IC Validator WorkBench Application Notes

Version T-2022.03-SP2, September 2022

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

Free and Open-Source Licensing Notices

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.

www.synopsys.com

Contents

About This Document

This preface includes the following sections:

- Related Products and Trademarks
- Conventions
- Customer Support

Related Products and Trademarks

This Application Note refers to the following products:

Synopsys Hercules™

Synopsys IC Validator

Synopsys Proteus™ WorkBench

Synopsys Sentaurus™ Lithography

Synopsys Sentaurus™ Process

Synopsys SolvNetPlus

OASIS®

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
`Courier`	Indicates command syntax.
`Italic`	Indicates a user-defined value, such as *`object_name`*.
Bold	• Within syntax and examples, indicates user input text you type verbatim. • Indicates a graphical user interface (GUI) element that has an action associated with it.
[ ]	Denotes optional parameters, such as: `write_file [-f filename]`

Convention	Description
. . .	Indicates that parameters can be repeated as many times as necessary: *`pin1 pin2 ... pinN`*
|	Indicates a choice among alternatives, such as `low | medium | high`
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the **Edit** menu and choosing **Copy**.
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl key and pressing the C key.

Customer Support

Customer support is available through SolvNetPlus.

Accessing SolvNetPlus

The SolvNetPlus site includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The SolvNetPlus site also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access the SolvNetPlus site, go to the following address:

https://solvnetplus.synopsys.com

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to sign up for an account.

If you need help using the SolvNetPlus site, click REGISTRATION HELP in the top-right menu bar.

Contacting Customer Support

To contact Customer Support, go to https://solvnetplus.synopsys.com.

Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

1

Distributed Processing Tcl Commands

This chapter describes the distributed processing (DP) features available. You can use these tools as masters and workers for DP jobs.

Although you can use DP in the IC Validator WorkBench and the Proteus WorkBench applications, the example in this document uses features found only in Proteus WorkBench.

These topics are available:

Distributed Processing Implementation

The Proteus WorkBench application enables you to use the DP feature in the following areas:

- A Tcl interface, which allows you to configure Proteus WorkBench as a DP master and to set up and schedule DP jobs.
- Proteus WorkBench has been modified so you can use it as a DP worker.

You can use Proteus WorkBench as a DP master or worker.

- Master Implementation–when running Proteus WorkBench as the master, the application can execute Tcl commands to set up and submit jobs.
- Worker Implementation–invoke the DP capabilities using the `-dp` switch in the `dp workerCfg [workers script]` command. When the application is launched by DP with the `-dp` switch, the application takes commands only from DP tasks.

Commands for Running a Distributed Process

The following commands are available in Proteus WorkBench for setting up and running distributed processes:

- dp job delete
- dp job new
- dp job status
- dp job submit
- dp job wait
- dp master delete
- dp master new
- dp task new
- dp task returnValue
- dp task status
- dp task text
- dp workerCFG new

For details about these commands, see the *Proteus WorkBench Command Manual*. Information about these commands is also available in the *IC Validator WorkBench WorkBench Command Manual*.

Example Flow and Scripts

The following sections describe the Tcl command flow, host file, and associated scripts:

- Tcl Command Flow
- Host File
- worker.sh File
- worker.tcl File
- contourTask.tcl File

Tcl Command Flow

The following is an example of a Tcl command flow.

```
module load dp
#
# Create a master.
#
set master [dp master new $runDir/hostfile]
#--> 1
#
# Create a job in the master.
#
set job [dp job new $master]
#--> 1
#
# Create a new worker configuration.
#
set worker [dp workerCfg new $job $workerType
 $minWorkers $maxWorkers $runDir/worker.sh]
#
# Create the tasks.
#
for { set i 0 } {$i < $cntX } { incr i } {
  for { set j 0 ] [ $j < $cntY } { incr j } {
   set task [ dp task new $job $worker "contourTask $cntX $cntY $i $j"]
    }
}
#
# Submit the distributed process job.
#
dp job submit $job
...
# Wait 500 seconds for the job to complete (locking the UI).

...
# Check the status, text, and return value of the task with ID 10.
#
dp task status 10
#--> COMPLETED

dp task text 10
#--> 10

dp task returnValue 10
#--> 0
...

# Finish or cancel the job.
dp job delete $job

# Delete the master.
dp master delete $master
```

Host File

The following is an example of a host file.

```
1|icvwb1|4|/tmp|0|rsh
1|icvwb4|8|/tmp|0|rsh
1|icvwb8|16|/tmp|0|rsh
```

worker.sh File

The following is an example of a worker script in the call to `dp workerCgf new` and the distributed process uses it to launch the workers.

```
/bin/ksh
#
# Transfer your license file to the workers.
#
export LM_LICENSE_FILE=26585@us01_lic4:26585@us01_lic5:26530@us01-lic10
#
# Only needed if you want to turn on the display to see
# what is happening.
#
export DISPLAY=zorro:9
#
# Master hostname to be used in worker.tcl.
#
export DPMASTER=icvwb8
#
# Run from the same directory that the master is in.
#
cd /path/to/logs
icvwb -dp /path/to/worker.tcl -log `hostname`.$$.log >
 `hostname`.$$.stdout_err.log 2>&1
```

worker.tcl File

The `worker.tcl` file was specified in the `worker.sh` file example and it sets up the worker to execute the following tasks:

- Reads `contourTask.tcl` to define the procedure `contourTask`.
- Opens the design.
- Sets up a socket to instruct the master to load the contour when each task is done.

```
#
# This is the file used to set up the worker's context that
# is reused by all the tasks assigned to this worker.
#
# Load the common routines.
#
source ../contourTask.tcl
#
# Load the layout.
#
```

```
layout open ../chip200.gds
cell open chip
#
# Load the model.
#
model load ../model/optdiff.cmdl
#
# Map the layer to the model.
#
model unmap 49:1
model map {49:1 0}
#
# Set up the socket to our master.
#
global sockChan
set server $env(DPMASTER)
set sockChan [socket $server 4444]
```

contourTask.tcl File

The `worker.tcl` file uses the `contourTask.tcl` file and defines the procedure that is executed by each task. It creates, saves, and deletes a contour, instructs the master to load the contour, and then sets the results to the result of a load of the contour.

```
proc contourTask { cntX cntY i j } {
#
# Your channel back to the master.
#
global sockChan
#
# Get the bbox of the cell to determine what part of the
# cell you are contouring.
#
set bbox [cell bbox]

set width [expr [lindex $bbox 2] - [lindex $bbox 0]]
set height [expr [lindex $bbox 3] - [lindex $bbox 1]]

set sectionWidth [expr $width/$cntX]
set sectionHeight [expr $height/$cntY]

set baseX [expr [lindex $bbox 0] + $i * $sectionWidth]
set baseY [expr [lindex $bbox 1] + $j * $sectionHeight]
#
# Show the view.
#
view "$baseX $baseY [expr $baseX + $sectionWidth] [expr $baseY +
 $sectionHeight]"
update
#
# Create the contour.
```

```
#
set contourId [contour make "$baseX $baseY [expr $baseX + $sectionWidth]
 [expr $baseY + $sectionHeight]"]
#
# Show the contour so you can see it on the screen and wait a second.
#
update
after 1000
#
# Save the contour.
#
set contourFileName "$i-$j.ctr"
contour save $contourFileName -list $contourId
#
# Delete the contour since you do not need it any more.
#
contour delete $contourId
#
# Send a command to the master to get it to load the contour file.
#
puts $sockChan "contour load $contourFileName"
flush $sockChan
#
set result "contour load $contourFileName"
```

2

Running Additional Applications

This chapter describes how to invoke applications or modules running in the IC Validator WorkBench application.

These applications and modules are available:

- Invoking the Sentaurus Lithography Integration Tool
- Invoking the Sentaurus Lithography Resist Calibrator Tool

Invoking the Sentaurus Lithography Integration Tool

This section covers the following topics:

- Setting Up the Environment
- Starting the Tool

Setting Up the Environment

The Sentaurus Lithography Integration tool is supported only for Linux operating systems. To have Sentaurus Lithography Integration binaries available throughout the directories, add the correct path to the binaries of the IC Validator WorkBench installation in the Linux search path `PATH` and adjust the environment settings to your needs and infrastructure requirements.

Note:

An additional Sentaurus Lithography installation and license is required.

Starting the Tool

The Sentaurus Lithography Integration tool is a module embedded in the IC Validator WorkBench application. You can start the Sentaurus Lithography Integration tool from the command line.

At the prompt, enter:

```
$ icvwb <ICVWB_HOME>/etc/sli/sliini.mac
```

Invoking the Sentaurus Lithography Resist Calibrator Tool

This section covers the following topics:

- Setting Up the Environment
- Starting the Tool

Setting Up the Environment

The Sentaurus Lithography Resist Calibrator tool is supported only for Linux operating systems. To have Sentaurus Lithography Resist Calibrator binaries available throughout the directories, add the correct path to the binaries of the IC Validator WorkBench installation in the Linux search path `PATH` and adjust the environment settings to your needs and infrastructure requirements.

Note:

An additional Sentaurus Lithography installation is required.

Starting the Tool

You can start the Sentaurus Lithography Resist Calibrator tool from the command line.

At the prompt, enter:

```
$ icvwb <ICVWB_HOME>/etc/src/srcini.mac
```

The Sentaurus Lithography Resist Calibrator dialog box opens in the lower-left corner.

3

User Modes

The application provides specialized user modes, that are designed to better support groups of users with different needs. Additionally, this allows you to switch between modes dynamically without closing/opening the session.

You can choose the user mode in which you want to work from the **User** group on the **Home** tab, or set the `ICVWB_USER` environment variable to one of the following before starting the application:

- ICVWB
- HERCULES
- ICV
- ICV64
- ICV_PLM
- SENTAURUS

Note:

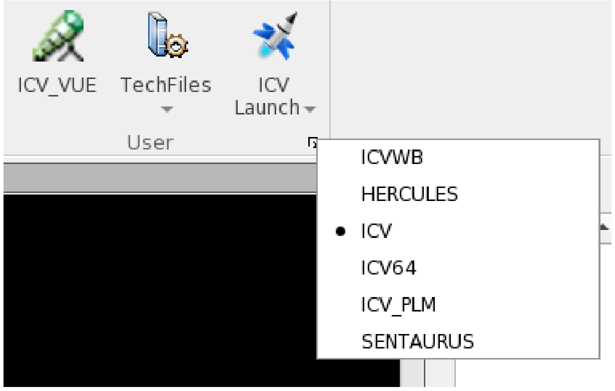
ICV is the default mode of the application when no user mode has been selected or if the `ICVWB_USER` environment variable has not been defined.

User modes are not available in the Windows environment.

ICVWB

To invoke this mode:

1. In the **User** group on the **Home** tab, click the Modes options button.



2. Choose ICVWB from the menu.

HERCULES

To invoke this user mode:

1. In the **User** group on the **Home** tab, click the Modes options button.
2. Choose **HERCULES** from the menu.

This user mode performs these tasks:

- Create a button ( ) that allows you to set up the socket and launch the Hercules program `vue`.
- Creates a user button ( ) in the **ICV** group of the **View** tab to mark vertices. Click this button and then click a location on the layout. All of the vertices of shapes under the cursor will then be marked with a point markup object. The cell hierarchy of the shapes will also be output in the command history in the Command Pane.

ICV

The ICV mode enables you to run IC Validator tools such as ICV Launch and ICV VUE from IC Validator WorkBench and import technology files.

To invoke this user mode:

1. In the **User** group on the **Home** tab, click the Modes options button.
2. Choose **ICV** from the menu.

This user mode enables you to perform these tasks:

- **TechFiles**

  Import the following in IC Validator WorkBench:

  - CC/Virtuoso technology file: import layer names, configurations, and layout connectivity.
  - ICC2 technology file: import layer names, configurations, and layout connectivity.
  - Layer Map: import layer names.
  - ICV Runset: import layer names.

  See

- **ICV_VUE**

  Launch the IC Validator VUE application to review DRC results. See

- **ICV Launch**

  Start **ICV Launch** in the DRC or LVS mode to initialize ICV check runs. See

ICV64

To invoke this user mode:

1. In the **User** group on the **Home** tab, click the Modes options button.
2. Choose **ICV64** from the menu.

This user mode performs these tasks:

- Creates a user button () that will set up the socket and launch the IC Validator application `icv64_vue`.
- Adds a TechFiles button () from where you can import the following in ICVWB:
  - CC/Virtuoso technology file: import layer names and configurations.
  - ICC2 technology file: import layer names, configurations, and layer connectivity.
  - Layer Map: import layer names.

  - ICV Runset: import layer names.

    See .

- Adds the **ICV Launch** button (), from where you can start **ICV Launch** in DRC or LVS mode to initialize ICV check runs. See .
- Creates a user button () in the **ICV** group of the **View** tab to mark vertices. Click this button and then click a location on the layout. All of the vertices of shapes under the cursor will then be marked with a point markup object. The cell hierarchy of the shapes will also be output in the command history in the Command Pane.

ICV_PLM

To invoke this user mode:

1. In the **User** group on the **Home** tab, click the Modes options button.
2. Choose **ICV_PLM** from the menu.

This user mode performs these tasks:

- Creates a user button () that will set up the socket and launch `icv_plm`.

SENTAURUS

To invoke this user mode:

1. In the **User** group on the **Home** tab, click the Modes options button.
2. Choose **SENTAURUS** from the menu.

This user mode performs these tasks:

- Changes the naming convention for highlights to `SIM3D#`.
- Changes the naming convention for gauges to `SIM2D#`.
- Changes the naming convention for points to `SIM1D#`.
- Enables the stretch markup.
- Loads the Sentaurus module. You can save information needed by the Sentaurus Process or the markup information by clicking the **Export** button in the **Sentaurus** group, which is to the right of the **Mode** button in the **User** group on the **Home** tab.
- Sets the layout unit for the GUI to nm.

- Sets `default bop_vertex_max` to 50.
- Allows highlights, stretches, and points to snap to the DBU when you draw them.