A System to Validate and Certify Soft and Hard IP

Bernard Laurent
Synopsys, Inc.
blaurent@synopsys.com

Thierry Karger
STMicroelectronics, Crolles, France
thierry.karger@st.com

Abstract

With the increasing use of Intellectual Property (IP) in the semiconductor industry, the demand to verify IP for quality is high. This paper describes ipscreen, a software tool that aims to both validate and certify IP. Although extendable to all kinds of flows, checks and commercial tools, ipscreen has been primarily designed to comply with the standards that designs produced within STMicroelectronics must adhere to. Both soft and hard IP are targeted.

1. Introduction

The emergence of Intellectual Property in System-On-Chips (SoC) designs requires tools to assess the quality of the IP. The quality assessment takes place in two phases: in the first phase, design teams validate their IP during the design process and this is performed by the developers themselves. They execute software tools that run important checks in order to provide not only a simple archive, but also a structured IP repository. When the IP validation occurs at the end of the design process, it may need a lot of iterations, as the set of input files is both large and structured.

In the second phase, the certification team runs the software tools again, but outside of the design environment. This certification step must verify that the IP complies with the standards of the company, whether it comes from a third-party vendor or not. The certification team then distributes the IP with a quality assessment report, which will be accessed by the designers through the IP catalogue. There may be several flows an IP must comply with within the same company. Two basic flows for STMicroelectronics have been developed in order to verify both hard and soft IP.

ipscreen is an automated software tool, written in Tcl/Tk [1], which targets both IP validation and IP certification. It provides automated corporate frames to allow fast IP quality assessment earlier in the design process. It does not however perform packaging or integration.

ipscreen has been developed jointly by STMicroelectronics and by Synopsys Professional Services during a period of 10 months. About 55,000 lines of Tcl/Tk code was necessary to implement the tool itself, as well as a hundred of either shared or specific scripts. It stands alone, assuming the design or certification team has all the required CAD tools installed.

2. ipscreen features

2.1. Methodology

As discussed in the Reuse Methodology Manual [2], previous waterfall methodologies, in which project transitions take place only when the previous phase is complete, usually lead to a lot of iterations. However, concurrent models, such as the spiral development model, allow better concurrency and flexibility, which lead to faster convergence. The same reasoning holds for IP verification because the certification team waits for the design team to complete the IP validation before beginning to certify it. Moreover, the same set of checks is usually required by both teams.

In order to reduce the IP validation and certification delay, the complete ipscreen product may be downloaded or sent by email in less than 2.5 MB. The distribution contains about 20,000 lines of Tcl/Tk source code, 35,000 lines of a hundred Tcl scripts, as well as the required libraries, executable, and documentation. STMicroelectronics aims to provide its third-party IP suppliers with such a distribution. The benefit is twofold: first, the design team prepares a ready-to-use IP repository and can focus on the significant checks to perform. Second, the certification team receives a complete and stand-alone IP repository and can focus on the new design environment. Both teams may also exchange common reports in order to converge on the final IP repository.
2.2. Flow compliance

Semiconductor companies usually define flows to test their designs. The flows depend not only on the nature of the IP to certify, hard or soft for instance, but also on the company know-how, technology, and the set of CAD tools it currently uses. STMicroelectronics has developed the Blue Book standard as the reference for design implementation within the company. The STMicroelectronics IP-Reuse Group has also defined the frames and the expected quality the IP must reach.

![Figure 1. Structure of an IP repository.](image)

The Blue Book standard [3] describes the set of deliverables recognized by STMicroelectronics. An IP within STMicroelectronics is structured with an index file (vc.bbview file), which defines the set of the IP deliverables. The standard is comprehensive enough to allow some parameterization, for instance according to the available cells and operating conditions. As illustrated in figure 1, the IP deliverables are completely characterized by the vc.bbview file.

The definition of the test frames and the expected results are configurable and programmable by the system manager. We have designed two ipscreen modules, BB_Logical and BB_Physical, to implement two major test flows, used to verify the soft IP and the hard IP respectively, in terms of standards, syntax and consistency:

- the front-end module BB_Logical contains the source code, the test benches, the design conventions, the synthesis libraries, the optional netlists, and scripts to perform synthesis, formal equivalence, verification, and testability estimate. Source code functionality includes the analysis of technology libraries, the design synthesis, the formal equivalence of the source code versus the provided and the generated netlists, and the ATPG flow. Figure 2 below illustrates the front-end module we have designed. Names correspond to the IP deliverables, although some of them may be missing, and are associated with syntactic checks. Dots stand for consistency checks between deliverables. For example, clicking on the dot above Netlist will read in the Verilog netlist and link it with the provided synthesis libraries.

![Figure 2. Front-end BB_Logical module.](image)

- the back-end module BB_Physical contains the gate-level and transistor-level netlists, the RC and timing information, the physical representation of the libraries and the design, and the test patterns to apply. Verilog netlist functionality includes the consistency checks with all the transistor, timing, parasitic, and physical pieces of information. Physical view functionality includes some other timing consistency, as well as some specific transistor-level checks. Figure 3 below illustrates the back-end module.

2.3. Extendable System

ipscreen is an extendable system, whose initial modules target basic STMicroelectronics IP deliverables. Many opportunities arise for the designer to extend the default system. From the biggest modification to the smallest one, a design or certification team can choose one of the following options:

1. the team may develop its own flows by designing another module. The syntax is quite simple and just asks the developer to provide the checks in
terms of names, locations on the GUI, and scripts directories.

2. the team can also add one or more checks in an existing module, and develop its own verification scripts.

3. the team may design another script that replaces the one provided with ipscr een, just by letting the tool know which specific script directory to use.

4. finally, the team may continue to use all the default modules and script directories, but specifying a set of parameters that will make the scripts behave differently.

2.4. Standard checks

Many different ways exist to verify the syntax of a deliverable (i.e., a standard view), or its consistency. In order to exploit the robustness and the consistency of existing parsers and to provide a standard certification report, ipscr een uses exclusively commercial tools (although local transformations have been performed with Tcl/Tk or Perl [4] scripts). Several tools or sets of tools may be used to perform a check. We have had to choose one set of tools to perform the default scripts, based on STMicroelectronics CAD environment. The commercial tools STMicroelectronics has selected by default are:

- the Synopsys tool suite including Physical Compiler(TM) for synthesis, design exploration, netlist and library analysis; PrimeTime(R) for timing related consistency checks; Formality(R) to perform formal equivalence; Chip Architect(R) and

Astro(TM) for physical checks; TetraMAX(R) for ATPG and fault pattern simulations.

- the Cadence Design Systems tool suite including Ncsim(TM) to perform the source code analysis, elaboration and simulation; Verification Cockpit(TM) for design conventions checks; Pearl(TM) for TLF related checks.

- the Mentor Graphics Corporation tool suite including Calibre DRC,LVS(TM) and Eldo(TM) for transistor-level netlists, and simulator Model Technology ModelSim(TM).

For the sake of simplicity, we did not design scripts for other tools. This may however be required when the design or certification team uses different commercial tools than those we have selected.

3. Implementation

3.1. Inputs

The IP validation or certification starts with a binding, which is a compilation of all the input pieces of information together. Whereas reading in each input involves syntactic checks, the binding step performs semantic checks to verify the consistency of the inputs. The required inputs are the following ones:

1. a module, which is a set of connected views to be checked in order to certify a design package.

2. an index file, vc.bbview, which is the STMicroelectronics reference file of a design package, based on the Blue Book standard. It describes the views of an IP repository as well as the associated deliverables, cells and operating conditions.

3. as some checks may require additional pieces of information that are not provided in the bbview index file, the designer is advised to help ipscr een when it is necessary through an information file, vc.info. The syntax of this additional file is close to that of the bbview one.

The vc.info format has been designed for ipscr een only. Verifying a set of deliverables usually requires information that is not included in the IP repository, or that may be hidden. A lot of checks often require some of the following information:

- some libraries, provided in several formats. They are used to link the netlists with the functionality, timing, or physical data of the technology.
the top design name, used to set the current design in the majority of the commercial tools. It is defaulted to the current cell name

- some scripts to replace the default setup or flow we have developed, which may not be enough for some complex designs. They usually refer to the names of ports and subdesigns.

- miscellaneous information such as technological environmental variables, custom rule sets for code verification, constraint files, or private technology files.

When the designer does not provide the previous optional information, ipscreen chooses a default value or assumes that the design stands alone. A good method is to run ipscreen with no information file at all, and find out in the reports what information needs to be specified in order to achieve a complete verification. Moreover, this file allows the designer to override the default values so that the default scripts can be generic enough for all the IP we have certified.

Once bound together, an internal database representation is created, in order to improve retrieval of this information. This database, named Funnel, has been developed by the Library Certification Group in STMicroelectronics Central R&D.

3.2. Syntactic and consistency checks

ipscreen provides both a graphical user interface and a batch mode. The GUI focuses of the user-friendliness of the tool, that is an easy-to-use interface and the possibility of launching checks separately and concurrently, whereas the batch mode allows the certification to be run without any interaction but with automatic report generation when all the checks complete. The certification may be run and the scripts written overnight.

There are both syntactic and consistency checks:

- Syntactic checks invokes reading the deliverables with the associated commercial tool, and searching for issues. Depending on the module under consideration, there may be several deliverables to check, and sometimes several tools to use. The syntactic checks are usually simple to design as the command to read in the deliverable is often straightforward. From figures 2 and 3, we can see that there are 16 and 14 syntactic checks to run for soft and hard IP respectively.

- the consistency checks (or cross checks) however, make sure that deliverables that share dependencies are compatible. There can sometimes be more than two deliverables to check at the same time, and they usually require a larger set of tools to run. These checks are more complex because they always require a specific set of commands to run in order to prove consistency. From figure 2 and 3, we can see that there are 10 and 19 consistency checks to run for soft and hard IP respectively.

It is worthwhile to note that the choice of the tool may lead to different certification status: some tools print a note, while others print warning, and others raise an error! As a consequence, the report needs to be interpreted by the certification team. In some cases, it may decide to force the status of a check, for instance by turning a failure into a warning completion, to enable the execution of the remaining checks.

Any check can be run as a stand-alone activity. Note that some dependencies between the activities exist so that there is no easy way to run them all in a sequence. Dummy activities for syntactic and consistency checks have been created to ease the job of doing this. They do nothing by themselves except wait on the completion of the checks and resolve the inter-check dependencies automatically.

3.3. Outputs

The status of the IP certification relies on the execution of the required set of commercial tools. ipscreen provides two kinds of output files:

- the log files contain not only the messages provided by ipscreen itself, but also the complete output of the tools execution. It allows the designer to investigate in depth the execution of the check.

- the summary file is an automatic comparison between the expected results and the actual ones, in order to identify the success or failure of the check. It is much shorter, and provides important steps in the execution of the checks as well. The system gives the user a diagnosis about the IP issues involved, with respect to the failure condition, in order to help the designer to fix the deliverables.

Both files are important: after looking at the summary report, the designer may analyze details of the check through the log file. He may then decide to change the status according to his own judgment or add some information before re-launching the checks. The raw log report is also useful to bypass the automatic diagnosis.

When at least one check has been executed, two ways are proposed to the designer to visualize the status of the current certification:
With the GUI option, the module layout is automatically updated with appropriate colors, ranking from green (success), red (failure), orange (success with warnings), yellow (in-progress), pink (in the pipe), grey (idle), and white (missing).

With both the GUI and the shell option, an HTML report is generated at the end of the session. It displays in a tabular form the certification status, and points to the scripts, log, and summary files used to perform the verification through hyperlinks. This report stands alone so that it may be sent to the design team. This report may also be linked as part of the IP catalogue documentation.

Second, a layout displaying the current module under certification. Once bound, it is flagged with the colors previously mentioned. The designer can click on any idle button, which corresponds to a syntactic or a consistency check. Each check provides three basic buttons: a setup button to install the check directory, a check button to execute the commercial tool(s), and a res button to extract the summary and perform the diagnosis.

Third, a log window, which prints all the notes, warnings, and errors issued from the designer commands. This log file, as well as the command history, is printed automatically in a separate file, and may be used afterwards.

 ipscreen is provided with a comprehensive on-line documentation that can be invoked from the main window of the GUI. The documentation describes in depth the way to use the tool, as well as the description of the supported EDA tools, the grammar the designer may use, and all the deliverables. The designer can navigate this documentation using the numerous hyperlinks.

5. Qualitative and quantitative results

 ipscreen was used to certify much IP that the STMicroelectronics IP-Reuse Group received from several sources. First, each piece of IP was restructured with an index and an information file (vc.bbview and vc.info file respectively) to comply with the chosen module (BB_Logical and BB_Physical respectively). Second, the IP was certified with the GUI option to assess its quality. Third, the ipscreen history file was reused as a command file for a shell execution that could serve as non-regression testing.

Some qualitative recommendations are presented below, based on the soft IP we have certified recently:

- Structure of IP repository
  - Use separate and stand-alone directories for different cells, operating conditions, and activities; useful for packaging, copy of repository.
  - Make sure the repository is stand-alone in terms of deliverables (technology, libraries, setup files); useful for packaging, consistency checks.
  - Move outputs from previous design validation (log files, reports, netlists); useful for copy of repository, diagnosis.

- Completeness of the IP deliverables

4. System GUI

The GUI displays three main parts: first, a menu bar whose entries allow the user to read, close, edit, and modify the certification inputs (module, index file, and information file), to perform and parameterize the binding, to parameterize the certification, and to perform setups. The setups include checking tool availability, reading parameter and command files, checking time violations, and configuring layout display and colors, etc.

Figure 4. Validation and certification flows.

Figure 4 illustrates the complete flow to validate or certify a IP. Once the module is bound with the bbview and the information file, the checks may be launched via the GUI or via a command file. The designer then generates a certification report and quit.
- Use significant file extension; useful for packaging, syntactic checks.
- Make sure the scripts stand-alone or are configurable at the installation phase, in terms of environmental variables, libraries, and paths to access; useful for consistency checks.
- Use technology independent scripts; useful for consistency checks.
- Use self-checking scripts, avoid hidden output redirection or personal reports; useful for automatic diagnosis.

- Documented IP environment

  - Specify list of the required CAD tools and the version used for validation; useful for checks.
  - Document structure of the IP; useful for packaging.

The improvements are felt by both the design and certification teams. They come mainly from the packaging and the automation. In fact, the effectiveness of a reuse approach depends on an intuitive criterion called the cognitive distance [5], which represents the intellectual effort required to reuse an IP. Abstraction and automation are the key features of a reuse approach because both of them decrease the cognitive distance, by allowing reduced delays and increased reliability. The figures below illustrate the increase in productivity the designers and the STMicroelectronics IP-Reuse Group have reached with ipscreen. Delays are abstracted by symbols + for an easy task, ++ for a medium task, and +++ for a complex task.

The improvements for the validation team are presented below. The first validation remains the same. But the designers do not need much work to package their own IP for ipscreen: a hour is often enough to design the bbview and information files. However, the validation of the improved repository is now partially automated: a change in the IP deliverables leads to a change in the bbview and information files, the history of the previous validation serves as a command file, and the GUI mode allows a fast verification.

<table>
<thead>
<tr>
<th>validation</th>
<th>manual</th>
<th>ipscreen</th>
</tr>
</thead>
<tbody>
<tr>
<td>packaging</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>validated changes</td>
<td>n*++</td>
<td>n*++</td>
</tr>
</tbody>
</table>

Below we give five IP to quantify the increase in productivity. These IP are all pure soft IP for which a direct mapping on a STMicroelectronics standard cell technology is possible. Except golden, these are microcontrollers peripherals without embedded PLL or memory cuts. The certification flow includes the source code assessment (about 150 design rules are checked on the elaborated design), the design exploration and logic synthesis activities, the execution of testbenches, the testability estimate, and the formal equivalence between the source code and the gate-level netlists.

They compare the first certification without ipscreen to the certification with correct packaging. They include the execution of the commercial tools, the diagnosis, and the generation of the certification report. We used a 0.18um library.

<table>
<thead>
<tr>
<th>golden</th>
<th>area (um2)</th>
<th>manual</th>
<th>ipscreen</th>
</tr>
</thead>
<tbody>
<tr>
<td>golden</td>
<td>8207</td>
<td>1 day</td>
<td>5 mins</td>
</tr>
<tr>
<td>axbui2also</td>
<td>34292</td>
<td>2 days</td>
<td>20 mins</td>
</tr>
<tr>
<td>sdtitcalso</td>
<td>69873</td>
<td>2 days</td>
<td>45 mins</td>
</tr>
<tr>
<td>a7licabso</td>
<td>45989</td>
<td>3 days</td>
<td>3 hours</td>
</tr>
<tr>
<td>a7licabso</td>
<td>37613</td>
<td>3 days</td>
<td>5 hours</td>
</tr>
</tbody>
</table>

The improvements for the certification team are presented below. The burden required to explore the IP repository and to re-package the IP is now completely removed. The first certification is more effective: either the designers have provided complete scripts, or ipscreen is able to generate on the fly default consistency checks. The certification team must however still assess the quality of the IP. Finally, the batch mode allows a completely automated non-regression test. The IP catalogue may also be automatically updated with the stand-alone certification report.

<table>
<thead>
<tr>
<th>repository exploration</th>
<th>manual</th>
<th>ipscreen</th>
</tr>
</thead>
<tbody>
<tr>
<td>certification</td>
<td>+++</td>
<td>++</td>
</tr>
<tr>
<td>non-regression tests</td>
<td>n*++</td>
<td>n*++</td>
</tr>
</tbody>
</table>

References