO 26262-8, 11.4.4.2.a – d Tool user to include a link to these documents (Synopsys SolvNet or local copy), and to add any additional company-internal tool usage guidelines.

Required Info	Tool Information	Reference / Comment
Known software tool malfunctions, and appropriate work arounds ...	For limitations, refer to Section 7 of this document. https://solvnet.synopsys.com/retrieve/2806205.html	ISO 26262-8, 11.4.4.2.e Tool user to include a link to these documents (Synopsys SolvNet or local copy), and to add any additional company-internal work around descriptions.
Measures for the detection of tool malfunctions ...		ISO 26262-8, 11.4.4.2.f To be completed by the tool user. Align with / verify against AoU for the use cases described in Section 6 of this document.

Appendix B
Complete List of CoU and AoU IDs

The complete list of Conditions of Use (CoU) for IC Validator is in the table below. CoU defines a condition of the design, software tool, design environment, or situation that is assumed and required to be fulfilled by the user.

ID	Description
CoU-ICV-001	User shall review all error and warning messages and take appropriate action.
CoU-ICV-002	User shall follow the IC Validator User Guide or use equivalent scripts.
CoU-ICV-003	User shall perform DRC run on final layout (after metal fill).
CoU-ICV-004	User shall not consider In-design DRC as a final signoff DRC run.
CoU-ICV-005	User shall not use selectable rules and selectable windows.
CoU-ICV-006	User shall do LVS run on final layout (after metal fill).
CoU-ICV-007	User shall only run Metal Fill on final layout (after DRC clean).

The complete list of Assumptions of Use (AoU) for IC Validator is in the table below. AoU defines an action that is assumed and required to be taken by the user of a software tool.

ID	Description
AoU-ICV-001	User shall review .RESULTS and .sum files for error, warning, and abort messages. User shall confirm that correct layout and runset are used.
AoU-ICV-002	User shall review .RESULTS file to determine whether it contains the expected steps, including design hierarchy identification, processing of polygons, and that all DRC rules are applied and executed.
AoU-ICV-003	User shall review output file timestamps to ensure they are generated for the current run and dated more recently than the run initiated, as listed in the .RESULTS file.
AoU-ICV-004	User shall perform subsequent DRC and LVS jobs to ensure metal fill generation was created correctly, and that it did not leave low-density regions unfilled due to missing metal fill polygons.

ID	Description
AoU-ICV-005	User shall perform subsequent DRC and LVS jobs to ensure metal fill generation was created correctly, and that it did not create shorts, spacing violations, or overly-dense regions from additional metal fill polygons.
AoU-ICV-006	The design shall include design-for-test (DFT) functionality and ATPG testing shall be performed on manufactured devices.
AoU-ICV-007	User shall review .RESULTS file to determine whether it contains the expected steps, including design hierarchy identification and processing of polygons.
AoU-ICV-008	User shall review .RESULTS and .sum files for error, warning, and abort messages. User shall confirm that correct layout, schematic, and runset are used.
AoU-ICV-009	User shall compare and check for consistency in their DRC output files between the "ERROR SUMMARY" section of the LAYOUT ERRORS file and the "WARNING ... violation(s) found" entries in the .sum file to ensure complete output. In case of any discrepancies, user shall rerun DRC.
AoU-ICV-010	User shall check for consistency between LAYOUT_ERRORS ASCII violation report and the corresponding "DRC Errors" violation report in the GUI. In case of any discrepancies, user shall rerun DRC.
AoU-ICV-011	User shall compare and check for consistency between the "FAILED equivalence point(s)" section of the LVS_ERRORS file and the "ICV_Compare Summary" section in the lvs.log file to ensure complete output. In case of any discrepancies, user shall rerun LVS.
AoU-ICV-012	User shall check for consistency between LAYOUT_ERRORS ASCII violation report and the corresponding LVS "Extraction Errors" violation report in the VUE GUI. In case of any discrepancies, user shall rerun LVS.
AoU-ICV-013	User shall check for consistency between LVS_ERRORS ASCII violation report and the corresponding "LVS Errors" violation report in the VUE GUI. In case of any discrepancies, user shall rerun LVS.
AoU-ICV-014	User shall review .RESULTS file to determine whether it contains the expected steps, including design hierarchy identification, processing of polygons, and that all Layout/LVS rules are applied and executed.
AoU-ICV-015	User shall review .RESULTS and .sum files for error, warning, and abort messages. User shall review tree files to ensure entire design hierarchy was processed.
AoU-ICV-016	User shall ensure that the foundry has qualified the IC Validator tool and runset.

ID	Description
AoU-ICV-017	User shall perform DRC checking in another tool, such as the IC Compiler or IC Compiler II tool.
AoU-ICV-018	User shall perform Mask Rule Checking (MRC) on individual masks to catch any layout errors that may have escaped.
AoU-ICV-019	User shall review steps in the .RESULTS file, including design hierarchy identification and polygon processing to ensure that all layers are included, no subset windows are selected, and all DRC rules are executed.
AoU-ICV-020	User shall review steps in the .RESULTS file, including design hierarchy identification and polygon processing to ensure that all layers are included, no subset windows are selected, and all Layout/LVS rules are executed.
AoU-ICV-021	User shall compare and check for consistency in their LVS output files between the "ERROR SUMMARY" section of the LAYOUT ERRORS file and the "WARNING ... violation(s) found" entries in the .sum file to ensure complete output. In case of any discrepancies, user shall rerun LVS