Feedback

Proteus™ Reference Manual

Version J-2014.06-10, December 2016

SYNOPSYS®

Feedback

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 http://www.synopsys.com/Company/Pages/Trademarks.aspx.
All other product or company names may be trademarks of their respective owners.

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

Feedback

Contents

Contents

About This Manual

The *Proteus™ Reference Manual* describes the Proteus™ tools commands, including their syntax and use.

Related Products and Trademarks

This manual refers to the following products:

Synopsys Proteus™
Synopsys Proteus™ LRC
Synopsys ProGen™
Synopsys Proteus™ WorkBench
Synopsys Proteus™ MLO
Synopsys SolvNet® support site

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 SolvNet online customer support and through contacting the Synopsys support center.

Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet 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 SolvNet:

1. Go to the SolvNet 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 with SolvNet.)

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

Contacting Synopsys Support

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

- Go to the Synopsys Global Support 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 site and open a case online (Synopsys user name and password required).

1

The Proteus Tool

Describes the main functional components of the Proteus tool and how they work together.

The Proteus OPC Engine

The Proteus OPC engine processes valid GDSII (Stream 6.0) or OASIS files in several major steps, including preprocessing, hierarchy management, context analysis, and correction.

The process is controlled with an ASCII job control file containing:

- global parameters
- references to external script recipes for dissection and correction
- hierarchy management parameters
- miscellaneous job control parameters

The high-level input file used by xmscript has the file extension .xjc, for “X Window job control.” The .xjc file references its necessary components, such as a dissection file, a correction recipe, and one or more model scripts.

The general data flow through the Proteus engine is shown in Figure 1.

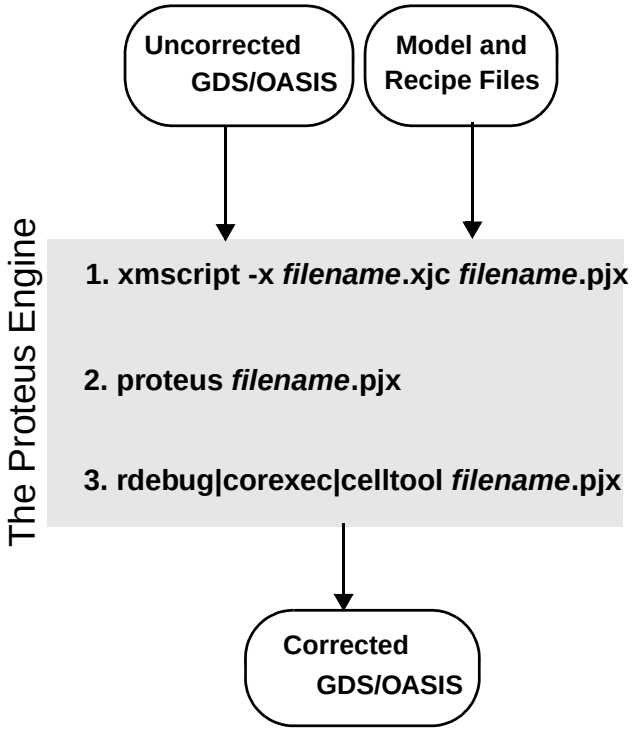


Figure 1 Correction With the Proteus Tool: General Data Flow

1. xmscript: The xmscript utility assembles the separate script file elements from the .xjc file into a single Proteus job executable file (.pjx). See Chapter 2, xmscript Directives and the *Proteus User Guide* for more information about xmscript.
2. proteus: Invoke the engine using the **proteus** executable. the Proteus tool (the tool) takes a job control file (which has been preprocessed by xmscript) and an input pattern file and performs a) hierarchical analysis; and b) an efficient distributed processing of the templates of each template block in succession. The template blocks must be called as part of a .pjx file that contains a `PROTEUS_JOB_FLOW` recipe section. (See Chapter 3, Groups and Data Handling: PROTEUS_JOB_FLOW and the *Proteus User Guide* for details on the components of a Proteus recipe.)

   The Hierarchy Manager executed within the proteus executable takes the job control file and input pattern file (GDSII or OASIS), analyzes and modifies the hierarchy, and outputs correction templates in hierman files. See Chapter 4, Hierarchy Management and the *Proteus User Guide* for more information.

The correction processor executed within the proteus executable runs each template through each template block defined within the `PROTEUS_JOB_FLOW` section. Between template blocks, the processor performs context comparison for each instance of the template, and creates new templates if the context resulting from previous `TEMPLATE_BLOCK` execution is different from instance to instance. See Chapter 5, Correction and the *Proteus User Guide* for more information about correction. For information about distributed processing, see Chapter 6, Distributed Processing.

3. rdebug/corexec/celltool: Additionally, after running the proteus executable and creating hierman output files, you have the option of running stand-alone rdebug (**rdebug**), corexec (**corexec *filename*.pjx -tb *n***), or celltool (**celltool filename.pjx [options] -tb n**). (See Chapter 9, Proteus Applications for details on these utilities.)

2

xmscript Directives

Describes the xmscript preprocessor directives and syntax.

More information about how xmscript works, including an explanation of the Proteus Script Builder window and a complete sample script for creating your own, is provided in the *Proteus User Guide*.

Running xmscript

Important: To use xmscript you must have Motif, free software from http://www.opengroup.org/openmotif, on your system. This is typically a part of your base operating system package, but if you have trouble launching xmscript, check to see that Motif is installed. If it is not, contact your workstation vendor for more information.

The xmscript utility assembles the separate script file elements referenced in the .xjc file into a single Proteus job executable file (.pjx).

The .pjx file can be assembled using one of two methods:

- Invoke xmscript from the command line by entering:

  `xmscript -x jobfilename.xjc jobcontrolfile.pjx`

  See xmscript on page 393 for a list of available command-line options.

- Invoke the interactive xmscript GUI from the command line by entering:

  `xmscript jobfilename.xjc jobcontrolfile.pjx`

  The xmscript GUI interface, the Proteus Script Builder, offers a form in which you can enter job control parameters prior to assembly of the job file.

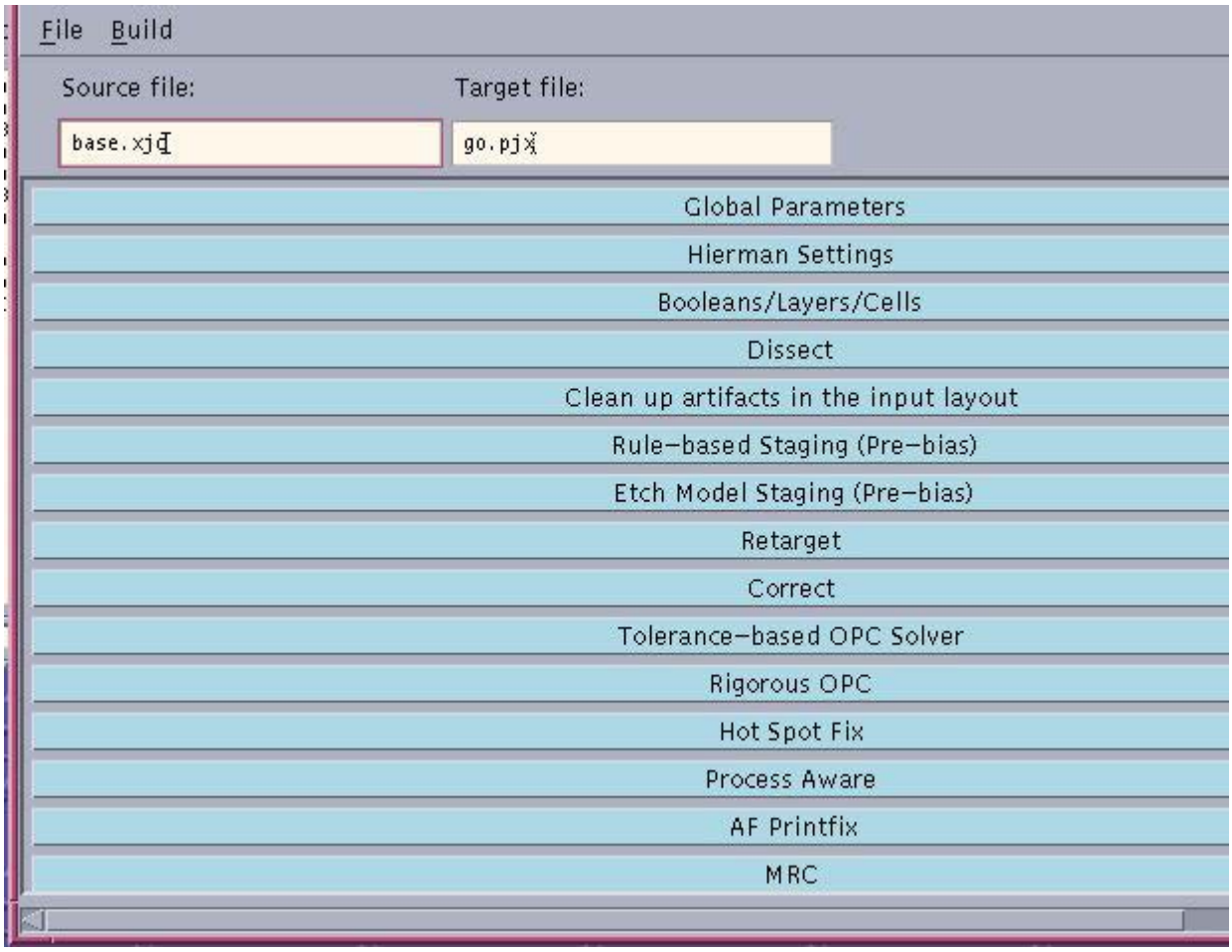


Figure 2 The Proteus Script Builder Window (xmscript GUI)

This form can be configured and provides a convenient way to modify recipe parameters you frequently change. See the tutorial in the *Proteus User Guide* for more information.

The Proteus documentation is also available (as PDF files) from the Proteus Script Builder Help menu. To view and print Proteus documentation, we recommend that you use Adobe Acrobat Reader version 6 or later.

xmscript Preprocessor Directives

This section describes the xmscript preprocessor directives available in the .xjc recipe file and `#INCLUDE`d files, unless noted otherwise. The xmscript directives are shown in the output file prepended with a comment character (`'`).

All xmscript directives must be in upper case.

All xmscript directives beginning with `#XM` are related to the Proteus script builder interface (`XM` refers to the X-Motif window system). These include directives for defining the window layout, text, display parameters, default values, and so forth. These are valid only in the top-level input file that is loaded into xmscript. If they appear in `#INCLUDE` files, these directives generate an error and halt processing, and no output file is produced.

Line Continuations

The backslash character (\) specifies a continuation for a line of commands. This allows you to treat multiple lines of the recipe as a single line. You can use a line continuation in xmscript, corBASIC, or job control directives.

Note: Because the line continuation indicates to “join the next line with this one,” if you put the backslash (\) at the end of a commented line, the next line is also considered commented.

When the Proteus script builder finds a valid xmscript command, it checks for the line-end backslash (\) and continues to read lines until no line-end backslash is found. If it is not a valid xmscript command, the line-end backslash is ignored and passed to the output file.

There is no limit to the amount of whitespace you can use following the backslash and before the carriage return. If anything but whitespace occurs after the backslash but before the carriage return, the line continuation is cancelled.

Example

```
#DEFINE \<cr>
<a>  \ <cr>
123
#DEFINE  \ <cr>
<b>  \ <cr>
poly
#EVAL_DEFINE \<cr>
<cell> \<cr>
<a>_<b>
```

is equivalent to:

```
#DEFINE  <a>  123
#DEFINE  <b>  poly
#EVAL_DEFINE <cell> <a>_<b>
```

In this example, the `<cr>` after each backslash indicates a carriage return, which must be performed after typing the continuation character.

Built-in Substitution Strings

The following describes the built-in text substitution strings, which xmscript uses as if they have been defined with `#DEFINE` directives.

- `<__INPUT_FILE__>` is replaced with the path of the file xmscript is processing when the string is encountered. This can be the job control file specified on the command line or a file that is being processed due to a `#INCLUDE` statement. The substituted text is the full string specified as the argument to the `#INCLUDE` directive.
- `<__LINE_NUM__>` is replaced with the line number (in the current input file) being processed when the string is encountered.
- `<__OUTPUT_FILE__>` is replaced with the name of the file to which xmscript is writing at the time the string is encountered. At the start of processing, this is the file specified in the xmscript command line or in the Target file field in the Proteus script builder window. If a `#OUTPUT_FILE` directive is encountered, the value of this string is changed.

  If the `#OUTPUT_FILE` directive does not contain the `-APPEND` argument, it opens the specific file as a new file, deleting any data already in the file. This is true even if the same file name was previously specified with another `#OUTPUT_FILE` directive. If the `#OUTPUT_FILE` directive contains the `-APPEND` argument, the output is appended to an existing file.

Thus, if the file `demo.xjc` contains the following:

```
#OUTPUT_FILE A
  Writing target file <__OUTPUT_FILE__> from <__INPUT_FILE__>
#OUTPUT_FILE B
  Writing target file <__OUTPUT_FILE__> from <__INPUT_FILE__>
#OUTPUT_FILE A
Writing target file <__OUTPUT_FILE__> again at line
 <__LINE_NUM__> of <__INPUT_FILE__>
```

then running xmscript results in A containing only

```
'#OUTPUT_FILE A
Writing target file A again at line 6 of demo.xjc
```

If the second `#OUTPUT_FILE A` in `demo.xjc` instead said `#OUTPUT_FILE A -APPEND`, running xmscript would result in A containing:

```
'#OUTPUT_FILE A
Writing target file A from demo.xjc
'#OUTPUT_FILE A -APPEND
Writing target file A again at line 6 of demo.xjc
```

#CALC_SYMBOLS

Description

This xmscript directive reassigns the symbols used to delimit xmscript expressions that the application evaluates at the time it builds the .pjx file. For example,

```
#DEFINE <a> 2
generic_variable_name = {4*<a>-1}
```

When xmscript builds the .pjx file, it substitutes <a> and evaluates the expression. In the .pjx you then have:

```
'#DEFINE <a> 2
generic_variable_name = 7
```

You can replace these defaults with symbol strings such as “>>>” and “<<<“. Open and close strings cannot include any of the following characters: + - * / \ % ( ). This directive can occur multiple times in a job script.

Within the .xjc and included files, `#CALC_SYMBOLS` can be set multiple times to different symbols. As the .pjx file is built, xmscript applies `#CALC_SYMBOLS` directives to the subsequent code in the order it encounters them, until a new `#CALC_SYMBOLS` directive is encountered.

By default, the open and close symbols are curly braces “{” and “}”.

Note: When Python is used in the recipe (inside `PYTHON_MODULE`/`END_PYTHON_MODULE` allowing you to define libraries and variables within modules), the `#CALC_SYMBOLS` and the expressions to evaluate cannot use the default curly braces ({ and }); xmscript automatically sets `#CALC_SYMBOLS` to double curly braces ({{ and }}) (the default for the `PYTHON_MODULE`) and resets to single curly braces ({ and }) at `END_PYTHON_MODULE`. Thus, inside a `PYTHON_MODULE`, if you want xmscript to evaluate a variable, modify your syntax so that it contains either double curly braces around that variable, instead of single ones, or parentheses instead of braces. See "#CALC_SYMBOLS with Python Recipes" in the *Python in Proteus User Guide* for further details.

Syntax

`#CALC_SYMBOLS open|close` *`string`*

Options

`open`

Specifies the open delimiter symbol.

`close`

Specifies the close delimiter symbol.

#COMMENT_SUBSTITUTE

Description

This xmscript directive turns comment parsing on and off. When present, comments on lines by themselves are copied into the output file with substituted values. Turning off `#COMMENT_SUBSTITUTE` puts the original text of such comments in the output file.

`#COMMENT_SUBSTITUTE` is in effect for all comments regardless of whether or not other directives occur after `#COMMENT_SUBSTITUTE`.

You can also turn on comment parsing by invoking xmscript with the `-C` option at the command line. For example,

xmscript -C *inputfile outputfile*

Syntax

```
#COMMENT_SUBSTITUTE ON|OFF
```

Options

`ON`

Turns on comment parsing.

`OFF`

Turns off comment parsing. This is the default.

Example

The input file:

```
#DEFINE <a> 0.1
#COMMENT_SUBSTITUTE OFF
CORGRID   <a>
' Here is <a> in a comment
#COMMENT_SUBSTITUTE ON
SCALE_OUT <a>
' Here is <a> in another comment
```

results in the output file:

```
'#DEFINE <a> 0.1
'#COMMENT_SUBSTITUTE OFF
CORGRID   0.1
' Here is <a> in a comment
 '#COMMENT_SUBSTITUTE ON
SCALE_OUT 0.1
' Here is 0.1 in another comment
```

#DEFINE
#ENVDEFINE
#EVAL_DEFINE
#REDEFINE

Description

These xmscript directives define text replacement strings. In all cases, text substitution is global and made in a single pass. The xmscript application replaces all instances of *`symbolic_name`* following the definition with *`substitution_string`*.

During processing, xmscript substitutes *substitution_string* for *symbolic_name* and converts the original #DEFINE directive to a comment.

A #ENVDEFINE directive takes the substitution string from a UNIX shell environment variable named in the second argument. When there are more than one #DEFINE or #ENVDEFINE directives for the same *substitution_string*, the default behavior is the first encountered #DEFINE or #ENVDEFINE directive is applied and subsequent #DEFINE and #ENVDEFINE directives are ignored.

To force a later #DEFINE directive to override a previous one, use a #REDEFINE directive.

To redefine the substitution string that was assigned with #ENVDEFINE, you can use #REDEFINE, but remember that the new value cannot be a UNIX environment variable. For example, you must redefine

```
#ENVDEFINE <test> PROTEUS_APPS_HOME
```

using the full path as the argument string, not simply the environment variable, as shown:

```
#REDEFINE <test> /local/install/proteus_<current_release>/
recipe_lib/
```

Use caution with #DEFINE directives when using xmscript to process a model or recipe, and then process the output file to create a final job control file. This is because xmscript converts any #DEFINE directives in the original file to comments, so the directives no longer have an effect in the second run of xmscript.

For #EVAL_DEFINE directives, xmscript first performs substitution and evaluation to the *substitution_string*, before saving it in the substitution list. This allows you to associate the evaluated *substitution_string* to the *symbolic_name*. The application substitutes input file lines recursively until it cannot make any more text substitutions. If it can make more than one substitution on an input line, the first substitution defined takes precedence. Except for #EVAL_DEFINE, xmscript does not substitute substitution strings themselves until used in a line that is not an xmscript directive.

Note: Using, for example, #XMONOFF <param_a> ON <true> <false> and then using <true> or <false> later in your file (for example in IF conditionals) could cause issues if you have not yet defined <true> and <false>. You must define the proper substitution (for example #DEFINE <true> 1 and #DEFINE <false> 0) prior to using the string that contains <true> or <false> (<param_a> in this case).

Syntax

```
#DEFINE symbolic_name substitution_string
#ENVDEFINE symbolic_name shell_variable
#EVAL_DEFINE symbolic_name substitution_string
#REDEFINE symbolic_name substitution_string
```

Options

symbolic_name

A unique name used to identify one entity, such as a segment, a layer, a cell, an array, a variable, or a path name.

By convention, symbolic names are written within enclosing angle brackets (< >), but these are not required. If used, the brackets are part of the string for substitution. If a symbolic name contains white space characters, enclose the string within double quotation marks (“ ”).

substitution_string

The replacement string for matching *symbolic_name* declarations.

If a substitution string contains white space characters, enclose the string within double quotation marks (“ ”).

shell_variable

The UNIX shell environment variable for matching *symbolic_name* declarations.

Examples

The input file:

```
#DEFINE <weight> 25
result = factor + <weight>
```

results in the output file:

```
'#DEFINE <weight> 25
result = factor + 25
```

In this example,

```
#DEFINE  <a>  123
#DEFINE  <b>  poly
#EVAL_DEFINE <cell> <a>_<b>
Result: <cell> == 123_poly
```

`<cell>` is defined as `123_poly` and is not dependent upon `<a>` or `<b>`.

```
#REDEFINE <b> metal
Result: <cell> == 123_poly
#EVAL_DEFINE <cell> <a>_<b>
Result: <cell> == 123_metal
#REDEFINE <b> diffusion
Result: <cell> == 123_metal
```

Recursive substitution example:

```
#DEFINE a c
#DEFINE ccb taxi
cab
```

In the previous example, the output is “tcxi”. The string “cab” is substituted using the first `#DEFINE`, and becomes “ccb”. The second substitution replaces “ccb” with “taxi”. Finally, since the “a” can still be replaced with “c”, the output string becomes “tcxi”.

Order dependence example:

```
#DEFINE cab taxi
#DEFINE a c
cab
```

In this example, the output is “tcxi” as in the previous recursive substitution example. Although both substitutions are possible, the replacement of “cab” with “taxi” is performed first, then the “a” is replaced with “c”.

```
#DEFINE a c
#DEFINE cab taxi
cab
```

In this example, the output is “ccb”. The first replacement string takes precedence, and once the “a” has been replaced with “c”, no further substitutions can be made.

Delayed substitution example:

```
#DEFINE a 3
#DEFINE b a
b
#REDEFINE a 5
b
```

In this example, you might expect both instances of “b” to be replaced with “3”, but except for `#EVAL_DEFINE`, xmscript delays substitutions until an input line is evaluated. Thus the first “b” is replaced with “a”, which is then replaced with its current definition, “3”. The second “b” is also replaced with “a”, and then the

"a" is replaced with its value at that line, which is "5". The output is thus:

```
'#DEFINE b a
'#DEFINE a 3
3
'#REDEFINE a 5
5
```

In general, if any of the symbolic names might be redefined later in the input files, use `#EVAL_DEFINE` to create substitution strings that contain other xmscript symbolic names.

#ENCRYPT
#END_ENCRYPT

Description

The xmscript application converts text surrounded by `#ENCRYPT` and `#END_ENCRYPT` directives into encrypted sections.

Encrypted sections can

- Contain `#INCLUDE` and `#INCLUDE_PARAMETERS` directives. The file is included and encrypted separately.
- Be nested. All content between the first `#ENCRYPT` directive and the last `#END_ENCRYPT` directive is encrypted.

If an encrypted section contains a `#FOR` directive, it must also contain the corresponding `#NEXT` directive, and vice versa.

Important: The Proteus encryption algorithm protects against casual disclosure of sensitive intellectual property. It does not provide complete protection against malicious attacks. Synopsys accepts no liability for any harm caused by the deciphering of an encrypted section.

While encrypted sections are never decryptable back into clear text, Synopsys tools can read them directly. These Synopsys tools accept encrypted content:

- Celltool (celltool with `-b`, `-bi`, `-bo`, and `-T` options only)
- corexec
- Hierarchy Manager (hierman)
- initgds

- mktop
- proteus
- Proteus LRC
- Proteus WorkBench (pwb)
- puf2vert
- xmscript

Note: Files encrypted with the current release can be used with this release and future releases. Synopsys does not guarantee their use with previous releases.

In order to protect the intellectual property of the author, some Synopsys tools do not run on encrypted sections. These Synopsys tools do not accept encrypted sections:

- Celltool (celltool with any options besides `-b`, `-bi`, `-bo`, and `-T`)
- Proteus AF (afgen)

  Only GUI inputs cannot handle encryption. Recipes generated by the Proteus AF GUI can be encrypted.
- Recipe Debugger (rdebug)

  Accepts a recipe with encrypted sections, but does not debug them; you cannot place a breakpoint in an encrypted section.

Password-Protected Files. Password protection is available for encrypted sections of recipes (using the `#ENCRYPT` and `#END_ENCRYPT` directives). xmscript generates password protection on the encrypted sections of a recipe.

To password-protect a recipe, create a password file and set the `JCL_PASSWD_FILE` environment variable to the path of the password file. (See the *Proteus User Guide* for information on setting environment variables.) The xmscript application uses the first eight characters contained in the file to create an encryption key.

For example:

```
% echo ABCD1234 > pwdfile.txt
% setenv JCL_PASSWD_FILE "`pwd`/pwfile.txt"
```

Applications cannot read password-protected encrypted sections unless the correct key is available through the `JCL_PASSWD_FILE`. A subsequent pass through xmscript does not affect or password-protect the encrypted sections of

a recipe that existed before the recipe was passed through xmscript with the password file. All applications can still read non-password-protected encrypted sections (generated by a previous pass-through xmscript and included using a `#INCLUDE` directive), even though a password-protected file is being used.

You can limit access to the password file using operating system file permissions. This allows the organization to control access to this file in order to prevent unauthorized accessing of the password file. The application generates an error message if you do not have the correct access permissions.

If `JCL_PASSWD_FILE` is defined as an environment variable but the password file it specifies has been changed, removed, or is non-existent, the application generates an error message.

Syntax

```
#ENCRYPT
[encrypted_text]
#END_ENCRYPT
```

Options

None

Example

If the input to xmscript is:

```
p
#ENCRYPT
q
#END_ENCRYPT
r
```

The output of xmscript is:

```
p
ENCRYPTED
encrypted text
END_ENCRYPTED
r
```

#ERROR

Description

This xmscript directive causes xmscript to display an error message with the text *`error_label_text`*, then stops the output file build process. If the desired error text has white space characters, enclose it in double quotation

marks (“”). The error message appears in a pop-up window or, if running xmscript noninteractively, in the terminal window. The build is terminated and you return to the command line or the Script Builder window.

Syntax

`#ERROR` *`error_label_text`*

Options

`error_label_text`

A user-defined error message.

{*expression*}

Description

Expressions are delimited by calculation symbols available in xmscript. The default calculation symbols are the curly braces “{” and “}” outside Python blocks and the double curly braces “{{” and “}}” inside Python blocks. (To redefine the symbols used to delimit xmscript calculations, see #CALC_SYMBOLS.) Expressions are evaluated after any `#DEFINE` substitutions are made. Expressions formed with numbers and the operators in Table 1 are evaluated and the {*expression*} is replaced by the calculated number. You can nest expressions.

All calculation within “{ }” is done in double-precision math. The size and precision of the calculated number is limited only by what can be represented by a double-precision number. Any number over 2^{53} is rounded according to standard double-precision math rounding rules.

Note: The expression within curly braces “{ }” represents a value that should be evaluated as a numerical expression; everything in them must evaluate to a number. In order to avoid unexpected results or errors, do not use curly braces for grouping, but instead use parentheses. Clearly distinguish expressions (curly braces) from grouping (parentheses) in your script.

The operators are listed in Table 1 in their order of precedence (which operation is performed first). The top row represents the highest precedence, performed first. Within a row, multiple operators are grouped left to right.

Table 1 Operators

Operator	Description
`( )`	Parentheses
`+, -`	Unary + and -
`^`	Exponentiation
`*, /, %, \`	Multiplication, Division, Modulus, Snapping
`+, -`	Addition, Subtraction
`<, <=, >, >=`	Comparison testing
`==, !=`	Comparison testing
`&&`	Logical AND
`||`	Logical OR

Example

```
#DEFINE  <grid>  15
#DEFINE  <ambit>  1792
...
OVERRIDE_COR_AMBIT  { (<ambit> + <grid> / 2) \ <grid> }
```

Causes the `OVERRIDE_COR_AMBIT` declaration to appear in the output script as:

```
OVERRIDE_COR_AMBIT 900
```

Note: Because `#IF` expressions evaluate only one comparison, to do a `#IF` on a more complex condition, use expressions to create a single comparison to the `#IF`. For example, the following is not one comparison; do *not* use:

```
#IF <a> == <b> && <c> == <d>
```

Instead, use:

```
#IF { <a> == <b> && <c> == <d> } == 1
```

so that the expression in braces will be evaluated and the single resulting number will be compared to 1.

#FOR
#NEXT

Description

These xmscript directives are used to execute code iteratively. In `FOR - NEXT` loops, the command is parsed and substituted before execution of the loop. As a result, all inputs can be variables.

The lines of code within the loop are executed. This means a substitution is performed, and the result copied to the output for each pass through the loop.

The `STEP` *m* term is optional. The default step value is 1.

Syntax

`#FOR` *i* `=` *j* `TO` *k* `[STEP` *m*`]`

`#NEXT`

Options

i

The variable.

j

The starting value of the loop.

k

The terminal value of the loop.

m

The increment value.

Example 1

If you have the following loop:

```
#FOR index = 0 to 5
index
#NEXT
```

xmscript performs six loops (0 to 5, inclusive). After the `FOR` loop, the index value is 6.

Similarly, if you have the following loop with a negative step:

```
#FOR index = 5 to 0 STEP -1
index
#NEXT
```

xmscript performs six loops (5 to 0, inclusive). After the `FOR` loop, the index value is -1.

Due to floating point effects, you might receive (in rare circumstances) a step value larger than your intended step, and a lower number of overall steps as a result.

```
#FOR index = 0 to 1 STEP .2
index
#NEXT
```

Although the expected index steps should occur at 0, 0.2, 0.4, 0.6, 0.8, and 1.0, depending on machine architecture (specifically, the associated floating point libraries) you might obtain values of 0, 0.2, 0.4, 0.6, and 0.8000000000000001. In this case, xmscript does not complete any more steps because the next value would be greater than the specified terminal value.

Note that, while it is possible in this example to get a value of 0.8000000000000001 for index, it is not possible to get a value of 0.7999999999999999.

In general, do not use floating point values in `#FOR` loops because of this unpredictable behavior.

Make the start, end, and step values integers. Thus, to eliminate the chance of floating point anomalies, the previous example would be better written as:

```
#FOR index = 0 to 5
{index/5}
#NEXT
```

Example 2

The starting value, terminal value, and increment values do not have to be numerical constants. If you have the following variable values:

```
#XMDEFINE <start> 0
#XMDEFINE <end>   5
```

Then the following loop:

```
#FOR index = <start> to <end>
index
#NEXT
```

behaves exactly the same as the loop:

```
#FOR index = 0 to 5
index
#NEXT
```

But there is one important difference: because you defined the variables with `#XMDEFINE`, the end user can modify them (not just the recipe writer). Additionally, when they are modified, the number of iterations and the successive values of the index variable will change.

#GENERATE_COMMENTS

Description

This xmscript directive enables or suppresses xmscript-generated comments after this directive in the job script. User comments in the job script are not affected. By default, xmscript generates comments from the preprocessor commands.

You can also control comment generation by invoking xmscript with the `-c` option at the command line. For example,

xmscript -c *inputfile outputfile*

Syntax

```
#GENERATE_COMMENTS ON|OFF
```

Options

`ON`

Causes xmscript-generated comment generation.

`OFF`

Suppresses xmscript-generated comment generation.

#IF [#ELSEIF|#ELSE] #ENDIF

Description

These xmscript directives are used to execute code depending on whether a condition is met. When xmscript encounters a `#IF` statement, it evaluates the *`T1 CMP T2`* expression (as shown in the following syntax statement).

- If true, it executes the block of true-state statements and then jumps to `#ENDIF`
- If false, it executes the block of false-state statements

`THEN` and `#ELSE` are optional. Twenty levels of nesting are support for `#IF`.

`#ELSEIF` directives work like a C-language `elseif` command. You can have many `#ELSEIF` directives after an `#IF`. Once one `#ELSEIF` is true, no other is executed. As such, this command is order dependent. Exactly one `#ENDIF` is required.

Note: Put *T1* and *T2* within quotation marks if they contain spaces. If they evaluate to numeric values, they must be within the defined `#CALC_SYMBOLS`. (See {expression} on page 18.)

Syntax

```
#IF T1 CMP T2 [THEN]
true-state statements
[#ELSEIF {if statement}|#ELSE {if statement}]
false-state statements
#ENDIF
```

Options

T1

The expression being evaluated.

T2

The expression that *T1* is evaluated against.

CMP

Term of comparison. Where *CMP* is one of these operators:

numeric	string
==	eq
!=	ne
<=	le
>=	ge
<	lt
>	gt

Examples

```
#IF “<recipe_choice>” eq “high-precision correction” THEN
#DEFINE <recipe>  ./demo8i.brcp
#ELSE
#DEFINE <recipe>  ./demo4i.brcp
#ENDIF
```

```
#IF {<min_feature>*2+<grid>} < 200 THEN
#DEFINE <avoid_ring_width> 200
#ELSE
#DEFINE <avoid_ring_width>  {<min_feature>*2+<grid>}
#ENDIF
```

#IFDEF
#IFNDEF

Description

These xmscript directives either include or exclude lines from your file based on whether a substitution exists. `#IFDEF` is true if the substitution value has been defined. `#IFNDEF` is true if no substitution has been defined. The application includes the lines between the `#IFDEF` or `#IFNDEF` directive and its matching `#ENDIF` only if the directive is true.

Syntax

```
#IFDEF substitution_value
#IFNDEF substitution_value
```

Options

substitution_value

The substitution value of interest.

Example

```
#DEFINE a 1
   .
   .
#IFDEF a
   .
   .
' true-state statements (included only if a is defined)
   .
   .
#ENDIF
   .
   .
#IFNDEF b
   .
   .
' false-state statements (included only if b is not defined)
   .
   .
#ENDIF
```

#INCLUDE

Description

This xmscript directive causes xmscript to open *filespec* (including full or relative UNIX path) and to process its contents (that is, the `#DEFINE` directives and math expressions). It copies other text to the output file as though it had occurred in the including file. At the end of *filespec*, xmscript continues processing the previous file past the `#INCLUDE` directive. 256 levels of `#INCLUDE` nesting are supported. If the file cannot be opened, xmscript raises an error unless you use the optional `OPT` keyword.

The xmscript application applies #XM* directives embedded in `#INCLUDE` files as non-GUI directives. For example, if a `#XMDEFINE` is in a `#INCLUDE` file, it is treated as a `#DEFINE`.

The encryption functionality can be used with `#INCLUDE`d files. See #ENCRYPT #END_ENCRYPT on page 15.

Syntax

`#INCLUDE` *`filespec`* `[OPT]`

Options

`filespec`

The filespec to be preprocessed.

#INCLUDE_PARAMETERS

Description

This xmscript directive is used to include a parameter file in a recipe. A parameter file has a simple syntax for initializing variables and for defining array initialization values used in other recipe files. Xmscript supports parameter files to ease the setting of recipe values, particularly arrays.

Within a parameter file, you can define individual symbolic names and array initializers.

The encryption functionality can be used with a parameter file. See #ENCRYPT #END_ENCRYPT on page 15.

Syntax

`#INCLUDE_PARAMETERS` *`pathname`*

Options

`pathname`

The file pathname to a parameter file. The number of characters cannot be greater than 64K.

To define a symbolic name within a parameter file, list the symbolic name and its value. For example,

```
<sensitivity>  0.42319
```

To define an array initializer within a parameter file, begin with an `ARRAY` line, which has the word `ARRAY`, a symbolic name for the array initializer, and the number of rows and number of columns the target array will have. Follow that

with one line defining the initialization values for each row, with the values for each column separated by spaces. For example:

```
ARRAY <newArrInitValues> 4  3
     12     0.1     3
      8     1.2     2.5
      6.2   2.5     2
      4     3       1.4
```

Comments, beginning with an apostrophe, are allowed in a parameter file just as they are in the other recipe files. For a full example of the use of an array initializer, see #UBOUND on page 31.

#OUTPUT_FILE

Description

This xmscript directive redirects subsequent output to a file with the filename specified in the first argument. The entry box at the top of the Proteus Script Builder window for the target file contains the output filename when the `#OUTPUT_FILE` command does not appear anywhere in the input job control files.

`#OUTPUT_FILE` generates additional job control files, but `#OUTPUT_FILE` does not preclude having a target file specified in the Proteus Script Builder window or in the command-line statement `xmscript -x file.xjc A.pjx`.

Syntax

`#OUTPUT_FILE` *`filename`* `[EXEC] [-APPEND]`

Options

`filename`

The file to which output is redirected. If the *filename* is defined as “ ”, the output is displayed in the parent terminal. This filename has an upper limit of 200 characters.

`EXEC`

When the argument `EXEC` appears with this command, the output file is made executable (similar to `chmod +x` in shell scripts).

`-APPEND`

When the argument `-APPEND` is used, xmscript appends the output to an existing file. It is not an error if the file does not yet exist.

Example

The following example shows how additional .pjx files are generated.

```
#SYNTAX_CHECK_OPC OFF

#DEFINE <a> 2

g_v_n = {4*<a>-1}    '<-everything up to this point is in A.pjx

#OUTPUT_FILE B.pjx   '<-everything after this appears in B.pjx
#DEFINE a c
#DEFINE ccb taxi
cab
...
```

Then, when `xmscript -x xjc A.pjx` runs, or the output file builds in the Proteus Script Builder window, xmscript generates the following job control files:

A.pjx

B.pjx

Each .pjx file ends where the following `#OUTPUT_FILE` statement occurs, and the next .pjx begins at that point.

See also

Built-in Substitution Strings on page 8

#REMOVE_FALSE_BLOCKS

Description

This xmscript directive handles comment generation in false branches by removing all lines from them. The logic statements themselves are kept, including the false parts, providing a context for the true branch. The xmscript application starts with `#REMOVE_FALSE_BLOCKS` set to `OFF`.

You can also turn on removal of false blocks by invoking xmscript with the `-r` option at the command line. For example,

xmscript -r *inputfile outputfile*

The xmscript application automatically removes comments and blank lines from false conditions of `#IF` statements.

Syntax

`#REMOVE_FALSE_BLOCKS ON|OFF`

Options

`ON`

Turns on removal of false blocks.

`OFF`

Turns off removal of false blocks. This is the default.

Example

In the following example, if `#REMOVE_FALSE_BLOCKS` set to `ON`, the `#IF` constructs are collapsed to remove the unused code. The `#FOR` loop, because it evaluates to zero iterations, is also removed.

```
#IF <true>  ' first condition

' first comment
first line

#IF <false>  ' second condition

' second comment
second line

#ELSEIF <true>  ' third condition

' third comment
third line

#ELSE

' fourth comment
fourth line

#ENDIF  ' end of inner #IF
```

```
#ELSE

' fifth comment
fifth line

#ENDIF  ' end of outer #IF

#FOR <i> = 1 TO 0
sixth line
#NEXT
```

The previous example is converted by xmscript to:

```
'#IF <true>  ' first condition

' first comment
first line

'  #IF <false>  ' second condition
'  #ELSEIF <true>  ' third condition

' third comment
third line

'  #ELSE
'  #ENDIF  ' end of inner #IF

'#ELSE
'#ENDIF  ' end of outer #IF

'#FOR <i> = 1 TO 0
'#NEXT
```

#SYNTAX_CHECK_OPC

Description

This xmscript directive enables or disables the Proteus syntax checking of output files. You can turn off syntax checking if you are generating a shell script or any other non-Proteus file. Syntax checking is done on a file basis and upon the close of the file if checking is currently enabled.

You can also turn off syntax checking by invoking xmscript with the `-s` option at the command line. For example,

xmscript -x -s *input_file output_file*

Syntax

`#SYNTAX_CHECK_OPC ON|OFF`

Options

`ON`

Turns on syntax checking. This is the default.

`OFF`

Turns off syntax checking.

Example

```
#OUTPUT_FILE a.out
…
#SYNTAX_CHECK_OPC OFF
#OUTPUT_FILE b.out
#SYNTAX_CHECK_OPC ON
…
#OUTPUT_FILE c.out
#SYNTAX_CHECK_OPC ON
…
#SYNTAX_CHECK_OPC OFF
…
```

Here, `a.out` is not syntax-checked because checking is turned off at the time a new output file (`b.out`) is specified. `b.out` is syntax-checked, but `c.out` is not checked because the last status of `#SYNTAX_CHECK_OPC` is `OFF`.

#UBOUND

Description

This xmscript directive sets a symbolic name to either the number of rows or number of columns in an array initializer. Array initializers are created in parameter files. Parameter files are included in a recipe through the #INCLUDE_PARAMETERS directive.

Syntax

`#UBOUND` *`name_to_set name_of_array_initializer`* `1|2`

Options

`name_to_set`

A user-defined symbolic name.

name_of_array_initializer

The array initializer whose number of rows or columns are given to the symbolic name.

1

When the last argument of a #UBOUND directive is 1, *name_to_set* is set to the number of rows in the array initializer.

2

When the last argument is 2, *name_to_set* is set to the number of columns.

Example

```
#INCLUDE_PARAMETERS arrInitParameterFile
   ' Assume that ArrInitParameterFile defined the
   '        array initializer <myArrInitValues>
#UBOUND <myArrRows> <myArrInitValues> 1
#UBOUND <myArrCols> <myArrInitValues> 2

GLOBAL DIM myArr( <myArrRows>, <myArrCols> )
myArr(0, 0) = INITARRAY( <myArrInitValues> )
```

See Also

#INCLUDE_PARAMETERS on page 26

#WARN

Description

This xmscript directive causes xmscript to display a warning message with the text *warning_text*. If the desired warning text has spaces, enclose it in quotation marks (“ ”). Like other warnings, it is displayed in the invoking terminal's window, and, if using the Proteus script builder window, a single window appears indicating that warnings occurred and can be found in the terminal's window. Warnings do not terminate the build.

Syntax

#WARN *warning_text*

Options

warning_text

User-defined warning message.

#XMBAR

Description

This xmscript directive displays a horizontal bar in the Proteus script builder interface. The bar visually divides form entry parameters into groups.

Syntax

#XMBAR

Options

None.

Example

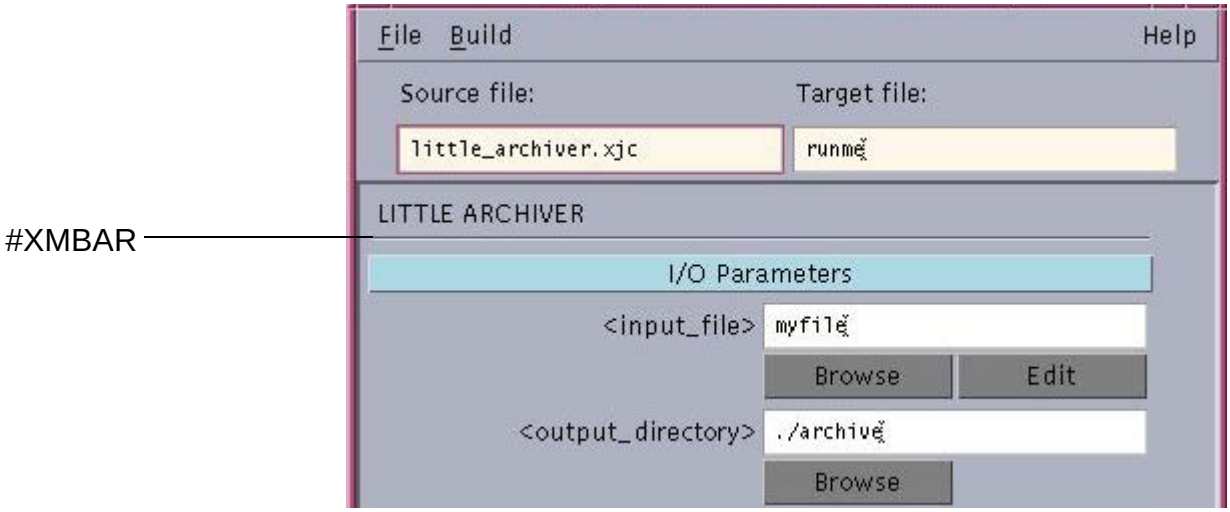


Figure 3 #XMBAR Results in Proteus Script Builder Window

#XMDEFINE

Description

This xmscript directive behaves like the `#DEFINE` directive except *`symbolic_name`* appears in the Proteus Script Builder window as a label with *`default_substitution_string`* displayed in a text entry field next to the label as the default. The *`symbolic_name`* and *`default_substitution_string`* parameters have a similar function in all `#XM` commands.

Syntax

`#XMDEFINE` *`symbolic_name default_substitution_string`*

Options

`symbolic_name`

A unique name used to identify one entity, such as a segment, a layer, a cell, an array, a variable, or a path name.

By convention, symbolic names are written within enclosing angle brackets (< >), but these are not required. If a symbolic name contains white space characters, the string must be enclosed within double quotation marks (“ ”).

`default_substitution_string`

The default replacement string for later uses of *`symbolic_name`*.

If a substitution string contains white space characters, the string must be enclosed within double quotation marks (“ ”).

Example

```
#XMDEFINE <num_copies> 10
```

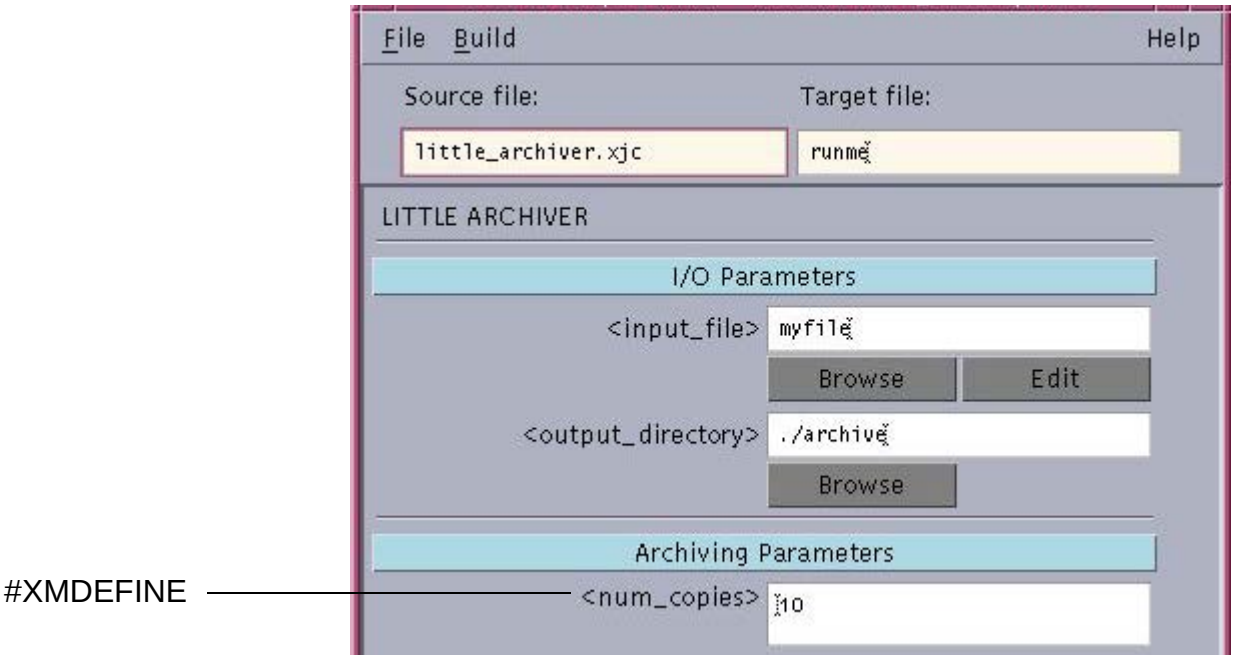


Figure 4 #XMDEFINE Results in Proteus Script Builder Window

#XMDIR

Description

This xmscript directive invokes a text box and **Browse** button. The **Browse** button invokes a file browser.

Syntax

`#XMDIR` *`symbolic_name`* *`default_directory_path`* [*`filter_string`*]

Options

`symbolic_name`

A unique name that can be used to identify one entity, such as a segment, a layer, a cell, an array, a variable, or a path name.

By convention, symbolic names are written within enclosing angle brackets (`< >`), but these are not required. If a symbolic name contains white space characters, the string must be enclosed within double quotation marks (`“ ”`).

`default_directory_path`

The default directory path replacement for the *`symbolic_name`* declaration.

`filter_string`

The filter applied in displaying choices in the graphical interface.

Example

```
#XMDIR <output_directory> ./archive
```

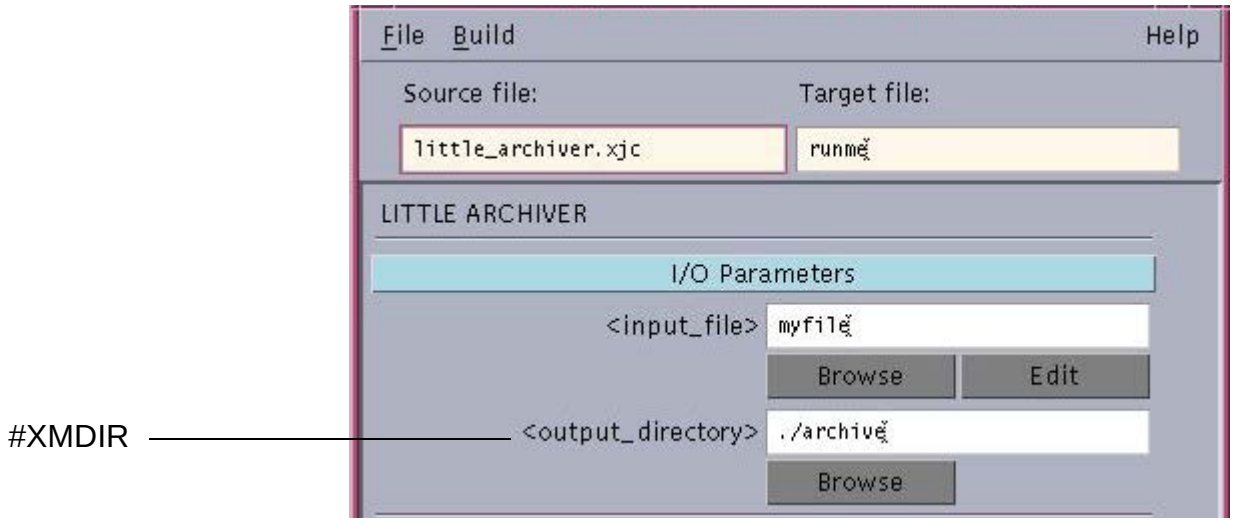


Figure 5 #XMDIR Results in Proteus Script Builder Window

#XMFILE

Description

This xmscript directive is a variant of the `#XMDEFINE` directive and includes, in addition to a text entry field, a **Browse** button to access a file browser, and an **Edit** button so you can to pull the file into a text editor. The *`default_filename`* parameter is the default file selection, and *`filter_string`* is the filter applied in displaying choices.

Syntax

`#XMFILE` *`symbolic_name default_filename`* [*`filter_string`*]

Options

`symbolic_name`

A unique name that can be used to identify one entity, such as a segment, a layer, a cell, an array, a variable, or a path name.

By convention, symbolic names are written within enclosing angle brackets (< >), but these are not required. If a symbolic name contains white space characters, the string must be enclosed within double quotation marks (“ ”).

`default_filename`

The default file selection.

filter_string

The filter applied in displaying choices in the graphical interface.

Example

```
#XMFILE <input_file> myfile
```

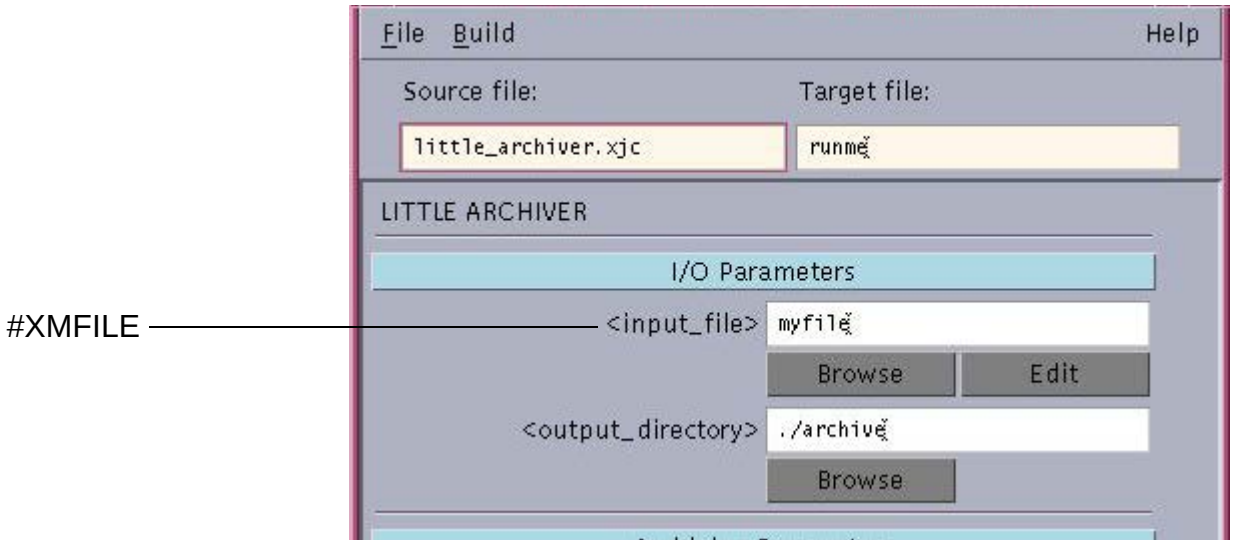


Figure 6 #XMFILE Results in Proteus Script Builder Window

#XMGROUP #XMENDGROUP

Description

This xmscript directive groups a set of #XM directives. A title bar is created for each group with the *label_text* option. When you choose the title bar, the group of interface elements toggles between visible and invisible. The second argument specifies whether the group is initially visible or invisible.

Syntax

```
#XMGROUP label_text [OPEN | CLOSED]
   #XM script statements
   #XMENDGROUP
```

Options

label_text

A quoted string to be displayed as the group's title button.

Example

```
#XMGROUP "I/O Parameters" OPEN
#XMFILE <input_file> myfile
#XMDIR <output_directory> ./archive
#XMBAR
#XMENDGROUP
```

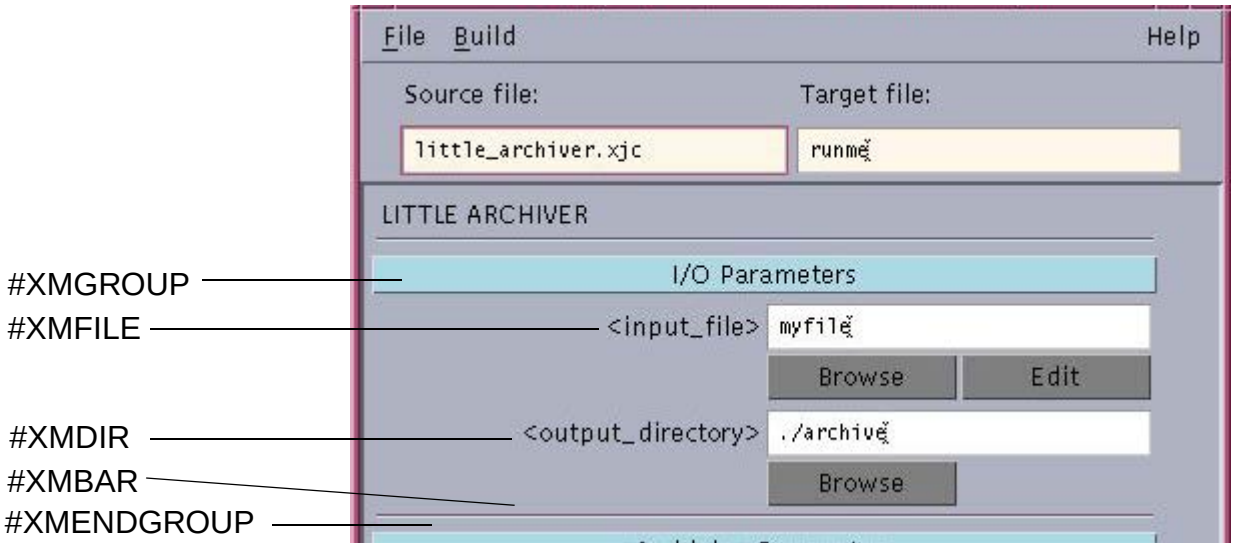


Figure 7 #XMGROUP Results in Proteus Script Builder Window

#XMONOFF

Description

This xmscript directive provides an easy way to select one of two possible values for *symbolic_name* by creating an **ON**|**OFF** toggle switch in the Proteus Script Builder window.

Syntax

```
#XMONOFF symbolic_name ON | OFF
[altsub_on altsub_off [altdispl_on altdispl_off]]
```

Options

symbolic_name

A unique name that can be used to identify one entity, such as a segment, a layer, a cell, an array, a variable, or a path name.

By convention, symbolic names are written within enclosing angle brackets (< >), but these are not required. If a symbolic name contains white space characters, the string must be enclosed within double quotation marks (" ").

`ON`

Enables the toggle switch.

`OFF`

DIsables the toggle switch.

`altsub_on`

Specifies an alternative value (other than 1) to assign to *`symbolic_name`* when the switch is on. Default: 1.

`altsub_off`

Specifies an alternative value (other than 0) to assign to *`symbolic_name`* when the switch is off. Default: 0.

`altdispl_on`

Specifies a label to display when the switch is on. Default: On.

`altdispl_off`

Specifies a label to display when the switch is off. Default: Off.

Note: Quotations are only required around arguments if they contain space characters.

Example

This example does the following:

- Defines the default state of **<underscore_separator**> to be ON.
- Specifies that an underscore character ("_") be assigned to <**underscore_separator**> when the switch is on and that an empty string ("") be assigned when the switch is off.
- Labels the ON button **yes, please** and the OFF button **no, thanks** in the graphical interface.

```
#XMONOFF <underscore_separator> ON "_" "" "yes, please" "no,\
     thanks"
```

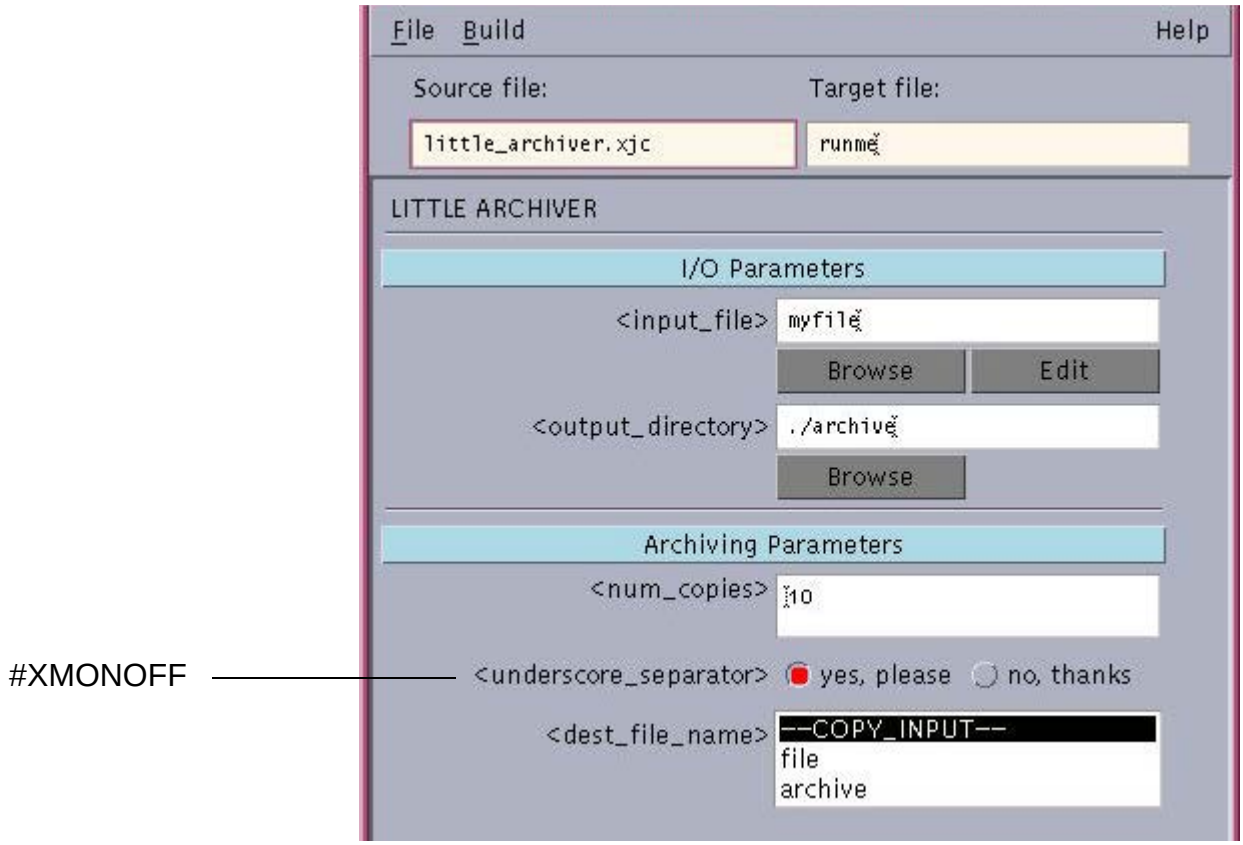


Figure 8 #XMONOFF Results in Proteus Script Builder Window

To apply custom screen labels while using the standard values for on and off (`1` and `0`), you must explicitly define those values as placeholders in the command line; for example,

```
#XMONOFF symbolic_name ON 1 0 "new ON label" "new OFF label"
```

#XMSELECT

Description

This xmscript directive is a variant of the `#XMDEFINE` directive where the choices are presented in a list box. The values in the list can themselves be things that were previously declared with `#DEFINE`.

Syntax

```
#XMSELECT symbolic_name default_choice_n choice_1...
   choice_N
```

Options

symbolic_name

A unique name that can be used to identify one entity, such as a segment, a layer, a cell, an array, a variable, or a path name.

By convention, symbolic names are written within enclosing angle brackets (< >), but these are not required. If a symbolic name contains white space characters, the string must be enclosed within double quotation marks (“ ”).

default_choice_n

The *default_choice_n* argument specifies the index (1 to N) of the default choice (initially highlighted).

choice_1 . . . choice_N

The available choices.

Example

This example results in the Proteus script builder window displaying three choices, **--COPY_INPUT--**, **file**, and **archive**. Number 1, **--COPY_INPUT--**, is the default.

```
#XMSELECT <dest_file_name> 1 --COPY_INPUT-- file archive
```

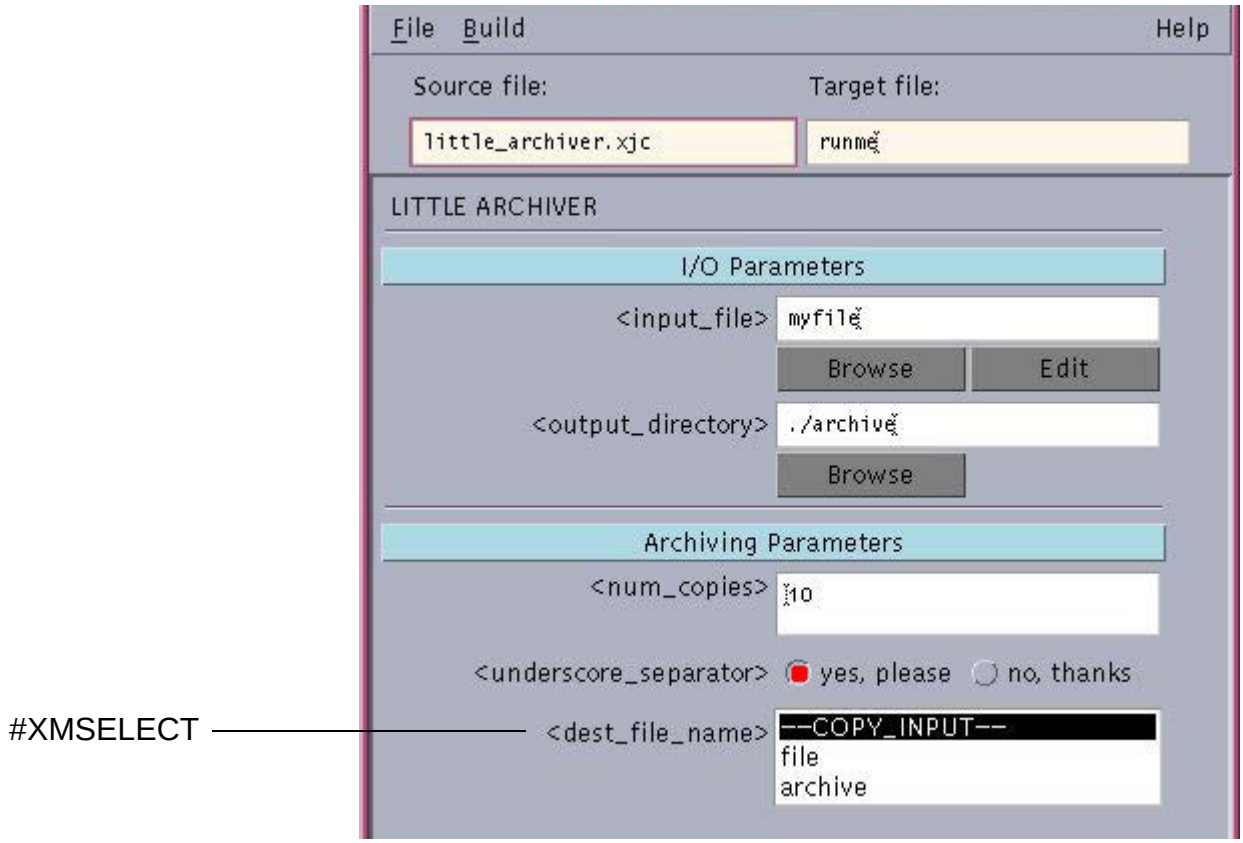


Figure 9 #XMSELECT Results in Proteus Script Builder Window

#XMTEXT

Description

This xmscript directive adds a text label with the text you supply.

Syntax

`#XMTEXT` *`text_string`*

Options

`text_string`

A user-defined text label. The *`text_string`* argument must be enclosed in double quotation marks (“ ”).

Example

```
#XMTEXT "LITTLE ARCHIVER"
```

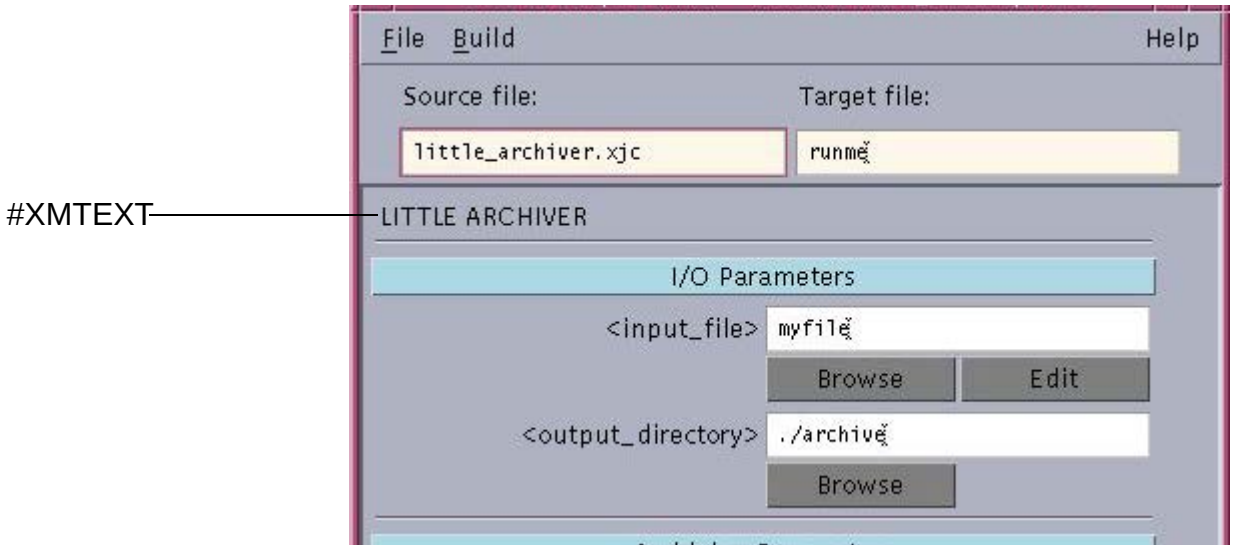


Figure 10 #XMTEXT Results in Proteus Script Builder Window

3

Groups and Data Handling: PROTEUS_JOB_FLOW

Explains the keywords and operations that control data flow and grouping in a PROTEUS_JOB_FLOW recipe, including output file parameters. Provides examples of group organization.

Keywords Controlling Data Flow with Groups

The keywords in this section organize the data flow into groups, which are the foundation for pattern manipulation and flow control from input to output. For more information, refer to the *Proteus User Guide*.

BOSS (Boundary-Oriented Sorted Stages)

BOSS (boundary-oriented sorted stages) allows recipe writers to create better correction at template boundaries. When using BOSS, the recipe takes into account the correction applied to neighboring templates when correcting any given template.

All templates (from a given template call) are sorted into a set of bins, in which no template has boundary effects on any other template in the same bin. The bins are executed serially through a series of automatically duplicated template calls. The number of template calls created is dependent on the design. This process is sometimes referred to as *coloring*.

With BOSS, your recipe can take into account the correction applied to neighboring templates while correcting any given template. You use a specialized kind of `TEMPLATE_CALL` called a `BOSS_CALL`, which reads a color mapping file that associates the instances in a design and their corresponding colors. The color is simply a number that indicates the execution priority of a

template – the lower the color (number) of a template, the sooner it will be corrected. (Color numbering is zero-based.)

The Proteus tool applies the results of the correction of one group of templates ("bin") in the correction of the next group of templates. Figure 11 presents an example with three bins (groups of templates that have no boundary effects on one another). Bin 0 is corrected first, and those results are applied when correcting bin 1. Similarly, the results of bin 0 and bin 1 are applied when correcting bin 2.

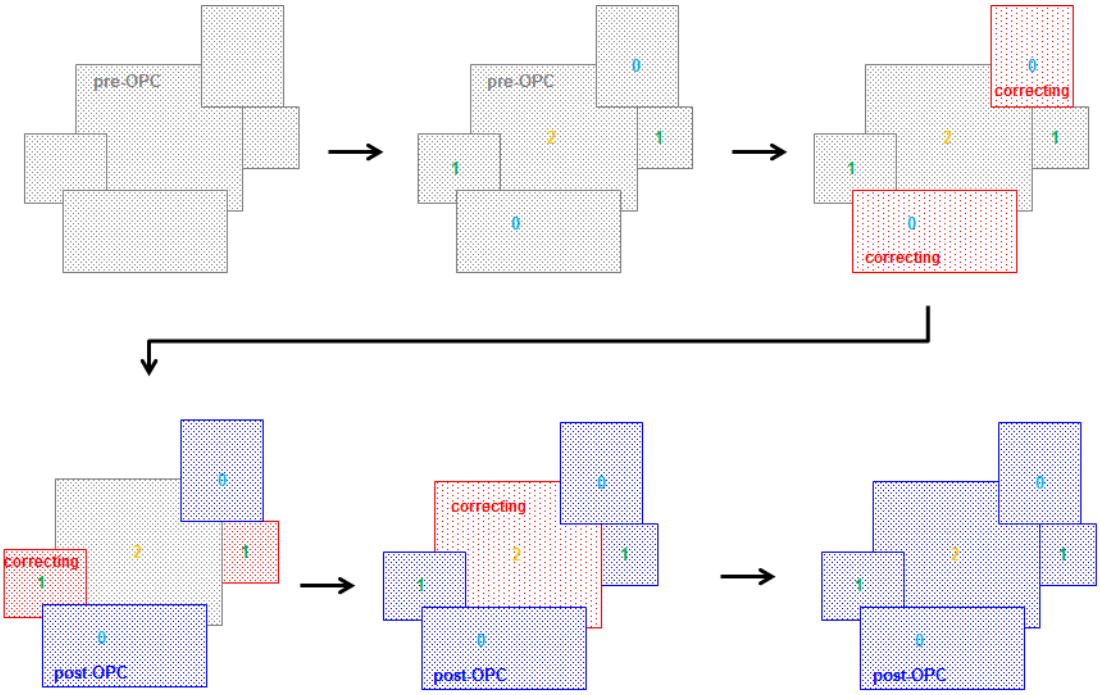


Figure 11 Boundary-ordered sorted stages

Every template executed is aware of its neighbors, and can decide if a neighbor has already produced its output (which is considered "static"). With this knowledge, the template can use blending algorithms where it abuts "static" data and place a preliminary solution at the other boundaries.

The Proteus tool determines the order in which templates are executed, so as to preserve as much compression as possible.

- The tool recognizes Periodic Boundary Correction (PBC) templates and gives them the highest priority during correction. Among PBC instances, XY periodic instances > Y periodic > X periodic > Corner > Non-periodic.
- Smart Block Compression (SBC) instances have the next highest priority.
- If the instances are not PBC or SBC, or if a higher priority instance between them cannot be identified, the instance with the higher instance number gets higher priority.

In most designs, the first bin ("color") contains the majority of the templates. This bin will most likely include all XY PBC templates, as well as the first placement of highly repetitive tiles from Smart Block Compression that do not interact with each other.

The second bin includes templates that have a boundary interaction with the first bin. The third bin includes templates that have a boundary interaction with the first and second bins. This continues until all templates have been placed in a bin. The total number of bins required is determined by the number of boundary interactions.

Figure 12 presents a constant tiled area with templates larger than ambit. This area requires four bins to ensure that no template executes in the same bin as another template within ambit of its boundary.

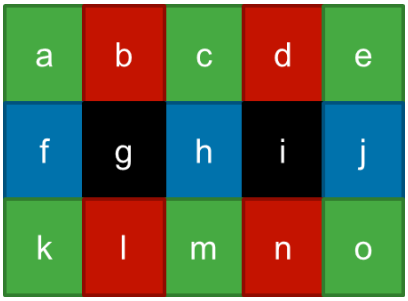


Figure 12 Four-color template example

However, if any row of templates were shorter than ambit in some direction, four colors would be insufficient. See Figure 13.

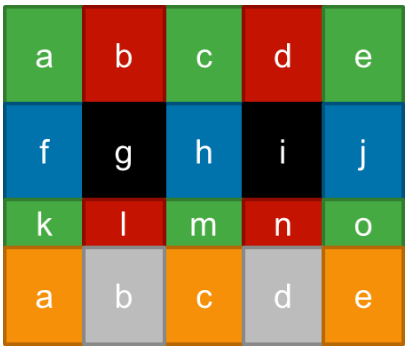


Figure 13 Six-color template example

For cases in which the templates are produced with compression, it is possible to have more colors than the maximum number of interactions. This is because the bins are chosen to maintain the maximum compression, not the least number of bins.

BOSS supports both `CLUSTER FLAT` and `CLUSTER NONE`.

BOSS_CALL

Description

`BOSS_CALL` is a specialized kind of `TEMPLATE_CALL` that uses boundary-ordered sorted stages (BOSS) to expand the current template block into multiple template calls (one for each color). With `BOSS_CALL`, your recipe can consider the correction applied to neighboring templates while correcting any given template.

When using `BOSS_CALL`, note the following restrictions:

- `NEW_PJF_PARSER` must be `ON`.
- `NEW_TEMPLATE_NUMBERS` must be `ON`.
- `GRAPH_SCAFF_OVERLAP` must be `0`.

Syntax

`out_array_name` `= BOSS_CALL(`*`tb_name`*`(`*`in_array_name`*`))`

Options

out_array_name

The user-defined name of an array to hold the layers output from the TEMPLATE_BLOCK. The number of elements placed in the array will match the number of layers output by the TEMPLATE_BLOCK.

tb_name

The user-defined name of the TEMPLATE_BLOCK.

in_array_name

The user-defined name of the input array. The number of elements in this array must match the number of input layers defined for the TEMPLATE_BLOCK *tb_name*.

Note: If you want to pass a layer from one color to the next, ensure that it is declared in both the TEMPLATE_BLOCK and END_TEMPLATE_BLOCK sections. It should be an OPTIONAL LAYER in the TEMPLATE_BLOCK header. For example:

```
TEMPLATE_BLOCK OPC (LAYER cor_layer, ... OPTIONAL LAYER newout)
....
END_TEMPLATE_BLOCK (LAYER newout)
```

Examples

```
OUTLIST = BOSS_CALL(OPC(cor_layer)
```

See also

SUPPRESS_ECOSYSTEM_WARNINGS on page 213

TEMPLATE_CALL on page 83

Python in Proteus User Guide

Global Layer Commands

Global layer commands in Proteus execute from the job flow section. Using these commands, you can identify globally large or long patterns that exceed some constraint bigger than (template size + ambit).

Each of these global layer commands takes a full layer of polygons or edges. Available global layer commands include:

- GLOBAL_AREA
- GLOBAL_LENGTH
- GLOBAL_LENGTHEDGE

GLOBAL_AREA

Description

This command checks the area of all polygons on the specified layer against the specified constraints. It returns an array of layers of polygons that meet the constraints and can be used in a future template call, or as an output.

The number of output layers returned matches the number of constraints, in the same order as the constraint list. For example, for a single constraint, only `output_layers[0]` is valid.

Syntax

```
GLOBAL_AREA (<input_layer>, <constraint> [,
   and=<filter_layer>] [, not=<filter_layer>])
```

Options

`input_layer`

A layer of polygons created from source groups or output from a template call.

`constraint`

You can specify one-sided constraints, two-sided constraints, or a list of both.

- In a *one-sided constraint*, the following operators are valid:

  ```
  <      <=      >
  >=     ==      !=
  ```

  You can use either of these formats to specify a one-sided constraint:

  - `constraint <operator> <number>`
  - `<number> <operator> constraint`

- *Two-sided constraints* can include only the < or <= operators, in the following format:

- *<number>* *<operator>* `constraint` *<operator>* *<number>*

- When specifying *a list of constraints*, enclose the list in square brackets [ ], and use a comma to separate individual constraints. The list can include both one-sided and two-sided constraints.

`and`

Accepts a *`filter_layer`*. Only shapes under the specified layer(s) will be considered. Specifically, the input layer will be trimmed using `mlo.boolean_and(input_layer, filter_layer)`. You can specify a list of `and=`*`filter_layer`* options, surrounded by square brackets and separated by commas.

`not`

Accepts a *`filter_layer`*. Shapes under the specified layer(s) will be ignored. The input layer will be trimmed using `mlo.boolean_not(input_layer, filter_layer)`. You can specify a list of `not=`*`filter_layer`* options, surrounded by square brackets and separated by commas.

Examples

```
large_layers = GLOBAL_AREA( source_m1, [10000 <= constraint <=
50000, constraint > 100000], and=source_and_layer)

a = GLOBAL_AREA(layer1, constraint >= 965000000, and=layer2,
not=layer3 )

b = GLOBAL_AREA(layer1, constraint >= 965000000, [ and=layer2,
and=layer3 ], [not=layer4, not=layer5] )
```

GLOBAL_LENGTH

Description

This command checks the length of all edges on the specified layer against the specified length constraints. It returns an array of layers of polygons whose edges meet the constraints and can be used in a future template call, or as an output.

The number of output layers returned matches the number of constraints, in the same order as the constraint list. For example, for a single constraint, only `output_layers[0]` is valid.

Syntax

`GLOBAL_LENGTH (<`*`input_layer`*`>, constraint= [, and=, not= ])`

Options

input_layer

A layer of edges created from source groups or output from a template call.

constraint

One or more constraints, as specified by the following operators:

```
<       <=      >
>=      ==      !=
```

You can specify a single constraint, a range, or multiple constraints in a list.

and

Accepts an AND layer. Only shapes under the specified layer(s) will be considered. Specifically, the input layer will be trimmed using `mlo.boolean_and(input_layer, and_layer)`.

not

Accepts a NOT layer. Shapes under the specified layer(s) will be ignored. The input layer will be trimmed using `mlo.boolean_not(input_layer, not_layer)`.

Examples

```
long_layers = GLOBAL_LENGTH( source_m1, constraint > 50000,
and=source_and_layer)
```

GLOBAL_LENGTHEDGE

Description

This command checks the length of all edges on the specified layer against the specified length constraints. It returns an array of layers of edges that meet the constraints and can be used in a future template call, or as an output.

The number of output layers returned matches the number of constraints, in the same order as the constraint list. For example, for a single constraint, only `output_layers[0]` is valid.

Syntax

```
GLOBAL_LENGTHEDGE (<input_layer>, <constraint> [,
   and=<filter_layer>] [, not=<filter_layer>])
```

Options

`input_layer`

A layer of polygons created from source groups or output from a template call.

`constraint`

You can specify one-sided constraints, two-sided constraints, or a list of both.

- In a *one-sided constraint*, the following operators are valid:

  ```
  <       <=      >
  >=      ==      !=
  ```

  You can use either of these formats to specify a one-sided constraint:

  - `constraint <operator> <number>`
  - `<number> <operator> constraint`

- *Two-sided constraints* can include only the < or <= operators, in the following format:
  - `<number> <operator> constraint <operator> <number>`

- When specifying *a list of constraints*, enclose the list in square brackets [ ], and use a comma to separate individual constraints. The list can include both one-sided and two-sided constraints.

`and`

Accepts a *`filter_layer`*. Only shapes under the specified layer(s) will be considered. Specifically, the input layer will be trimmed using `mlo.boolean_and(input_layer, filter_layer)`. You can specify a list of `and=`*`filter_layer`* options, surrounded by square brackets and separated by commas.

`not`

Accepts a *`filter_layer`*. Shapes under the specified layer(s) will be ignored. The input layer will be trimmed using `mlo.boolean_not(input_layer, filter_layer)`. You can specify a list of `not=`*`filter_layer`* options, surrounded by square brackets and separated by commas.

Examples

```
long_layers = GLOBAL_LENGTHEDGE( source_m1, constraint > 50000,
and=source_and_layer)
```

HIERARCHY_SKELETON

Description

Specifies layers in the INPUT file to use during smart block creation. Including this keyword in an `INPUT ... END_INPUT` of the Proteus Job Flow causes hierman to use the specified files and layers to form the smart block compression (SBC) tiles for the current run.

For the input file, you can specify either the output of an earlier run, which used `HIERARCHY_SKELETON_OUTPUT`, or another pattern file that has a more well-formed hierarchy on which to base SBC decisions.

Note: This feature is valid only when the following restrictions are met:

- `CLUSTER FLAT` is used
- `NEW_DESIGN_READER_HIERMAN_SCAN` is `ON`
- Only one INPUT section specifies `HIERARCHY_SKELETON`
- All other INPUT section parameters are valid

Syntax

```
HIERARCHY_SKELETON [layer1[:datatype1]
   [layer2[:datatypeN] ... ]]
```

Options

[*layer1*[:*datatype1*][*layer2*[:*datatypeN*] ...]]

The layers in the input file to use for smart block compression analysis. The layers you specify will not be available to the recipe as main or context. You also can define group names or layer specifications from the same file. These named groups will be available to the recipe and will be treated like any other input file or group. If you do not specify any layers, the tool uses all layers to extract hierarchy hint information.

See also

HIERARCHY_SKELETON_OUTPUT on page 159

INPUT and END_INPUT

Description

The `INPUT` statement is used to select pattern data based on layer and datatype and assign a user-defined name to it. Multiple input flows can be

declared within the `INPUT` statement and each input flow can contain data from one or more layers and datatypes.

Layers are defined by mapping a GDS or OASIS layer and (optionally) datatype to a Proteus name. The layer will be extracted from the input file.

It is possible to have up to eight input files by using separate `INPUT` declarations. The input files do not need to be of the same graphics format; it is possible to have one GDS file and one OASIS file, for example. When using multiple input files, note the following restrictions:

- If `TOPCELL_IN` is not specified for a file, the file will be read in and take any topcells it has.
- If the names of any cells in the input files are the same, `INPUT_CELL_PREFIX` must be used to avoid a "name re-use" error.
- Switching the order of multiple input files can produce a different template count.
- Layers that are operated on by recipes are defined by the `INPUT` sections only, not by which layers are present in the input file. In other words, in the following example,

  ```
  INPUT a
    alpha = 1
  END_INPUT

  INPUT b
    beta = 1
  END_INPUT
  ```

  ...an operation on layer alpha operates only on layer_1 of a.gds and an operation on layer beta operates only on layer_1 of b.gds.

The input files are read before the first `TEMPLATE_CALL`, during front-end hierarchical processing. The files are merged into one design, but typically you would load one file into one set of layers and the next file into another set.

Because the input files are read before the first `TEMPLATE_CALL`, you cannot have, for example, one `TEMPLATE_CALL` create the OPC output and another `TEMPLATE_CALL` read it in and merge it with the original. A subsequent run of proteus is required in such a case.

Syntax

```
INPUT [input_format] [input_file]
   input_name = layer1[:datatype1] [layer2[:datatypeN] ...
   layerN[:datatypeN]]
```

```
   [file_specific_job_control_parameter]
   [text = layer:[datatype]   TEXT]
END_INPUT
```

Options

input_format

Optional. Can be GDSII or OASIS. If INPUT_FORMAT is omitted, it is automatically determined from the input file. If INPUT_FORMAT has been specified previously, this argument is unnecessary.

When there are multiple input files, the output format is determined as follows:

- If either input format is OASIS and OUTPUT_FORMAT has not been specified in the job control file, the output is OASIS.
- If all input formats are GDS and OUTPUT_FORMAT has not been specified, the output is GDS.

input_file

Optional. The user-defined name of the input file. If INPUT_FILE has been specified previously, this argument is unnecessary. For a multiple-input recipe, define *input_file* for each INPUT section.

The input files can also be zip files that decompress into GDSII or OASIS:

- If the defined *input_file* has a .z, .gz, .zip, or .gzip extension, proteus verifies that the decompressed version of the file is in the base path (defined by the BASEPATH keyword). If it is not present, the zip file is decompressed into the base path. After a successful run, proteus deletes the decompressed file, unless NO_FRAG_CLEAN is set.
- If the defined *input_file* does not have a .z, .gz, .zip, or .gzip extension and the file as named does not exist, proteus searches for a compressed version, using the defined input file name but with a .z, .gz, .zip, or .gzip extension. If found, a message indicates that the zip file is being used.

Note: The tool checks the input format of the file or files. If the actual format is different from the format specified in the recipe, the tool uses the actual format and issues a warning message.

input_name

The user-defined name of the data flow, also known as a *group*. There can be a maximum of 4095 groups.

layer

An integer value or “UNUSED.” (UNUSED is the same as not listing a layer.) The layer and datatype for OASIS files can be an integer value from 0 to 2^{31} - 2, and for GDSII files from 0 to 32767.

datatype

Optional integer value. The layer and datatype for OASIS files can be an integer value from 0 to 2^{31} - 2, and for GDSII files from 0 to 32767.

The default is all datatypes for the specified layer. If the input file has, for example, layers:datatypes1:12, 1:2, and1:3, and your recipe has "1" without the datatype, all those layers in the input file get merged.

Note: The asterisk (*) is not valid syntax with `PROTEUS_JOB_FLOW` recipes.

file_specific_job_control_parameter

When there are multiple input files, you can declare certain job control parameters within the `INPUT` statement to override the parameter in the job control file. These parameters include `DBU_IN`, `INPUT_CELL_PREFIX`, `SCALE_IN`, `TOPCELL_IN`, and `ROTATE_IN`.

text

The user-specified text-record name.

`TEXT`

An optional qualifier allowing you to copy text records (GDSII or OASIS) from a user-specified layer in the input layout to user-specified layers on the corrected output layout. (*Text records* refers to text strings, not text created as polygon layout.) You can view this text in a layout editor after correction and pass the information, if necessary, to another design automation tool.

The output file stores text records in separate cells from corrected graphics which exist in a hierarchy mimicking the cells from the input layout that contained them. There will be a hierarchy of cells containing text records for each input topcell. The cells that store text are prefixed with "t_". The topcell of each text hierarchy is attached to the output file's topcell.

The `TEXT` output is not affected by the `BASE_GROUP` or `ENVIRONMENT_GROUP` statements.

The `TEXT` qualifier applies only to text in the input layout. If you want polygon layout data, you must include that layer in another statement in the `INPUT` section. If you want both text and polygon layout data from a particular layer,

you must explicitly identify that layer both with and without a `TEXT` qualifier. Text layers cannot be used within template blocks; they can only be referenced in the `OUTPUT` section at the end of your `PROTEUS_JOB_FLOW` recipe.

Note: When the format of the input layout is GDSII, but the format of the output layout is OASIS, the ANGLE and MIRROR information in the original TEXT records will be lost in the output layout.

Examples

Example 1: One Input File

```
PROTEUS_JOB_FLOW

INPUT OASIS original.oas
  orig_layer = 1:0
END_INPUT
```

Example 2: Two Input Files

With two input files, you can add specification lines between the `INPUT` and `END_INPUT` lines to define values that in a one-input-file recipe are either global or nonexistent. For example, in a multiple-input-file recipe, `TOPCELL_IN` and `INPUT_CELL_PREFIX` can be file-specific.

```
PROTEUS_JOB_FLOW

INPUT OASIS original.gds
  orig_layer = 1:0
  DBU_IN  1.0
  SCALE_IN  3.0
  ROTATE_IN 90
  INPUT_CELL_PREFIX orig_
  TOPCELL_IN orig_orig1 orig_orig2
END_INPUT

INPUT GDSII opc_output.gds
  opc_d_layer = 2:0
  DBU_IN  1.0
  SCALE_IN  3.0
  ROTATE_IN 270
  INPUT_CELL_PREFIX output_
  TOPCELL_IN output_OUTPUT1
END_INPUT

END_PROTEUS_JOB_FLOW
```

Example 3: Two Input Files and Input Array Declaration

```
PROTEUS_JOB_FLOW

INPUT GDSII pre_OPC.gds
  in_layer_1 = 2
END_INPUT

INPUT GDSII post_OPC.gds
  in_layer_2 = 0
END_INPUT

IN_LAYER_LIST = [in_layer_1, in_layer_2] 'This line defines an
                                         'in_array you can use
                                         'to refer to all INPUT
                                         'layers with one name.

OUTLIST = TEMPLATE_CALL(TB1(IN_LAYER_LIST) 'In this line you pass
                                           'the input array to the
                                           'template block's
                                           'layers via a
                                           'TEMPLATE_CALL

OUTPUT GDSII out.gds
  2 = OUTLIST[pre_OPC_layer_out]
  0 = OUTLIST[post_OPC_layer_out]
END_OUTPUT
END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK TB1(LAYER pre_OPC_layer, LAYER post_OPC_layer)
pre_OPC_layer_out = pre_OPC_layer.main
post_OPC_layer_out = post_OPC_layer.main
END_TEMPLATE_BLOCK(LAYER pre_OPC_layer_out,
                   LAYER post_OPC_layer_out)
```

Example 4: TEXT qualifier

To include text records in the proteus output file of a `PROTEUS_JOB_FLOW` recipe, create a layer in the input file's `INPUT` section and assign it the layer numbers of the relevant text records. Add the qualifier `TEXT` to the end of the layer assignment, as shown in the following example:

```
PROTEUS_JOB_FLOW

  INPUT OASIS layout.oas
      graphics_layer = 0.0
      text_layer_one = 0:0 TEXT
      text_layer_two = 1:0 TEXT
      graphics_layer_two = 2:0
  END_INPUT

    ' one or more TEMPLATE_BLOCK calls here …
```

Note that a named layer can be assigned either text or graphics, but never both. In this case, text_layer_one gets the text records from input layer 0 while graphics_layer gets the graphics from layer 0. The text layers text_layer_one and text_layer_two can now be put in the output file by assigning them output layer numbers in the same way you would output a graphics layer:

```
  OUTPUT GDSII corrected_output.gds
      7:0 = corrected_layer_one
      8:0 = corrected_layer_two
      9:0 = graphics_layer
      9:0 = text_layer_one
     10:0 = text_layer_one
     11:0 = text_layer_two
  END_OUTPUT

END_PROTEUS_JOB_FLOW
```

As shown above, text and graphics layers can be put on the same layer number in the output. Also, a text layer can be output multiple times to different layer numbers.

Example 5: Defining the hierarchy skeleton

You can define layers to use for smart block compression (SBC) analysis by including the `HIERARCHY_SKELETON` keyword between the `INPUT` and `END_INPUT` lines.

```
PROTEUS_JOB_FLOW

INPUT hint.oas
  ROTATE_IN 90
  HIERARCHY_SKELETON 0:4
END_INPUT

INPUT dpt_out.oas
  cor_layer = 6
END_INPUT

END_PROTEUS_JOB_FLOW
```

INPUT_CELL_PREFIX

Description

Use this optional keyword in the `INPUT` statement to prepend *`prefix_string`* to all cell names in the input file. This prevents a cellname conflict error if there are cells in multiple input files that have identical cellnames. Commands that reference cells by name must specify those cellnames using any prefixing that is in effect. In particular, `FORCE_TEMPLATE` and `MARK` by name must specify their cellnames using the cellname with *`prefix_string`*.

Syntax

`INPUT_CELL_PREFIX` *`prefix_string`*

Options

`prefix_string`

The string to prepend to all cells in the input file.

Example

```
INPUT a.gds
  INPUT_CELL_PREFIX aaa_
  TOPCELL_IN TOP
END_INPUT
```

See also

INPUT and END_INPUT on page 54

LAYER and OPTIONAL

Description

The `LAYER` command is used within a template block to define input and output layers for that template block. You can specify an edge for the input if it was created by a previous `TEMPLATE_CALL` of the recipe.

In a recipe, if a named layer is specified in a `corbasic` call and this layer is not explicitly declared using a `LAYER` command, proteus will fail if you try to use this named layer in the `corbasic` function definition. Ensure that any named layer that is specified in a `corbasic` call (and subsequently used in the `corbasic` function) is explicitly declared through a `LAYER` statement.

When defining inputs to a template block, a keyword prefix of `OPTIONAL` allows a layer or edge to be unspecified within the array of inputs to `TEMPLATE_CALL`. `OPTIONAL` layers or edges not specified are considered `UNUSED` in the `TEMPLATE_CALL`. (You can use the Python method`layer.isUsed()` to test for `UNUSED` layers.)

Without the `OPTIONAL` keyword prefix, a layer or edge defined by `LAYER` must be specified within the array of inputs to `TEMPLATE_CALL` or a parse-time error will be generated.

The `OPTIONAL` keyword is legal only with input layers (within the `TEMPLATE_BLOCK`). An error is generated if `OPTIONAL` is used in the output layer definition section after `END_TEMPLATE_BLOCK`.

Syntax

See TEMPLATE_BLOCK and END_TEMPLATE_BLOCK on page 81

NEW_EARLY_OUTPUT_FROM_PJF

Description

Using the `NEW_EARLY_OUTPUT_FROM_PJF` keyword, you can make early output files available before the end of the job, ready after the completion of the last `TEMPLATE_BLOCK` with layers in the output. This works on the output files specified in the `PROTEUS_JOB_FLOW` (PJF) section, although they can be part of a staged output which is also specified in one or more `END_TEMPLATE_BLOCK` sections. You must specify

`NEW_EARLY_OUTPUT_FROM_PJF ON` in the job control file to enable early output defined in the `PROTEUS_JOB_FLOW` section. `PROTEUS_JOB_FLOW` syntaxes conforming to the required support syntaxes described in OUTPUT and END_OUTPUT on page 64 cause the tool to identify early outputs when this keyword is `ON`. Without this keyword, these situations will not cause early outputs.

Syntax

```
NEW_EARLY_OUTPUT_FROM_PJF ON|OFF
```

Options

`ON`

Enables early output from PJF.

`OFF`

Disables early output from PJF. This is the default.

See also

OUTPUT and END_OUTPUT on page 64

NEW_PJF_PARSER

Description

By default, the Proteus tool uses a new implementation of the `PROTEUS_JOB_FLOW` parser to better track layer data dependency between template calls. The new parser provides additional robustness, especially for layer assignment.

To use the original parser, set `NEW_PJF_PARSER OFF`.

Syntax

```
NEW_PJF_PARSER [ON | OFF]
```

Options

`ON`

Uses the new implementation of the `PROTEUS_JOB_FLOW` parser. This is the default.

`OFF`

Turns off the new parser and uses the original parser.

OUTPUT and END_OUTPUT

Description

The OUTPUT section defines how output groups are placed into a generated output file, and assigns layers and datatypes.

OUTPUT can be defined anywhere inside a PROTEUS_JOB_FLOW statement or at the end of each TEMPLATE_BLOCK, as part of the END_TEMPLATE_BLOCK section (known as *staged output*; see TEMPLATE_BLOCK and END_TEMPLATE_BLOCK on page 81 for details).

It is possible to specify an OUTPUT/END_OUTPUT section multiple times to create multiple output files in the PROTEUS_JOB_FLOW section or the END_TEMPLATE_BLOCK sections. Each of these OUTPUT sections can specify different filenames. Global output file settings in the recipe (like DBU_OUT, for example) are applied to all output files and need not be specified individually in multiple OUTPUT sections. If they are specified individually in an OUTPUT section, this value will take precedence over the global setting. Pass-around layers and text records can also be specified for multiple output files.

The output format (GDSII or OASIS) specified in a TEMPLATE_BLOCK must match that of the PROTEUS_JOB_FLOW section or an error is generated. When OUTPUT is defined in an END_TEMPLATE_BLOCK section, the PATH keyword is not allowed.

Early completion of output files is possible in cases where all layers defined for an output file are available before the last TEMPLATE_CALL in a PROTEUS_JOB_FLOW recipe using the staged output syntax or the PROTEUS_JOB_FLOW output syntax. (See TEMPLATE_BLOCK and END_TEMPLATE_BLOCK on page 81 for details on the staged early output syntax, or NEW_EARLY_OUTPUT_FROM_PJF on page 62 for details on the PROTEUS_JOB_FLOW earlyoutput syntax.) In these situations, the Proteus tool automatically identifies which output files could be produced early and completes them near the end of the TEMPLATE_CALL when the last-available layers are consumed by the output file.

The output format is defined according to the following rules:

- If the output format (GDSII or OASIS) is defined in the `OUTPUT` section of the `PROTEUS_JOB_FLOW` statement, that is used as the output format.
- If the `PROTEUS_JOB_FLOW` statement does not specify the output format, but `OUTPUT_FORMAT` is specified, then the value of `OUTPUT_FORMAT` is used.
- If neither the `PROTEUS_JOB_FLOW` statement nor `OUTPUT_FORMAT` specify the format, the input format (whether specified in the recipe or automatically determined by scanning the input file) is used as the output format.

The output file names and format in the `PROTEUS_JOB_FLOW` section and the `TEMPLATE_BLOCK`s must match or an error is generated.

Syntax

```
OUTPUT [output_format] output_file_or_alias|CATS2
  [PATH output_file]
  [optional_output_properties]
  [layer[:datatype] = output_name_1]
  [layer[:datatype] = output_name_2]
  [layer[:datatype] = output_name_n]
END_OUTPUT
```

Options

`output_format`

Optional. Can be GDSII or OASIS. For a single input file, if *`output_format`* is not defined and `OUTPUT_FORMAT` has not been specified in the job control file, the Proteus tool assumes the same format as the input.

When there are multiple input files, if any input is OASIS and `OUTPUT_FORMAT` has not been specified in the job control file, the output is OASIS. If all inputs are GDS and `OUTPUT_FORMAT` has not been specified, the output is GDS.

This option is not allowed with the `CATS2` option.

`output_file_or_alias`

Always optional within `END_TEMPLATE_BLOCK`.

Can be either the user-defined name of the output file or an alias, which is resolved by the argument to the `PATH` keyword. (See PATH on page 71.)

This option is not allowed with the `CATS2` option.

`CATS2`

Optional. When this is present, the Proteus tool invokes the Proteus-CATS Exchange interface (PCX2) and the `OUTPUT` arguments define layers of interest to the CATS application and the layers:datatypes on which they will be found in the fragment files. If inputs have both GDSII and OASIS format, the output is OASIS.

When using PCX2 from internal `TEMPLATE_BLOCK`s or with multiple output files, the presence of an `OUTPUT CATS2` section is required.

When `CATS2` is present, the *`output_file_or_alias`* option is not allowed.

Note: The presence of an `OUTPUT CATS2` section takes precedence over the `PROTEUS_EXCHANGE` keyword. If the `OUTPUT CATS2` section is not present and `PROTEUS_EXCHANGE CATS2` is present, the older PCX2 behavior is retained.

`optional_output_properties`

Optional parameters to adjust the properties of the output file on an individual basis. Includes the following possible keywords:

- `OASISOUT_COMPACTION`
- `OASISOUT_MODALS`
- `OASISOUT_ZLIB_LEVEL`
- `DBU_OUT`
- `ROTATE_OUT`
- `SCALE_OUT`
- `TOPCELL_OUT`
- `STR_PREFIX`

Note: If `DBU_OUT` is not specified per output file, the `DBU_OUT` specification from outside the `OUTPUT` /`END_OUTPUT` section applies.

These settings can be specified in the `END_TEMPLATE_BLOCK OUTPUT` section and the `PROTEUS_JOB_FLOW OUTPUT` section. An error is generated if conflicting settings are specified for the same output file.

`OUTPUT CATS2` can only take `DBU_OUT` and `SCALE_OUT`.

See Output File Parameters on page 107 for details on these keywords.

output_name

Either the data flow name (as defined in the `INPUT` section) or “UNUSED.” (UNUSED is the same as not listing a layer.)

Keep in mind the following: if multiple different *output_names* are assigned to the same layer number (for example, 20:0 = layer_x in a `TEMPLATE_BLOCK OUTPUT` statement, and 20:0 = layer_y in another `TEMPLATE_BLOCK OUTPUT` statement or in an `OUTPUT` section), the graphics on these layers will be merged onto the same layer in the output file (layer 20:0 in the example).

layer

An integer value. The layer and datatype for OASIS files can be an integer value from 0 to 2^{31} - 2, and for GDSII files from 0 to 32767.

datatype

An integer value. The layer and datatype for OASIS files can be an integer value from 0 to 2^{31} - 2, and for GDSII files from 0 to 32767. The default is 0.

Examples

Example 1: One Output File

```
PROTEUS_JOB_FLOW
...
RESULT = TEMPLATE_CALL(BLOCK1(IN_LAYERS))
OUTPUT OASIS output.oas
  3:0  = RESULT[out_layer3]
  4:0  = RESULT[out_layer4]
  6:0  = UNUSED
END_OUTPUT
END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK BLOCK1 (LAYER l1, LAYER l2)
  out_layer1 = bias (10, l1)
  out_layer2 = bias (20, l2)
  out_layer3 = bias (30, l1)
  out_layer4 = bias (40, l2)
END_TEMPLATE_BLOCK
(
LAYER out_layer1,
LAYER out_layer2
OUTPUT OASIS output.oas
  2:0 = out_layer3
  23:0 = out_layer4
END_OUTPUT
)
```

Example 2: Multiple Output Files

```
DBU_OUT 1.0

PROTEUS_JOB_FLOW
...

out1 = TEMPLATE_CALL( BLOCK1( in_layer1, in_layer2))

OUTPUT GDSII output1.gds
  DBU_OUT 2.0
  1:0  = out1[0]
END_OUTPUT

out2 = TEMPLATE_CALL( BLOCK2(out1[0]))

OUTPUT OASIS output2.oas
  OASISOUT_COMPACTION ON
  OASISOUT_MODALS     ON
  OASISOUT_ZLIB_LEVEL  5
  2:0  = out2[0]
END_OUTPUT

END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK BLOCK1 (LAYER in1, LAYER in2)
  ...
END_TEMPLATE_BLOCK (
  LAYER first_out
)

TEMPLATE_BLOCK BLOCK2 (LAYER second_in)
  ...
END_TEMPLATE_BLOCK (
  LAYER second_out
)
```

Example 3: Multiple Output Files with PCX2

```
OUTPUT GDSII ./opc_out1.gds
DBU_OUT dbu_out1
SCALE_OUT scale_out1
  layer1:datatype1 = pass_layer1
  layer2:datatype2 = OPC_OUTLIST [0]
END_OUTPUT

OUTPUT OASIS ./opc_out2.oas
DBU_OUT dbu_out2
SCALE_OUT scale_out2
  layer3:datatype3 = OPC_OUTLIST [1]
END_OUTPUT

OUTPUT CATS2
DBU_OUT dbu_out3
SCALE_OUT scale_out3
  layer3:datatype3 = pass_layer3
  layer4:datatype4 = OPC_OUTLIST[0]
END_OUTPUT
```

Example 4: Early Output from `PROTEUS_JOB_FLOW`

Note that the `NEW_EARLY_OUTPUT_FROM_PJF` keyword must be `ON` in the job control file in order to get early output.

```
NEW_EARLY_OUTPUT_FROM_PJF

PROTEUS_JOB_FLOW
```

```
...
  OUT1 = TEMPLATE_CALL( BLOCK1(in_layer1))
  OUT2 = TEMPLATE_CALL( BLOCK2(OUT1[0])

OUTPUT OASIS earlyoutput.oas
    2:0 = OUT1[0] 'This could also be a named layer, e.g. OUT1[name]
END_OUTPUT

OUTPUT OASIS ./mainoutput.oas
    39:0 = OUT2[0]
END_OUTPUT

END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK BLOCK1 (LAYER first_in)
  Writelayer1 = ...
END_TEMPLATE_BLOCK
(
  LAYER Writelayer1
)

TEMPLATE_BLOCK BLOCK2 (LAYER second_in)
  Outlayer1 = ...
END_TEMPLATE_BLOCK
(
  LAYER Outlayer1
)
```

See also

NEW_EARLY_OUTPUT_FROM_PJF on page 62

NEW_MULTIPLE_OUTPUT_FILES on page HIDDEN

Output File Parameters on page 107

PATH on page 71

PROTEUS_EXCHANGE on page 262

TEMPLATE_BLOCK and END_TEMPLATE_BLOCK on page 81

PATH

Description

The absence or presence of the `PATH` keyword within `OUTPUT/END_OUTPUT` section of the `PROTEUS_JOB_FLOW` section of a recipe determines whether the name specified with `OUTPUT` keyword is an alias. If `PATH` is absent, the

OUTPUT name is assumed to be a real file name. If PATH is present, the OUTPUT name is assumed to be an alias and the name following the PATH keyword is the real output file name.

Syntax

See OUTPUT and END_OUTPUT on page 64

Example

```
PROTEUS_JOB_FLOW
...
OUTPUT OASIS myalias
  PATH output.oas
  3:0  = out_layer1
  4:0  = out_layer2
  6:0  = UNUSED
END_OUTPUT
END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK BLOCK1 (LAYER l1, LAYER l2)
...
END_TEMPLATE_BLOCK(
...
OUTPUT OASIS myalias
  2:0 = out_layer3
  23:0 = out_layer4
END_OUTPUT
)
```

See also

OUTPUT and END_OUTPUT on page 64

PBC_COVER_LAYER

Description

Use PBC_COVER_LAYER to input a layer, or layers, of drawn shapes from which the tool will select array candidate cells. Cells with instances that intersect the drawn cover layer shapes are candidates for periodic boundary condition (PBC) array creation. PBCs create arrays of highly repetitive hierarchy to optimize correction for large array processing. This function also selects any instances of array candidate cells in the cell hierarchy that intersect the drawn cover layer.

Note: `PBC_COVER_LAYER` and other periodic boundary correction (PBC) keywords are not supported with `MARK`, `COMPACT_CONTEXT`, `HIERARCHY_SKELETON`, or any `PIPELINE_STRATEGY` mode other than `NONE`.

Syntax

`PBC_COVER_LAYER` *`layer`*`[:`*`datatype`*`] {`*`layerN`*`[:`*`datatypeN`*`]} | UNUSED`

Options

`layer`

An integer value. Specify a value for the layer and datatype for OASIS files from 0 to 2^{31} - 2, and for GDSII files from 0 to 32767.

`datatype`

An integer value. Specify a value for the layer and datatype for OASIS files from 0 to 2^{31} - 2, and for GDSII files from 0 to 32767. The default is 0.

`UNUSED`

Turns off `PBC_COVER_LAYER` functionality, as if you removed the keyword from the recipe.

Example

```
PROTEUS_JOB_FLOW
INPUT ic_top.gds
     cor_layer = 5
     cover_layer = 11:0
     PBC_COVER_LAYER 11:0
END_INPUT
```

Figure 14 shows a non-rectangular cover layer shape drawn over an array, with a mirrored placement of the same array and cover layer shape. Hierman processes the cover layer shape, and afterward, uses its bounding box as the final cover layer shape for instance selection.

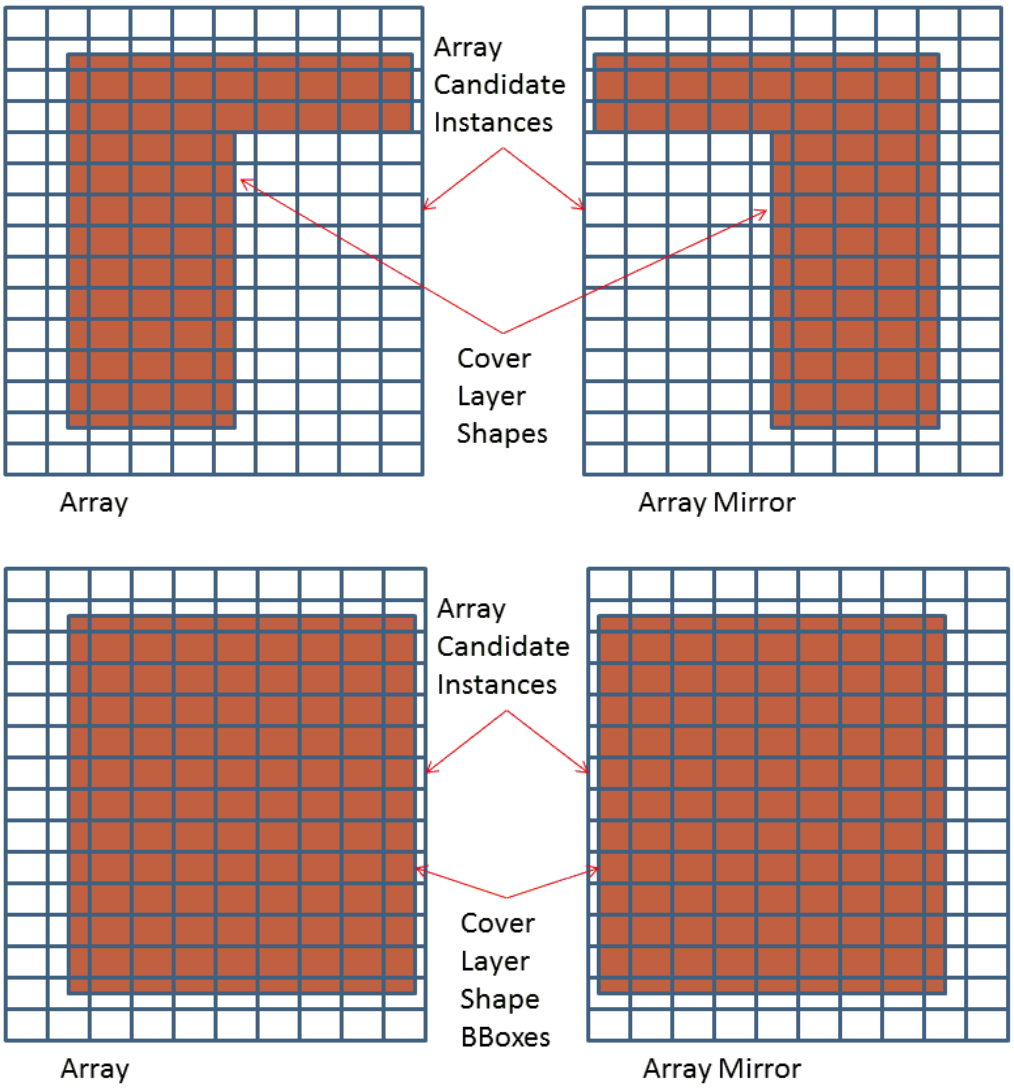


Figure 14 Cover Layer Shape Processing

Figure 15 shows an example where the selected array instances used for array creation are limited by their intersection to the cover layer shape bounding boxes. Array generation proceeds normally, creating boundary and bridge cells, and allowing bridge cells between PBC array cores, even when they do not intersect the cover layer shapes.

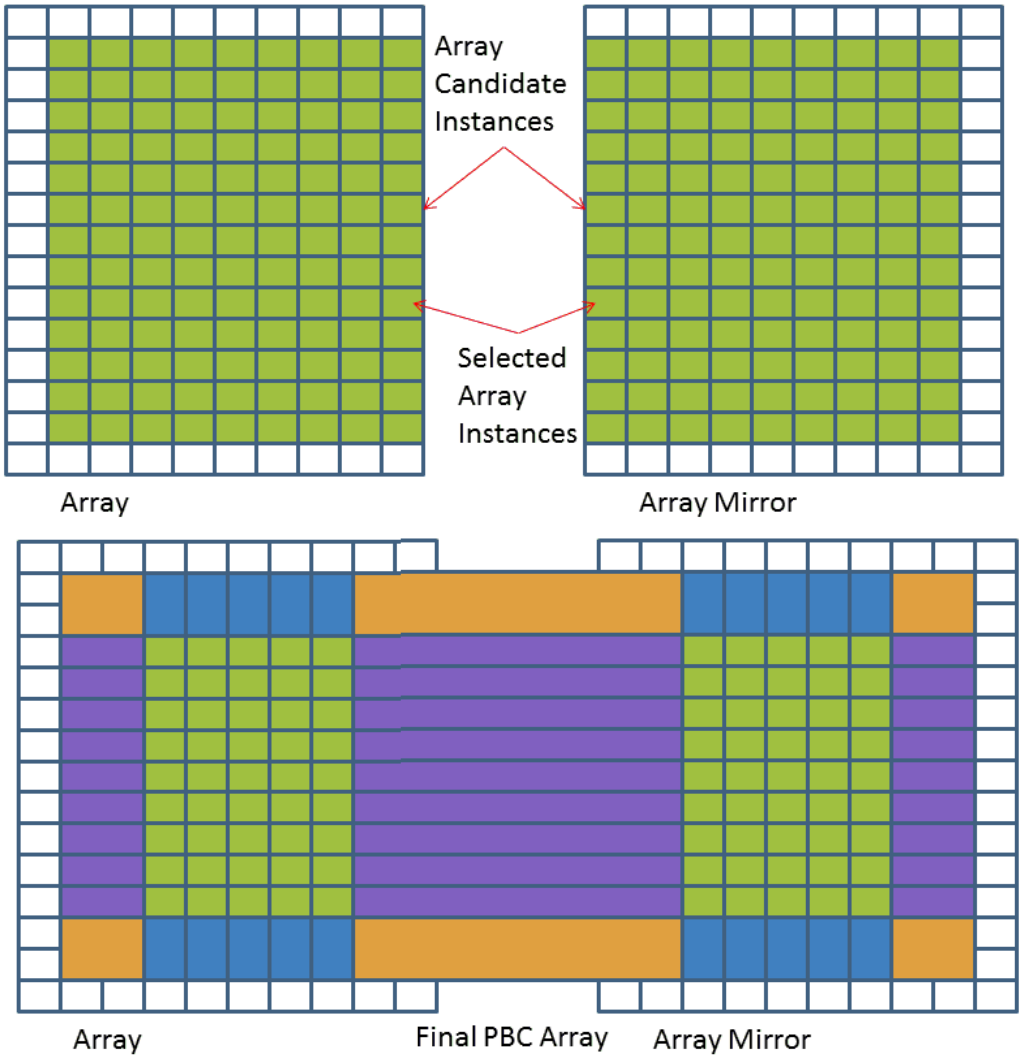


Figure 15 Array Creation With Cover Layer Shapes

PROTEUS_JOB_FLOW and END_PROTEUS_JOB_FLOW

Description

The `PROTEUS_JOB_FLOW` section of the job control file defines the chip-level flow for your Proteus job. Within `PROTEUS_JOB_FLOW`, the flow of events is defined, including inputs, outputs, and order of execution of template blocks or

full-chip layer operations (such as Boolean operations). In the recipe, this information is contained in the `.flow` file.

The maximum line length within the `PROTEUS_JOB_FLOW` section is 32K.

Syntax

```
PROTEUS_JOB_FLOW
INPUT [GDSII|OASIS] input_file
...
END_INPUT

[INPUT [GDSII|OASIS] input_file
...
END_INPUT]

in_array_name = [ \
...           \
]
out_array_name = TEMPLATE_CALL(tb_name([in_array_name]))

OUTPUT [GDSII|OASIS] output_file_or_alias
...
END_OUTPUT
END_PROTEUS_JOB_FLOW
```

Options

`input_file`

The user-defined name of the input file. It is possible to have multiple input files using separate `INPUT` declarations.

`in_array_name`

Optional. The user-defined name of the input layer array.

`out_array_name`

The user-defined name of the output layer array.

`tb_name`

The user-defined name of the template block, which must match a `TEMPLATE_BLOCK` name defined in the recipe.

`output_file_or_alias`

The user-defined name of the output file or an alias, which is resolved by the argument to the `PATH` keyword. (See PATH on page 71.)

#PROTEUS_PYTHON_MODULE

Description

The TEMPLATE_BLOCK structure supports the use of Python calls from within the Boolean flow. All Python code must be contained within the `#PROTEUS_PYTHON_MODULE` and `#END_PROTEUS_PYTHON_MODULE` keywords in the job control file. (The `PYTHON_MODULE` and `END_PYTHON_MODULE` keywords have equivalent behavior.)

For more information about these keywords, see the *Python in Proteus User Guide*.

See also

TEMPLATE_BLOCK and END_TEMPLATE_BLOCK on page 81

RECIPE_BASIC and END_RECIPE

Description

The TEMPLATE_BLOCK structure supports the use of corBASIC function calls from within the Boolean flow. All corBASIC code must be contained within the `RECIPE_BASIC` and `END_RECIPE` keywords in the job control file.

For more information about these keywords, see the *corBASIC Reference Manual*.

See also

TEMPLATE_BLOCK and END_TEMPLATE_BLOCK on page 81

REPROCESS_CALL

Description

`REPROCESS_CALL` is similar to a `TEMPLATE_CALL`, but is used with a point-fix `TEMPLATE_BLOCK` to designate a specific region of a template for reprocessing (for example, a template boundary or hot spot).

Syntax

```
out_array_name = REPROCESS_CALL(repair_tb_name(in_layers),
   (BASE_REGION marker_layer)
   [, (REPROCESS_DISTANCE value)]
   [, (override_params)])
```

Options

out_array_name

The user-defined name of an array to hold the layers output from the TEMPLATE_BLOCK. The number of elements placed in the array will match the number of layers output by the TEMPLATE_BLOCK.

repair_tb_name

The user-defined name of the TEMPLATE_BLOCK designated for reprocessing.

in_layers

A comma-separated list of the user-defined names of the input layers.

BASE_REGION *marker_layer*

Required. Identifies *marker_layer* as the area of interest.

REPROCESS_DISTANCE *value*

Optional. Defines the distance from the boundary of the current template, within which context graphics from a neighbor can be considered for processing within the current template. The graphics under consideration must be covered by a *marker_layer* that is defined as a BASE_REGION.

The default is 300, which means that context graphics within 300 nanometers (and covered by the *marker_layer*) of the boundary of the current template can be taken from a neighbor to be processed within the current template.

value should be smaller than the value of ambit. OVERRIDE_HIER_AMBIT/COR_AMBIT should also be set to the “correction ambit + REPROCESS_DISTANCE”.

Context graphics that are within this distance from the main of the current template are considered for inclusion as part of main of the current template.

If a marked region from a neighboring template is included as part of main of the current template, it is subtracted from the main of the neighbor.

Any part of context that was not included as part of the current template will be included as main of a different template.

Note: RECIPE_GRAPHICS_EXTENSION is forced to 0, even if defined by the user.

override_params

Optional TEMPLATE_BLOCK-based override parameters, which allow you to override certain parameters used during context analysis. These include APPS_CURRENT_ITER, APPS_NUM_ITERS, BASE_GROUP, DBU_PROC, DBU_PROC_OUT, ENVIRONMENT_GROUP, OVERRIDE_COR_AMBIT, OVERRIDE_HIER_AMBIT, REMOVE_REFIN_CUTLINE, and SYMMETRY.

For example, OVERRIDE_HIER_AMBIT can be changed for each TEMPLATE_BLOCK to better control the actual context considered during context analysis. OVERRIDE_COR_AMBIT can be changed to control how much neighboring graphics are loaded at correction time by distributed processing.

Note: Floating point values are allowed for override values for OVERRIDE_COR_AMBIT, OVERRIDE_HIER_AMBIT, and RECIPE_GRAPHICS_EXTENSION. Although floating point values are accepted for OVERRIDE_*_AMBIT values, the floating point value is truncated to an integer. If the floating value had a non-zero fractional value, a warning message is issued.

Examples

In Figure 16, the red box around Template 1 denotes the region within REPROCESS_DISTANCE from the boundary of Template 1. The tool considers

graphics from neighboring templates within this box for processing in Template 1.

The orange box denotes a marker polygon that lies on the boundary between Templates 1 and 2. Only graphics under this polygon are processed in the `REPROCESS_CALL`.

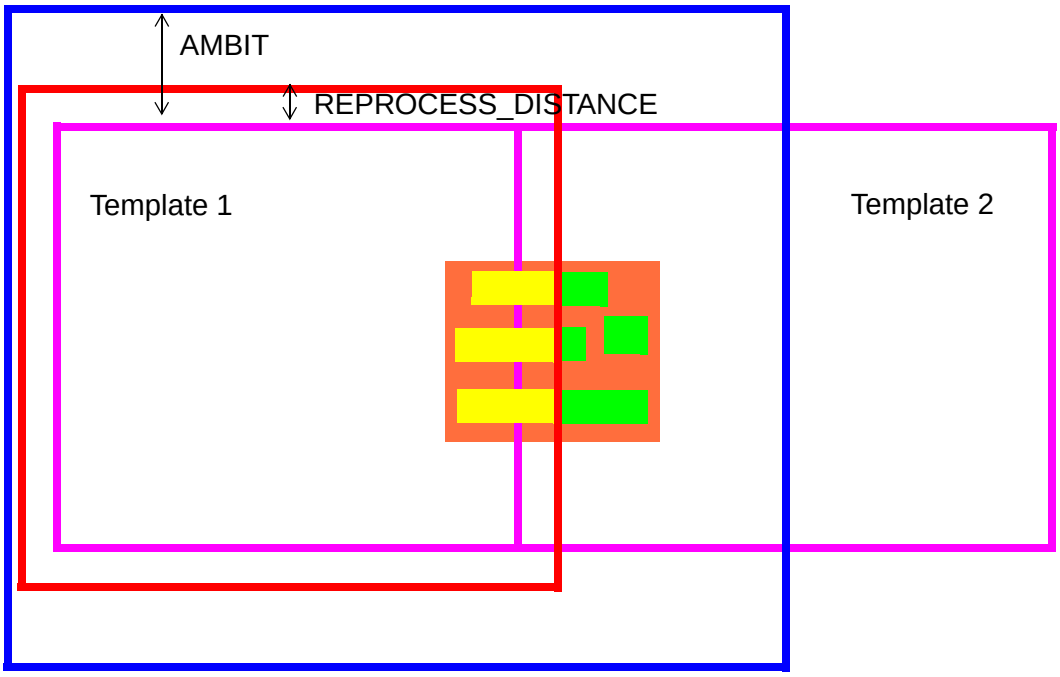


Figure 16 REPROCESS_DISTANCE

The following example uses a REPROCESS_CALL to designate a region of REPROCESS_BLOCK for reprocessing. The REPROCESS_DISTANCE is 250 nanometers.

```
PROTEUS_JOB_FLOW

INPUT test.gds
  cor_layer = 46
  marker_layer = 3
END_INPUT

OUTLIST = REPROCESS_CALL(REPROCESS_BLOCK
         (cor_layer, marker_layer), (BASE_REGION marker_layer),
         (REPROCESS_DISTANCE 250))

OUTPUT OASIS out.oas
  0 = OUTLIST[0]
  1 = OUTLIST[1]
END_OUTPUT

END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK REPROCESS_BLOCK(LAYER cor_layer, LAYER marker)

  from proteus import mlo2 as mlo

  cor_main = cor_layer.main

marker = marker.main

cor_layer = mlo.size(cor_main, size = 20)

END_TEMPLATE_BLOCK(LAYER cor_layer, LAYER marker)
```

TEMPLATE_BLOCK and END_TEMPLATE_BLOCK

Description

In the job control file, the TEMPLATE_BLOCK declaration marks the point at which the processing of a TEMPLATE_CALL or REPROCESS_CALL begins, while END_TEMPLATE_BLOCK marks the point where the processing of a TEMPLATE_CALL or REPROCESS_CALL ends. Between these two keywords, you use Boolean statements, including Python and corbasic calls, to define the operations performed on each template.

A TEMPLATE_BLOCK is executed when its name appears in a corresponding TEMPLATE_CALL or REPROCESS_CALL as part of the PROTEUS_JOB_FLOW. You can execute a single TEMPLATE_BLOCK multiple times as part of a flow by using a separate TEMPLATE_CALL or REPROCESS_CALL for each desired execution.

This structure supports multiple Python calls from within the Boolean flow. The execution order of Python functions is defined by the order of calls in the TEMPLATE_BLOCK section. Python code may be factored into modules with the #PROTEUS_PYTHON_MODULE and #END_PROTEUS_PYTHON_MODULE keywords to allow for reuse in other modules and TEMPLATE_BLOCKs. See the *Python in Proteus User Guide* for more information.

This structure also supports multiple corBASIC function calls from within the Boolean flow. The execution order of corBASIC functions is defined by the order of corbasic calls in the TEMPLATE_BLOCK section. All corBASIC code must be contained within the RECIPE_BASIC and END_RECIPE keywords in the job control file. See the *corBASIC Reference Manual* for more information.

TEMPLATE_BLOCK is the final stage of organizing the input data into process-specific groups. TEMPLATE_BLOCK combines polygons, referred to by global input layer or mark, together into the Boolean flow. The Boolean flow is where all pattern manipulations occur (merge, intersect, bounding box, corBASIC, and so forth).

It is possible to add an OUTPUT declaration to the TEMPLATE_BLOCK section. The OUTPUT statement must be put at the end of the END_TEMPLATE_BLOCK statement after all the layers are listed. This method, referred to as *staged output*, can add layers to an already existing output file that is declared elsewhere, or it can create a new output file. If a new output file is created inside the END_TEMPLATE_BLOCK, this results in the output file being finished near the completion of the template block and “early.” If layers are being added to an existing output file, the availability of this combined set of layers in the output file must be considered to determine whether the output will be early. You can use aliases with staged output. (See the following syntax statement.)

The tool does not accept a list element in the END_TEMPLATE_BLOCK section. Instead, you must assign each one of the output layers in the list to its own layer and then output those layers.

The tool flags characters not interpreted by the TEMPLATE_BLOCK parser as errors. The tool allows comment lines in the middle of the layer lists.

Syntax

TEMPLATE_BLOCK *tb_name* (

```
   OPTIONAL LAYER in_layer1,
   OPTIONAL LAYER in_layer2,
   LAYER in_layer3 'Required layer. This comment is legal.
  ...
)
#Boolean processing...
#Python or corBASIC calls...

END_TEMPLATE_BLOCK (
  LAYER out_layer1,
  LAYER out_layer2,
  LAYER out_layer3,
  LAYER out_layer4   'Note that there is no need for a comma
                     'after the last LAYER specification
                     'even if followed by an
                     'OUTPUT statement.
  [OUTPUT [output_format] [output_file_or_alias]
    ...
  END_OUTPUT]
)
```

Options

tb_name

The user-defined name of the given template block.

in_layer1, in_layer2, ...

The user-defined name of an input layer.

out_layer1, out_layer2, ...

The user-defined name of an output layer. If the `OUTPUT` keyword is present, zero or more layers are allowed, rather than one or more.

See also

OUTPUT and END_OUTPUT on page 64

TEMPLATE_CALL

Description

`TEMPLATE_CALL` is used within `PROTEUS_JOB_FLOW` to transfer control to a template-distributed section of code where each template is processed according to the set of instructions defined within the `TEMPLATE_BLOCK`. The `TEMPLATE_CALL` must be named and defined in the current file.

Template generation occurs at the beginning of each `TEMPLATE_CALL`. There might be a different number of templates, each representing a different number of instances, from one `TEMPLATE_CALL` to another. Template size (and therefore template boundaries) might change due to, for example, the addition of correction graphics.

You can choose to define only a subset of the input or output layers. Existing layers are maintained.

You can specify an edge as an input in the `TEMPLATE_CALL`. However, a previous `TEMPLATE_CALL` in the Proteus tool must have created the edge.

The order of layers or edges in the input or output lists is not important if you use name mapping. For backward compatibility, if the input layer or edge is passed in without any name mapping, the `TEMPLATE_CALL` treats the list as ordered and expects the same number of layers or edges in both the `TEMPLATE_CALL` and the `TEMPLATE_BLOCK`.

You can also refer to the elements of the array returned from a `TEMPLATE_CALL` by name. The name that should be used is the name of an output layer or edge from the corresponding `TEMPLATE_BLOCK`. For example, if OPC_OUTLIST is the array of layers output from a `TEMPLATE_CALL`, refer to OPC_OUTLIST[*layer1*] (or OPC_OUTLIST[*number*]).

Syntax

```
out_array_name = TEMPLATE_CALL(tb_name(in_array_name)
   [, (override_params)]
   [, TB_INSTANCE_NAME "tb_name"]
   [, (PREEXECUTE(Module.Function))]
   [, (POSTEXECUTE(Module.Function))])
```

Options

`out_array_name`

The user-defined name of an array to hold the layers output from the `TEMPLATE_BLOCK`. The number of elements placed in the array will match the number of layers output by the `TEMPLATE_BLOCK`.

`tb_name`

The user-defined name of the `TEMPLATE_BLOCK`.

`in_array_name`

The user-defined name of the input array. The number of elements in this array must match the number of input layers defined for the `TEMPLATE_BLOCK` *`tb_name`*.

override_params

Optional TEMPLATE_BLOCK-based override parameters, which allow you to override certain parameters used during context analysis. These include APPS_CURRENT_ITER, APPS_NUM_ITERS, BASE_GROUP, BASE_REGION, BLIND, CONTEXT_INSTANCE_DIAMOND_GRID, CREATE_EMPTY_TEMPLATES, DBU_PROC, DBU_PROC_OUT, DISABLE_TC_SNAP_SIZING, DPT_RANDOM_COLOR, ENVIRONMENT_GROUP, INVISIBLE, OVERRIDE_COR_AMBIT, OVERRIDE_HIER_AMBIT, PASSTHRU_GROUP, PLRC_DB, RECIPE_GRAPHICS_EXTENSION, REMOVE_REFIN_CUTLINE, REPROCESS_DISTANCE, SUPRESS_SYMMETRY_WARNING, SYMMETRY, SYNCHRONIZE, and TC_SCRIPT.

For example, OVERRIDE_HIER_AMBIT can be changed for each TEMPLATE_BLOCK to better control the actual context considered during context analysis. OVERRIDE_COR_AMBIT can be changed to control how much neighboring graphics are loaded at correction time by distributed processing.

Note: Floating point values are allowed for override values for OVERRIDE_COR_AMBIT, OVERRIDE_HIER_AMBIT, and RECIPE_GRAPHICS_EXTENSION. Although floating point values are accepted for OVERRIDE_*_AMBIT values, the floating point value is truncated to an integer. Note also that a warning message is issued if the floating value had a non-zero fractional value.

TB_INSTANCE_NAME "*tb_instance_name*"

This option allows you to define an instance name for a TEMPLATE_CALL so that TEMPLATE_CALLs made to the same TEMPLATE_BLOCK can be distinguished from one another.

If "*tb_instance_name*" is not specified, it defaults to *tb_name* (the name of the TEMPLATE_BLOCK).

You can use the proteus.info.parameter() function to return the value of TB_INSTANCE_NAME. For example:

```
tbname = proteus.info.parameter("TB_INSTANCE_NAME")
```

PREEXECUTE|POSTEXECUTE (*Module.Function*)

These options import a specified module (*Module*) and invoke a specified function (*Function*).

The `PREEXECUTE` option is invoked before any correction of templates for that `TEMPLATE_CALL` has started, and the `POSTEXECUTE` option is invoked after all templates for that `TEMPLATE_CALL` have completed correction. A `TEMPLATE_CALL` can have only one `PREEXECUTE` and one `POSTEXECUTE` option, although one of each can be present. In order to execute two functions before correction, for example, create a function that calls both of them and specify the new function as a `PREEXECUTE` option to the `TEMPLATE_CALL`.

Note: When using recovery (`proteus -f`) or `proteus -restart`, all `PREEXECUTE`/`POSTEXECUTE` statements from all `TEMPLATE_BLOCK`s are executed, regardless of the recovery or restart point. For example, if the job is started at `TEMPLATE_BLOCK` 2, any `PREEXECUTE`/`POSTEXECUTE` statements from `TEMPLATE_BLOCK` 1 will also be executed.

Keywords for Hierarchy Control

By default, the Hierarchy Manager treats all cells containing any of the layers defined in the `INPUT` statement as though they contain data to be corrected. (For more information, see Chapter 4, Hierarchy Management as well as the *Proteus User Guide*.)

You can improve correction efficiency by identifying which components of the pattern must be treated as correctable data, and which components can be treated only as context.

AREF_BRIDGE_DISTANCE

Description

During periodic boundary condition (PBC) processing, the tool creates bridge cells between matched PBC arrays (arrays with the same pitch and alignment of rows or columns). `AREF_BRIDGE_DISTANCE` specifies the maximum distance between arrays across which the tool creates a single bridge cell. Bridging creates a single array that tiles the region between two arrays and contains the original boundary arrays for each array.

Note: `AREF_BRIDGE_DISTANCE` and other periodic boundary correction (PBC) keywords are not supported with `COMPACT_CONTEXT`.

Syntax

`AREF_BRIDGE_DISTANCE` *`distance`*

Options

`distance`

Specify distance in 1x nanometers. The default is 3x `OVERRIDE_HIER_AMBIT`. If the distance specified for `AREF_BRIDGE_DISTANCE` is greater than `MAX_CLUSTER`, the tool ignores the `AREF_BRIDGE_DISTANCE` setting.

AREF_MIN_1D_SEPARATION_DISTANCE

Description

During periodic boundary condition (PBC) processing, `AREF_MIN_1D_SEPARATION_DISTANCE` specifies the minimum distance between non-bridged opposing PBC arrays.

Note: `AREF_MIN_1D_SEPARATION_DISTANCE` and other periodic boundary correction (PBC) keywords are not supported with `COMPACT_CONTEXT`.

Syntax

`AREF_MIN_1D_SEPARATION_DISTANCE` *`distance`*

Options

`distance`

Specify distance in 1x nanometers. The default is 10 database units (DBU).

AREF_MIN_1D_WIDTH

Description

During periodic boundary condition (PBC) processing, the PBC array is ringed by 1D boundary cells and corner cells. `AREF_MIN_1D_WIDTH` specifies the minimum 1D boundary cell width. During PBC classification and grouping, the tool does not use this parameter when it must reform arrays to find fully periodic

sets of 2D and 1D boundary.

Note: `AREF_MIN_1D_WIDTH` and other periodic boundary correction (PBC) keywords are not supported with `COMPACT_CONTEXT`.

Syntax

`AREF_MIN_1D_WIDTH` *`width`*

Options

`width`

Specify width in 1x nanometers. The default is `OVERRIDE_HIER_AMBIT`.

BASE_GROUP

Description

The `BASE_GROUP` keyword specifies a list of input groups for which correction templates are required. Cells containing data on any of the listed input group layers generate correction templates. Cells that have no data on any of the listed input group layers do not generate correction templates; their data appears only as context data in other correction templates. In addition, the size of the cell bounding box for receiving context is calculated only from the listed input groups. By default, all input groups are included in `BASE_GROUP`.

This keyword can be specified only within a `TEMPLATE_CALL`.

Syntax

`BASE_GROUP` *`input_group1`* [*`input_groupn`*]

Options

`input_group`

A previously declared input group.

Example

```
INLAYERLIST = [poly, diffusion]
TB1_OUTLIST=TEMPLATE_CALL(OPC_BLK(INLAYERLIST),
(BASE_GROUP INLAYERLIST[poly]))
```

By default, the two placements of cell A in Figure 17 have different environments because the diffusion areas are within an ambit of each other.

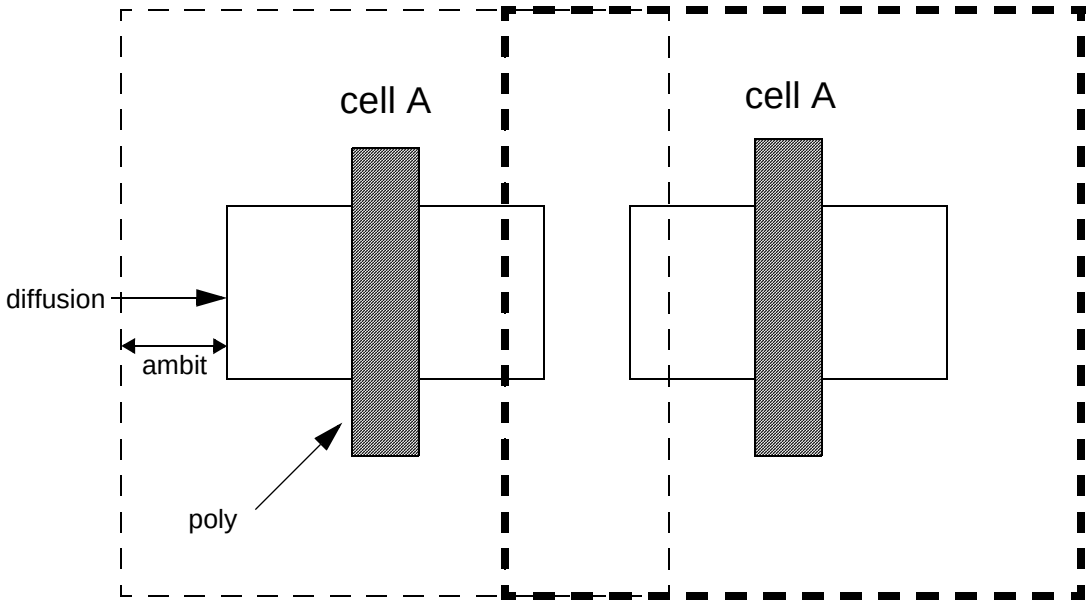


Figure 17 Context Bounding Boxes Calculated from All Source Groups

If you are correcting only poly, but using diffusion as reference (to locate gate regions), the line

```
BASE_GROUP poly
```

reduces the size of the area searched by cell A, as shown in Figure 18. As a result, the two placements of cell A now have the same environment.

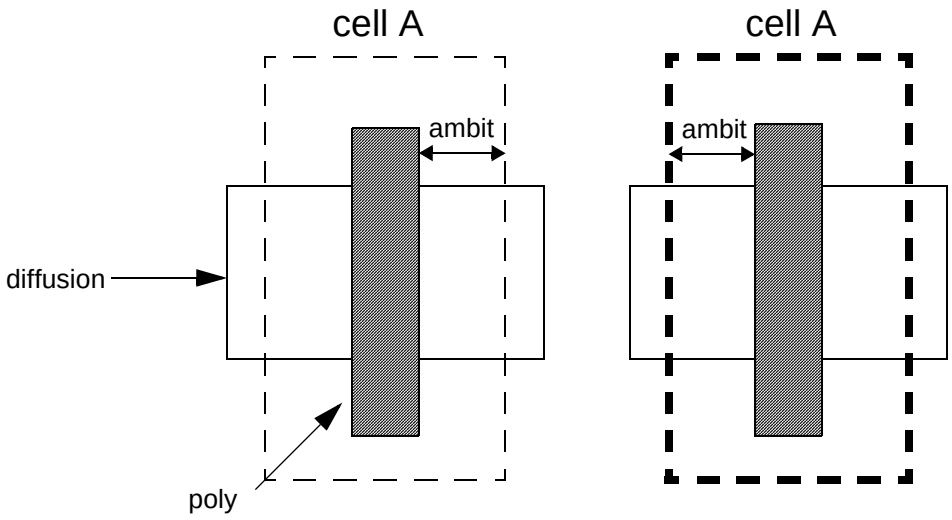


Figure 18 Reduced Context Bounding Boxes from Using BASE_GROUP

BLIND

Description

`BLIND` declares that the identified cell instances do not see their environment. As a result, these cell instances generate only one correction template, regardless of environments. `BLIND` cells have no context information during correction.

Syntax

`BLIND[!]` *`mark_name`*

Options

`mark_name`

A previously declared mark.

`!`

Indicates cell instances that are not identified by the *`mark_name`*.

ENVIRONMENT_GROUP

Description

Note: `ENVIRONMENT_GROUP` is not supported with `CLUSTER NONE`.

The `ENVIRONMENT_GROUP` statement declares which layers (input groups) to use for context calculations. Only data on layers defined by the listed input groups are included for context determination in the Hierarchy Manager. By default the tool includes all input groups.

You can specify this keyword only within a `TEMPLATE_CALL`.

Syntax

`ENVIRONMENT_GROUP` *`input_group1`* `[`*`input_groupn`* `...]`

Options

`input_group`

A previously declared input group.

INVISIBLE

Description

`INVISIBLE` declares that the identified cell instances are not seen in the environment of other cells, and therefore ignored for calculating context.

Syntax

`INVISIBLE[!]` *`mark_name`*

Options

`mark_name`

A previously declared mark.

`!`

Indicates cell instances that are not identified by the *`mark_name`*.

FORCE_TEMPLATE and END_FORCE_TEMPLATE

Description

Note: `FORCE_TEMPLATE` is not supported with `CLUSTER NONE`.

You can force particular cells to become templates by using these keywords to override internal clustering or scaffolding of cells.

A warning is issued if a named cell has a descendant that is also named as a `FORCE_TEMPLATE`. In this case, the cell that is lower in the hierarchy takes precedence and the cell that is higher in the hierarchy is removed from the `FORCE_TEMPLATE` list.

`FORCE_TEMPLATE` can also accept the asterisk (*) as a wildcard specification by cell name (not by path name).

Syntax

```
FORCE_TEMPLATE
cell_name1
[cell_namen ...]
END_FORCE_TEMPLATE
```

Options

`cell_name`

The cell being forced to become a template.

Example

An example is shown in Figure 19.

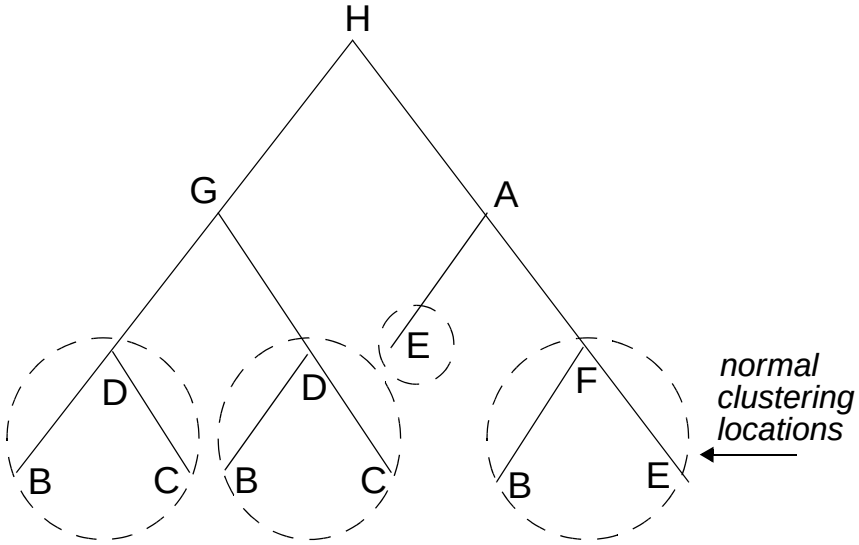


Figure 19 FORCE_TEMPLATE Input Hierarchy

Now force cells A and C to override the internal clustering:

```
FORCE_TEMPLATE
C
A
END_FORCE_TEMPLATE
```

In this example, clustering at cell D is overridden because cell C is named as a forced template. As a side effect, cell B also becomes a cluster. Additionally, cells E and F are overridden and clustering is forced at the larger cell A. The result appears in Figure 20.

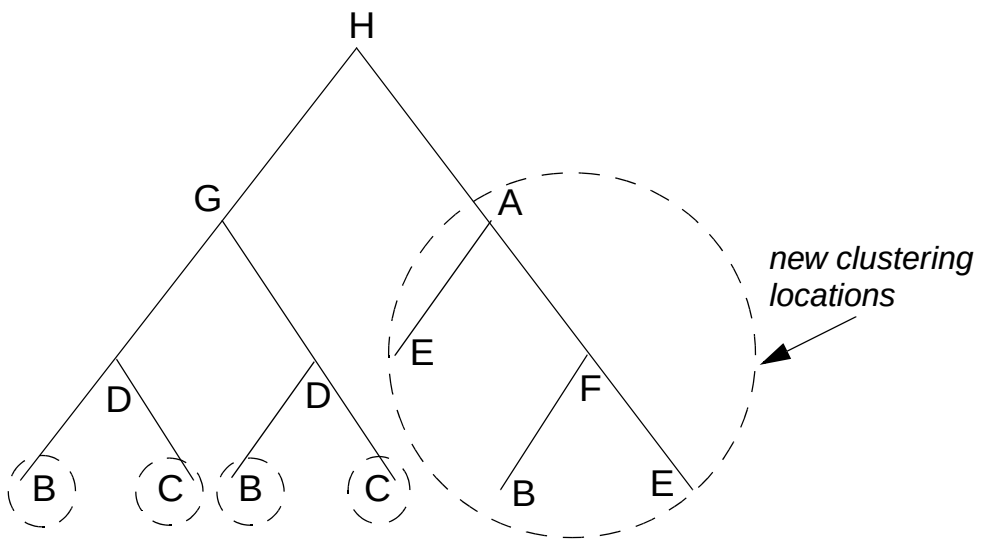


Figure 20 FORCE_TEMPLATE Output Hierarchy

The following is an example using the wildcard (*):

```
FORCE_TEMPLATE
  SUB*
END_FORCE_TEMPLATE
```

See also

CLUSTER on page 144

MARK and END_MARK

Description

Note: `MARK` is not supported with `CLUSTER NONE` or `PBC_COVER_LAYER`.

`MARK` identifies a subset of cell instances that require special treatment. Marked data can be identified during Boolean operations during correction, or can be assigned special treatment during hierarchy management.

This operation labels elements based on components of their cell name and hierarchy. Up to four marks can be applied to the input pattern at one time,

allowing for up to 16 distinct mark configurations with which to refine the pattern selection.

Polygons can then be referenced by any combination of global input layer and marks. In advanced applications, multiple global input layers and multiple marks can cross-index one another.

More than one mark can be applied to the same cell. The `MARK` state (marked or not) is accumulated as the list of mark elements is processed. Each mark is treated independently. Marks are applied to each cell instance and all polygons contained within them.

Marks can identify cells in three ways:

1. By the cell's position within the design hierarchy
2. By enclosure or intersection with an arbitrary rectangle
3. By enclosure or intersection with a global input layer

When defining a mark by method 2 or 3, you can specify whether it marks a cell by intersection or by enclosure.

Each correction template is homogenous in that every polygon has the same set of marks. The mark is also maintained through all `TEMPLATE_BLOCK`s.

Each mark element must be preceded by an operator type indicating whether the mark element is to be added to, subtracted from, or `AND`ed against the results of mark elements that have already been processed.

Syntax

```
MARK mark_name
  [+|-|&] [hier_node_id|rect_spec|cover_layer_spec]
END_MARK
```

Options

`mark_name`

The user-defined name of the mark.

`+`

Adds the cell instances to the mark element results if they match.

`-`

Subtracts the cell instances from the mark element results if they match.

`&`

Subtracts the cell instances from the mark element results if they do not match.

`hier_node_id`

Describes the hierarchy tree of a particular cell instance.

`hier_node_id` takes the form:

```
*.hier_node_id.* [*.text.*]
```

A period (.) indicates an explicit change in hierarchy level (a node), while an asterisk (*) is a wildcard match. A wildcard matches any number of characters, including node separators. An instance node path is created by appending `.cellname` at each node as the hierarchy tree is traversed, so you might have, for example:

```
topcell.firstcell.nextcell.sram.BitRow.quad.bit.halfbit
```

for the lowest level cell. The mark node paths can be specified to identify all instances of cells under the `BitRow` cell by `*.BitRow.*` or only those instances of cells under BitRow if BitRow falls under sram: `*.sram.BitRow.*` or `*.sram.*BitRow.*`. The node path separator (.) after `BitRow` makes this mark element refer explicitly to `BitRow`'s descendants. If a cell does not have descendants, it is not matched by this mark element. If it is not known whether a cell has descendants, two mark elements can be used to match either case.

For example:

```
+*.bitcell
+*.bitcell.*
```

marks all instances of `bitcell` and its descendants (if it has any).

`rect_spec`

Specifies a rectangle to a `MARK`.

`rect_spec` takes the form:

```
RECTANGLE x1 y1 x2 y2 [ENCLOSE|INTERSECT]
```

where:

- *x1* and *y1* specify the lower-left vertex and *x2* and *y2* specify the upper-right vertex of the rectangle.

- `ENCLOSE` specifies that the bounding box of an instance must be entirely enclosed by the rectangle to be marked. This is the default behavior.
- `INTERSECT` specifies that the bounding box of an instance must only intersect the rectangle to be marked. This is inclusive of touching edges.

`MARK RECTANGLE` only applies rectangle coordinates to unscaled GDS. The `MARK RECTANGLE` function is not affected by `SCALE_IN`.

`MARK RECTANGLE` uses the `INPUT` layers of the template cell to define the bounding box used to determine intersection or enclosure, regardless of whether `BASE_GROUP` or `ENVIRONMENT_GROUP` is specified.

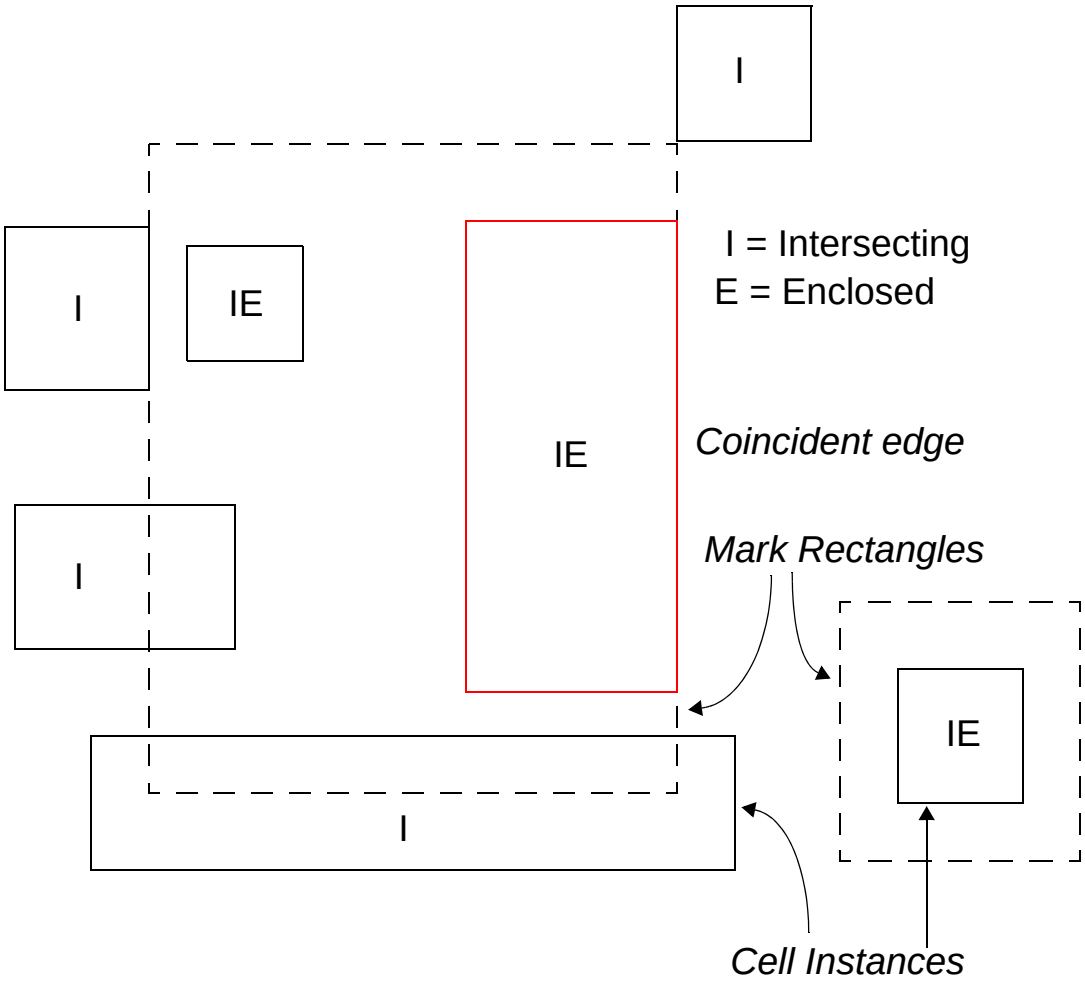


Figure 21 MARK RECTANGLE Examples

Pathological definitions of mark rectangles are accommodated as follows:

- It is legal for a rectangle to degenerate into a line. For example:

  ```
  + RECTANGLE 45900 27000 45900 28000 INTERSECT
  ```

  In this case, intersections are still possible.

- It is legal for a rectangle to degenerate into a point. For example:

  ```
  + RECTANGLE 45900 45900 45900 45900 INTERSECT
  ```

  In this case, intersections are still possible.

- It is *illegal* to define negative rectangles where the coordinates defining the rectangle do not have the proper lower-left and upper-right relationships. In such a case, the program exits with an appropriate error message. For example:

  ```
  + RECTANGLE 45900 27900 49900 16000 INTERSECT
  ```

  This example results in an error because the last number (y2=16000) cannot be smaller than the second number (y1=27900).

cover_layer_spec

Specifies a cover layer to a MARK. This makes it possible to derive the coordinates of the rectangle from the design automatically.

cover_layer_spec takes the form:

```
COVER_LAYER layer_name [ENCLOSE | INTERSECT]
```

where:

- *layer_name* specifies the name of the global input layer to be used for the cover layer.
- ENCLOSE specifies that the bounding box of an instance must be entirely enclosed by the COVER_LAYER to be marked. This is the default behavior.
- INTERSECT specifies that the bounding box of an instance must only intersect the COVER_LAYER to be marked. This is inclusive of touching edges.

Use MARK COVER_LAYER in conjunction with the MAX_CLUSTER override feature to mark specific chip regions or hierarchies where an alternate MAX_CLUSTER value is to be applied. (See MAX_CLUSTER on page 160.)

A message is printed to stdout stating how many cover layer polygons are generated when processing cover layer MARKs. This is typically a small number and usually correlates to large regions within a design (for example, a RAM area).

MARK COVER_LAYER uses the INPUT layers of the template cell to define the bounding box used for determining enclosure or intersection, regardless of whether BASE_GROUP or ENVIRONMENT_GROUP is specified.

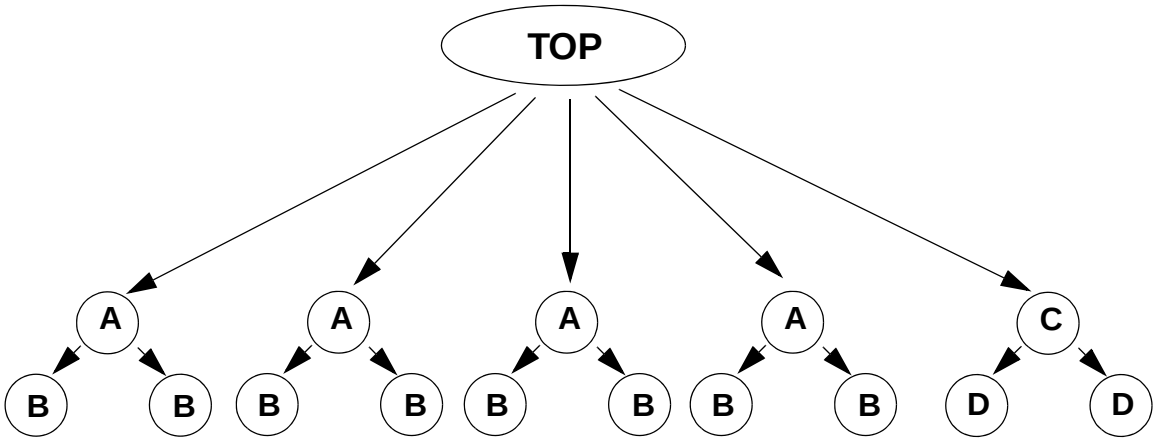
Example: MARK + COVER_LAYER

```
JOBNAME PROTEUS
BASEPATH ./hier_tmp/
OVERRIDE_HIER_AMBIT 1024
OVERRIDE_COR_AMBIT 1024
MIN_FEATURE 70
SIZE_CLUSTER 10000
MAX_CLUSTER 10000

' Example of MARK using a COVER_LAYER
MARK ram_part
  + COVER_LAYER ram_cover ENCLOSE
END_MARK

' Example of MAX_CLUSTER override using a MARK that has a
' COVER_LAYER
MAX_CLUSTER 500 ram_part

PROTEUS_JOB_FLOW
  INPUT GDSII ./polycover.gds
    metal1 = 1
    ram_cover = 3:0
  END_INPUT

  OUTLIST = TEMPLATE_CALL(Metal_OPC_BLK(metal1),  \
  (BASE_GROUP metal1))
' Specifying only the metal1 layer in the BASE_GROUP keeps
' templates from incorrectly using the ram_cover layer as part
' of content in template generation.

  OUTPUT GDSII ./output.gds
    1 = OUTLIST[metal_ram]
    2 = OUTLIST[metal_non_ram]
  END_OUTPUT
END_PROTEUS_JOB_FLOW
```

```
TEMPLATE_BLOCK Metal_OPC_BLK(LAYER metal_layer)
  metal_ram = metal_layer.filter("ram_part")
  metal_non_ram = metal_layer.filter("!ram_part")
END_TEMPLATE_BLOCK(LAYER metal_ram, LAYER metal_non_ram)
```

Figure 22 shows a simple hierarchy with a MARK called `ram_part` applied to the design with the following syntax:

```
MARK ram_part
  + COVER_LAYER ram_cover ENCLOSE
END_MARK
```



Cell B contains a rectangle on the cover layer ram_cover. See Figure 23.

Figure 22 Hierarchical Design

Figure 23 shows how the polygon in Figure 22 is represented at the top level of the cell.

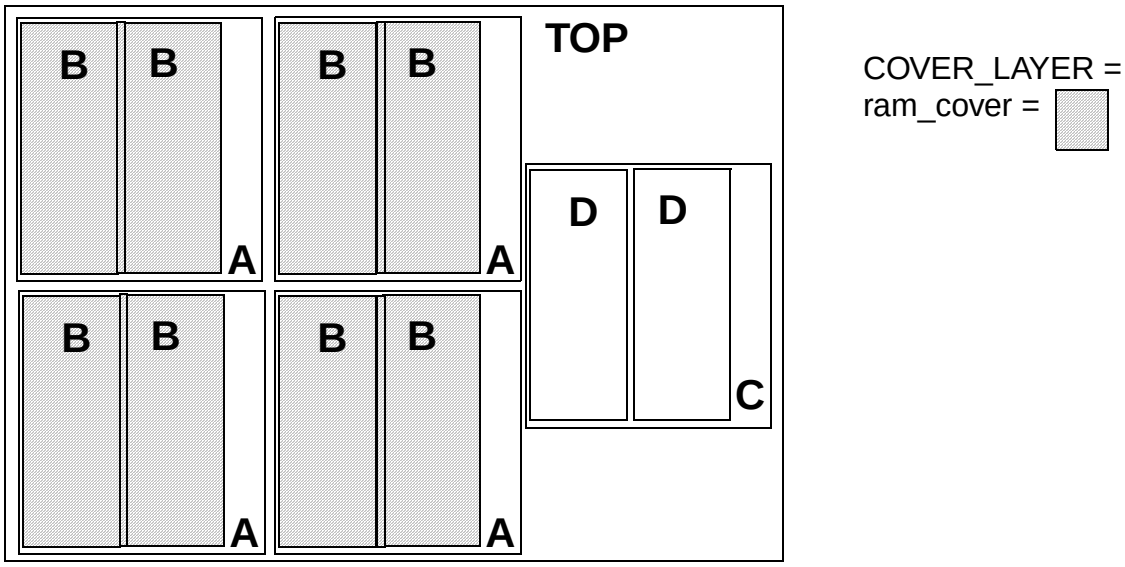


Figure 23 COVER_LAYER on Leaf Cell B

The leaf-level cell B in Figure 23 has a layer defined as `ram_cover` that is automatically expanded to a polygon with coordinates relative to the top-level cell of the design. A polygon (in this case a rectangle) is automatically generated on the layer `ram_cover` by combining the polygons of each instance of cell B.

Note: Cover layer polygons that touch each other are merged into one. This example results in four cover layer rectangles. (See MAX_CLUSTER on page 160.)

PBC_COVER_LAYER_MERGE_SIZE

Description

During periodic boundary condition (PBC) processing, you can specify this keyword to remove small gaps in the layers between shapes created from `PBC_COVER_LAYER` during cover-layer generation, based on the distance you specify.

Note: `PBC_COVER_LAYER_MERGE_SIZE` and other periodic boundary correction (PBC) keywords are not supported with `COMPACT_CONTEXT`.

Syntax

`PBC_COVER_LAYER_MERGE_SIZE` *`distance`*

Options

`distance`

Specify distance as 0 or any positive integer in nanometers. The default is 0.5 of `OVERRIDE_HIER_AMBIT` distance.

SUPPRESS

Description

`SUPPRESS` declares that the identified cell instances do not have templates generated from them. They are, however, seen in the environment of other cell instances.

This statement can occur only once in the job control file.

When the `SUPPRESS` and `INVISIBLE` keywords are both used (for the same marked region), the effect is as if the original cells in the marked region did not exist.

Syntax

`SUPPRESS [!]` *`mark_name`*

Options

`mark_name`

A previously declared mark.

`!`

Indicates cell instances that are not identified by the *`mark_name`*.

Example

```
MARK X
+ *.cellA.*
END_MARK

MARK Y
+ *.cellB.*
END_MARK

MARK Z
+ *.cellC.*
END_MARK

BLIND X
INVISIBLE Y
SUPPRESS Z
```

Assume all cells near each other are within the ambit.

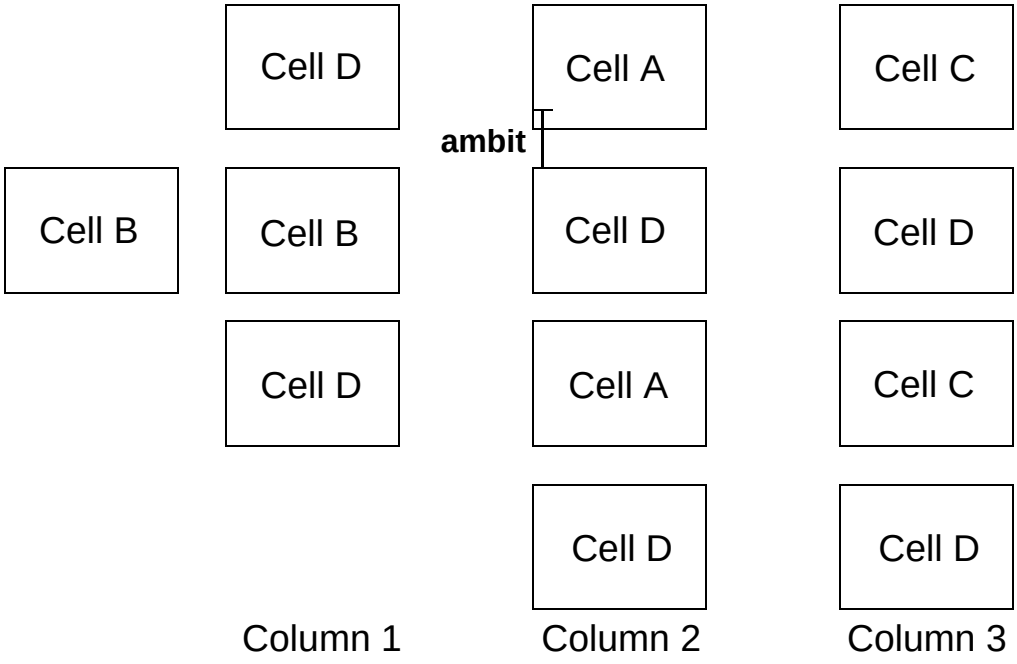


Figure 24 Cell Environments

In Column 1, normally, cell D would see cell B in its environment and create two templates. Cell B would also have two templates. With the `INVISIBLE` keyword as specified, cell D has only one environment, but cell B still has two.

In Column 2, normally, both cell A and cell D would each have two environments. But with the `BLIND` keyword as specified, cell A would have only one environment, while cell D would continue to have two unique environments.

In Column 3, normally, both cell C and cell D would each have two unique environments, and two templates created. With the `SUPPRESS` keyword as specified, cell D still has two unique environments and two templates, but cell C no longer has any templates generated for it.

Pattern Scaling with Fractional Grids

Pattern scaling with fractional grids observes the following guidelines:

- Arbitrary database units can be specified for the corrected output file.
- Output scaling is performed on the scaled input file coordinate data.
- Internal gridding algorithms minimize the effect of correction-grid snapping.

The following sections describe the parameters used for pattern scaling with fractional grids.

CORGRID

Description

When using `PROTEUS_JOB_FLOW` syntax, the `CORGRID` parameter is not used during correction.

Because `CORGRID` is still used by hierman, removing `CORGRID` from the recipe results in hierman using the default value of `CORGRID`.

Syntax

`CORGRID` *`value`*

Options

`value`

A value in nanometers.

DBU_IN

Description

This specifies the database unit of the input. This value is taken from the input file by default. Use the `DBU_IN` keyword to specify a floating-point internal processing database unit in the Proteus recipe if you do not wish to use the default. `DBU_IN` must be within 0.001 of the DBU of the input file.

Syntax

`DBU_IN` *`dbu`*

OR

`DBU_IN` *`numer denom`*

Options

`dbu`

The DBU of the input.

`numer`

The numerator when defining `DBU_IN` as a fraction.

`denom`

The denominator when defining `DBU_IN` as a fraction.

Example

To specify a long DBU, you can use the fraction option. For example,

```
DBU_IN            1.11111111111
```

is the equivalent of:

```
DBU_IN            1 .9
```

DBU_PROC

Description

This keyword specifies the database unit for all internal processing. The `DBU_PROC` value can be overridden for an individual `TEMPLATE_CALL`.

Syntax

`DBU_PROC` *`dbu`*

OR

`DBU_PROC` *`numer denom`*

Options

`dbu`

The DBU of all internal processing. The default is 1.0 nanometer.

`numer`

The numerator when defining `DBU_PROC` as a fraction.

`denom`

The denominator when defining `DBU_PROC` as a fraction.

DBU_PROC_OUT

Description

`DBU_PROC_OUT` allows you to specify the intermediate graphics file resolution in `PROTEUS_JOB_FLOW` recipes.

The `DBU_PROC_OUT` value can be overridden in each `TEMPLATE_CALL`. The specified value should have an integral ratio with the value of `DBU_PROC`.

Syntax

`DBU_PROC_OUT` *`dbu`*

Options

`dbu`

By default, this value is equal to the value of `DBU_PROC` of the current template block.

SCALE_IN

Description

This is the stream-in scaling factor, which specifies the input magnification. This specifies the magnification factor to scale the pattern to 1X. The default is 1.0.

Syntax

`SCALE_IN` *`scale_factor`*

OR

`SCALE_IN` *`numer denom`*

Options

scale_factor

The stream-in scaling factor.

numer

The numerator when defining `SCALE_IN` as a fraction.

denom

The denominator when defining `SCALE_IN` as a fraction.

Output File Parameters

The following section lists the keywords used to define output file parameters.

CELLNAME_MAX_LEN

Description

This keyword defines the maximum cellname length in the output file. The default length is 256 characters.

Syntax

`CELLNAME_MAX_LEN` *length*

Options

length

The maximum cellname length.

CORLIB

Description

This optional keyword overrides the default library name in the output file. The default library name is `CORRECTED`.

Syntax

`CORLIB` *user_library_name*

Options

user_library_name

The user-defined library name.

DBU_OUT

Description

This keyword specifies the database unit (distance per unit coordinate) of the output file in nanometers. The default value is 1.0 nanometer. `DBU_OUT` does not change the magnification of the output (performed using `SCALE_OUT`). Coordinate numbers in the output file are adjusted so that modifying the database unit does not change pattern scaling. All coordinates and cell placements are snapped to integer multiples of `DBU_OUT`. As a result, using a DBU that is incompatible with the data could result in gaps in the output.

Syntax

`DBU_OUT` *dbu*

OR

`DBU_OUT` *numer denom*

Options

dbu

The DBU of the output.

numer

The numerator when defining `DBU_OUT` as a fraction.

denom

The denominator when defining `DBU_OUT` as a fraction.

OASISOUT_COMPACTION

Description

Compaction creates smaller OASIS output files by creating repetitions of graphics in the output. `OASISOUT_COMPACTION` turns compaction on and off. The default is `ON`. If the `OASISOUT_MODALS` keyword is turned `OFF` no repetition or compaction is performed, regardless of this setting.

Note: In addition to using it as a global parameter, it is also possible to use `OASISOUT_COMPACTION` within the `OUTPUT/END_OUTPUT` section of the `PROTEUS_JOB_FLOW` section. It is not allowed within the `OUTPUT/END_OUTPUT` section of a `TEMPLATE_BLOCK`. If `OASISOUT_COMPACTION` is specified within the global section as well as in the `PROTEUS_JOB_FLOW OUTPUT` section, the latter overrides the former.

If OASIS input files have compaction or other differences in repetition structure, the method by which Hierarchy Manager performs scaffolding might be affected. This can result in slightly different template creation, which can affect correction results. Keep this potential effect in mind when comparing correction results.

Input GDS and OASIS files with identical data, hierarchy, and repetitions are treated the same by Hierarchy Manager and correction.

Note: This keyword behaves differently from the `CLUSTER` and `OASISOUT_ZLIB_LEVEL` keywords.

Syntax

```
OASISOUT_COMPACTION ON|OFF
```

Options

`ON`

Turns on compaction. This is the default.

`OFF`

Turns off compaction.

OASISOUT_MODALS

Description

This specifies the use of modal variables and allows relative coordinates in an output OASIS file. When set to `OFF`, the output does not use any modal variables, and only absolute coordinates are used. When set to `ON`, any modal variables can be used, and both absolute and relative coordinates are allowed in the output OASIS file. Turning modal variables off also forces compaction to be off (see OASISOUT_COMPACTION on page 108 for details).

Note: In addition to using it as a global parameter, it is also possible to use `OASISOUT_MODALS` within the `OUTPUT/END_OUTPUT` section of the `PROTEUS_JOB_FLOW` section. It is not allowed within the `OUTPUT/END_OUTPUT` section of a `TEMPLATE_BLOCK`. If `OASISOUT_MODALS` is specified within the global section as well as in the `PROTEUS_JOB_FLOW OUTPUT` section, the latter overrides the former.

Syntax

```
OASISOUT_MODALS ON|OFF
```

Options

`ON`

Turns on use of modal variables. This is the default.

`OFF`

Turns off use of modal variables.

OASISOUT_ZLIB_LEVEL

Description

This specifies the level of Zlib compression on an output OASIS file. The Zlib compression is applied to OASIS CBLOCK (compressed block) records with compression type 0 (currently the only type of compression OASIS allows). For more information on CBLOCK, refer to the OASIS standard documentation.

This keyword has no function when a GDSII file is being output.

Note: In addition to using it as a global parameter, it is also possible to use `OASISOUT_ZLIB_LEVEL` within the `OUTPUT/END_OUTPUT` section of the `PROTEUS_JOB_FLOW` section. It is not allowed within the `OUTPUT/END_OUTPUT` section of a `TEMPLATE_BLOCK`. If `OASISOUT_ZLIB_LEVEL` is specified within the global section as well as in the `PROTEUS_JOB_FLOW OUTPUT` section, the latter overrides the former.

Syntax

`OASISOUT_ZLIB_LEVEL` *`zlib_level`*

Options

`zlib_level`

A value from 0 to 9. 0 indicates no compression; 1 is minimum compression (least CPU cycles required); 9 is maximum compression (most CPU cycles required). If not specified, *`zlib_level`* defaults to 6.

OUTPUT_VERT_MAX

Description

This specifies the maximum number of vertices per boundary (or polygon, for OASIS) allowed in the output file. Setting this to a number larger than the default of 197 (for example, 600) can reduce the number of polygons in the correction output. The maximum setting is dependent on limitations of tools later in the process, or 2^{31}-1, whichever is smaller.

Syntax

`OUTPUT_VERT_MAX` *n*

Options

n

The maximum number of vertices per boundary (or polygon, for OASIS) allowed in the output file. The default is 197.

ROTATE_OUT

Description

The `ROTATE_OUT` JCL keyword allows you to rotate a layout around the origin on output, with the rotated layout snapped to `DBU_OUT`. The default is 0 degrees. You can specify the value as either positive or negatIve, and it must be a multiple of 90 degrees.

For additional control, you can add this keyword to output file definition section to allow file specific `ROTATE_OUT` settings. Otherwise, the setting is applied globally.

Examples

In the following example, `out3.oas` gets the global rotation of -90 degrees.

```
OUTPUT_OASIS/out3.oas
 SCALE_OUT 1
 TOPCELL_OUT TOP_TB1
 ROTATE_OUT -90
 28:40 =dummy_out
 28:20 =not_out
END_OUTPUT
```

In the following example `out1.oas` is rotated -90 degrees and `out2.gds` remains with the default rotation of 0 degrees.

```
ROTATE_OUT 0
OUTPUT OASIS ./out1.oas
 SCALE_OUT 0.5
 ROTATE_OUT -90
 28:40 =dummy_out
 28:20 =not_out
END_OUTPUT

OUTPUT GDSII ./out2.gds
TOPCELL_OUT TOP_TB1
 28:1 =cor_layer
 28:40 = dummy_layer
END_OUTPUT
```

See also

ROTATE_IN on page 211

OUTPUT and END_OUTPUT on page 64

SCALE_OUT on page 112

SCALE_OUT

Description

This is the stream-out scaling factor, which specifies the output magnification from the internal database dimension. The default is 1.0. The physical dimension of output data is set by this parameter alone.

Syntax

`SCALE_OUT` *`scale_factor`*

OR

`SCALE_OUT` *`numer denom`*

Options

`scale_factor`

The stream-out scaling factor.

`numer`

The numerator when defining `SCALE_OUT` as a fraction.

`denom`

The denominator when defining `SCALE_OUT` as a fraction.

STR_PREFIX

Description

This optional statement prepends *`prefix_string`* to all generated cell names in the output file (except `TOPCELL_OUT`).

Syntax

`STR_PREFIX` *`prefix_string`*

Options

`prefix_string`

The string to prepend to all generated cells in the output file. The maximum length is 6 characters. If the construction creates a cell name longer than this, the name is truncated (from the right).

See also

TOPCELL_OUT on page 113

TOPCELL_OUT

Description

This optional statement overrides the default top cell name in the output file. The default top cell corresponds to those in the original input file. This is reported at the end of the processing initiated with hierman.

You can increase the `TOPCELL_OUT` string length using the `CELLNAME_MAX_LEN` keyword. A user-defined `TOPCELL_OUT` top cell name does not prepend the `STR_PREFIX` characters.

If a recipe uses multiple topcells, some of which have hierarchy and some of which do not, and some of the topcells without hierarchy have text records or pass-around graphics that appear in the output file, then the cells containing text or pass-around graphics for the topcells with no hierarchy will appear as new topcells. The topcells can all be collapsed under a single topcell by specifying a topcell name using `TOPCELL_OUT` in the recipe.

Syntax

`TOPCELL_OUT` *`user_topcell_name`*

See also

CELLNAME_MAX_LEN on page 107

STR_PREFIX on page 113

TOPCELL_IN on page 219

OASIS Examples

The following examples demonstrate the differences between GDSII and OASIS keywords in a recipe.

For GDS input and output, you could use the following structure:

```
PROTEUS_JOB_FLOW
INPUT incoming.gds
...
END_INPUT

OUTLIST = TEMPLATE_CALL(TB_Name(...))

OUTPUT GDSII finished.gds
...
END_OUTPUT
END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK TB_Name(...)
...
END_TEMPLATE_BLOCK(...)
```

For OASIS input and output with Zlib compression level 6 and modal variables and relative coordinates allowed, you could use the following structure:

```
OASISOUT_MODALS ON
OASISOUT_ZLIB_LEVEL 6

PROTEUS_JOB_FLOW
INPUT incoming.oas
...
END_INPUT

OUTLIST = TEMPLATE_CALL(TB_Name(...))

OUTPUT OASIS finished.oas
...
END_OUTPUT
END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK TB_Name(...)
...
END_TEMPLATE_BLOCK(...)
```

Note that, in these examples, the input file format is automatically discovered from the file in the `INPUT` line. In addition, the output file format lines could have been omitted in each example because `output file format` defaults to the same type as the input file if the output file format is not specified.

Sample Applications Defining Groups

The following examples show a few basic applications. The groups defined in these examples include:

target

Forms the main or corrected component of correction.

recursion

Used as the recursion component of the correction template — meaning, context data that is to be corrected, but the copy of that data that is added to the periphery of each correction template (to be corrected but later discarded).

pass_thru

Any chrome that does not meet the cell name or mark criteria. It is intended to be merged with the corrected chrome at output.

pass_ref

All data in the correction template that is not to be corrected (either as target/main or recursion/context), but which must be present as reference for the model to estimate proximity effects correctly on nearby corrected chrome. Reference elements must be allowed for in both the target and recursion pattern components of a correction template, so no main or context distinction is made.

No Booleans are required in defining the groups for this example.

Example 1: Requirements for a Simple, One-layer Correction

The following example illustrates the minimum job control statements required to specify a simple, one-layer correction over the entire chip.

```
‘Note that no MARKs are needed.

PROTEUS_JOB_FLOW
‘get pattern from input layer 6
INPUT incoming.gds
chrome = 6
END_INPUT

OUTLIST = TEMPLATE_CALL(TB_Name(chrome))

OUTPUT finished.gds
23 = OUTLIST[corrected]
END_OUTPUT
END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK TB_Name(LAYER chrome)
target = chrome.main
recursion = chrome.context
#Define correction and recursion data in the corbasic() call.
#No reference data is used in this example.
corbasic (0, mainCorrectF(), \
          COR_IN:0, target, \
          REC_IN:0, recursion, \
          corrected, COR_OUT: 0)
END_TEMPLATE_BLOCK(LAYER corrected)
```

Example 2: Selected Cell Correction

Extend Example 1 to select a portion of the chip. The objective is to correct the selected cells (cell instances), and for the unselected cells to be considered for

context and passed to the output uncorrected. Changes to the previous recipe are shown in purple.

```
MARK corrected
     + *.RAM_ARRAY.*
     + *.TEST_CELL.*
     - *.DRIVE_AMP.*
END_MARK

BLIND !corrected

PROTEUS_JOB_FLOW
‘get pattern from input layer 6
INPUT incoming.gds
chrome = 6
END_INPUT

OUTLIST = TEMPLATE_CALL(TB_Name(chrome))

‘put corrected and pass-thru data onto output layer 23.
OUTPUT finished.gds
23 = OUTLIST[final]
23 = OUTLIST[pass_thru]
END_OUTPUT
END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK TB_Name(LAYER chrome)
target = chrome.main.corrected
recursion = chrome.context.corrected
pass_ref = chrome.!corrected

#Define correction and recursion data in the corbasic() call.
corbasic (0, mainCorrectF(), \
          COR_IN:0, target, \
          REC_IN:0, recursion, \
          REF_IN:0, pass_ref, \
          final, COR_OUT: 0)

#Pass through only .main of data to the output.
     pass_thru = chrome.main.!corrected

END_TEMPLATE_BLOCK(LAYER final, LAYER pass_thru)
```

Example 3: Brightfield Phase Mask Correction

Correct a brightfield phase mask where the input defines a chrome level and two phase levels (-90 and +90).

```
PROTEUS_JOB_FLOW
‘get pattern from input layer 6
INPUT incoming.gds
chrome = 6
phase_up = 17
phase_down = 18
END_INPUT

OUTLIST = TEMPLATE_CALL(TB_Name(LAYER), (BASE_GROUP chrome))

‘put corrected data (chrome only) onto output layer 23.
OUTPUT finished.gds
23 = OUTLIST[final]
END_OUTPUT
END_PROTEUS_JOB_FLOW

TEMPLATE_BLOCK TB_Name(LAYER chrome, LAYER phase_up, \
                       LAYER phase_down)
     #convert from brightfield to darkfield for correction.
     #goal is to provide corrected chrome, so phase operations
     #can be lumped together.
     ph_up   = phase_up - chrome
     ph_down = phase_down - chrome
     temp1 = phase_up + phase_down + chrome
     template_bb = bias(+2000, bb(temp1))
     ph_zero = template_bb - chrome
     #separate chrome for target and recurse
     chrome_targ = chrome.main
     chrome_recurse = chrome.context

#put chrome_targ into target data, all else into recursion
corbasic (0, mainCorrectF()   \
     COR_IN:0,  chrome_targ,    \
     REC_IN:0,  chrome_recurse, \
     REC_IN:1,  ph_zero,        \
     REC_IN:2,  ph_up,          \
     REC_IN:3,  ph_dn           \
     final, COR_OUT:0)

#note: output types could be enhanced to reconstruct phase to
     #chrome overlap margins.
END_TEMPLATE_BLOCK(LAYER final)
```

4

Hierarchy Management

Provides information on design hierarchy and the job control parameters used in the hierarchy management phase of Proteus jobs.

Hierarchy Management

You invoke hierarchy management using the following command:

`proteus [-options] job_control_file`

This executes proteus using the settings in your job control file. The proteus binary is a single executable supporting hierarchical processing and distributed correction. It accepts `PROTEUS_JOB_FLOW` recipes. See proteus on page 387 for more information and a list of available options.

The purpose of hierarchy management is twofold:

- provide a set of unique correction templates that make up the design
- modify the design hierarchy to try to optimize correction runtime

Correction templates combine content data (graphical information defining a portion of the design) and context data (data necessary to properly correct the content data). The context data is necessary because the presence of a graphical element can affect the printed image of other graphical elements over some distance of influence, called the ambit.

The Hierarchy Manager accepts and produces data conforming to GDSII Stream Release 6.0. GDSII is a database standard used to represent two-dimensional shapes useful for descriptions of circuit layouts. While GDSII allows both graphics and references in the same cell, after hierarchy processing in the Proteus tool, the graphics and references are separated.

In addition to GDSII, the Hierarchy Manager also accepts OASIS input files. OASIS is a file format created by the Semiconductor Equipment and Materials

Institute (SEMI) that has improvements over the GDSII file format, including a reduced file size. For information on OASIS-related Proteus keywords, as well as examples, see Chapter 3, Groups and Data Handling: PROTEUS_JOB_FLOW.

The hierarchic layout structure is represented as a list of structures that define a set of graphic elements. Relevant GDSII elements include:

GDSII Element	Description
BOUNDARY	A polygon.
PATH	A polyline of finite width.
SREF	A single-placement reference to another cell structure.
AREF	A multiple-placement reference to another cell structure.

Relevant OASIS records include:

OASIS Element	Description
POLYGON	A polygon.
TRAPEZOID	A trapezoid.
CTRAPEZOID	A trapezoid figure in compact form.
RECTANGLE	A rectangle.
PATH	A polyline of finite width.
PLACEMENT	A single-placement reference to another cell structure.
REPETITION	A multiple-placement reference to another cell structure.

Note: OASIS provides a large number of different record types. One of these is `circle`. Because the interpretation is ambiguous, an error occurs if a `circle` record is found in the input data. `Textstring`, `Layername`, `Xname`, `Xelement`, `Xgeometry`, and unknown `Property` records are ignored if they are present in the input data.

Hierarchy Concepts: Placement, Reference, Instance

A *placement* includes location, rotation, magnification, and mirror parameters.

A cell representation of a hierarchical layout file is shown in Figure 25. The first structure, CELL1, contains a set of path and boundary elements (E1) and a single placement reference to another cell structure, SREF CELL3. The cell structure CELL2 contains more geometric elements (E2), two single-placement references (SREF CELL1 and SREF CELL3), and an arrayed placement reference to CELL3 (AREF (2x2) CELL3). Closed cycles (loops) of references are not permitted. Because of this, there must be one or more cells that are not referenced by any other cells. Such cells are called *top cells*. In Figure 25, the only top cell is CELL2.

Figure 25 illustrates a convenient way to count the *references* in this design. Structure CELL1 contains one reference to CELL3, CELL2 contains one reference to CELL1 and two references to CELL3 (an AREF is always a single reference), CELL3 contains two references to CELL4, and CELL4 contains no references. The total number of references in this design, then, is six.

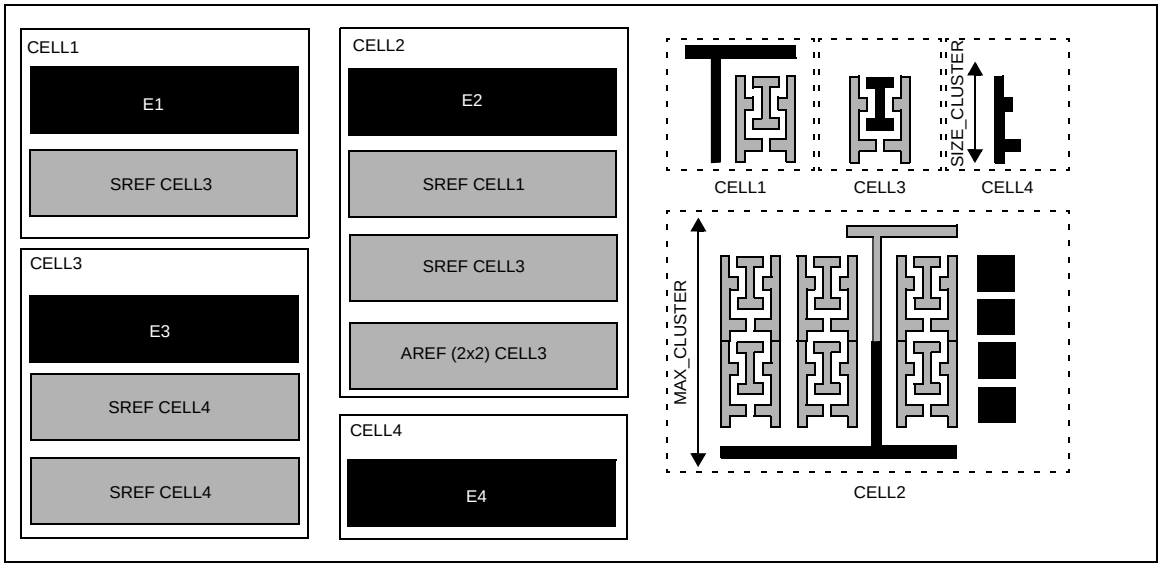


Figure 25 Cell-based View of Hierarchical Layout File

The expanded hierarchy of this sample file is shown in Figure 26. (Only one branch of the 2x2 AREF has been shown here, for compactness.) Because cycles are excluded, the hierarchy must take the form of a tree. The leaves of the tree represent *instances* of structures. The number of instances in a design

is different from the number of references; counting the total number of leaves and branches in the tree gives the number of instances. For this design, the number of instances is 20. The structures defined in the hierarchy are the candidate correction cells examined by the Hierarchy Manager to create a list of correction templates.

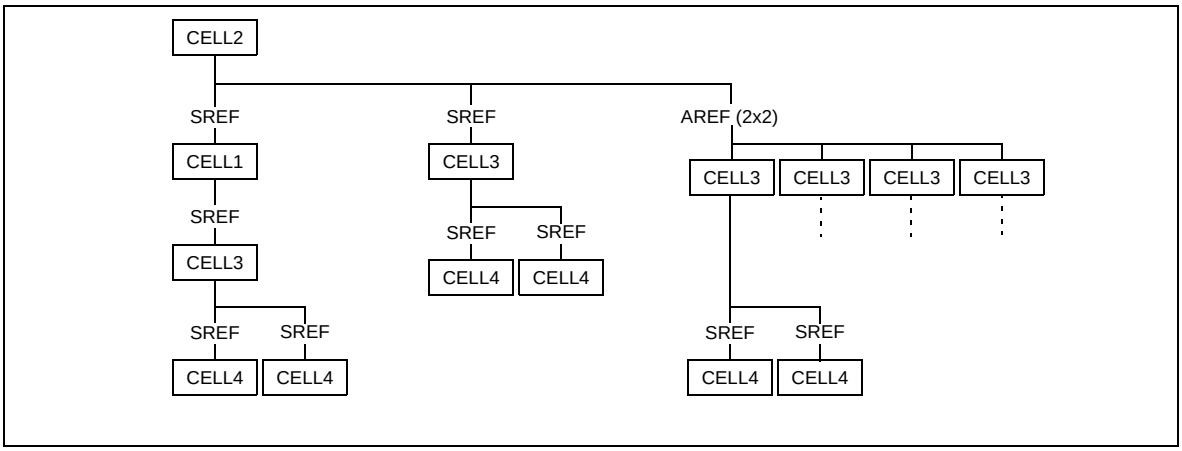


Figure 26 Hierarchical View of Layout File

In summary, *references* are what you find in the input GDSII or OASIS file, whereas *instances* are what get physically placed in the layout. References are counted only once, while instances are counted every time they appear.

Scaffolding and Clustering

The Hierarchy Manager performs several steps to construct a set of correction templates. Each template consists of content data to be corrected and kept, and context data needed to ensure the proper correction of the content data. The context data is typically corrected and then discarded. The most critical (and most time consuming) step in hierarchy management is context calculation.

Context calculation enables the Proteus tool to represent multiple instances of the same structure with a single structure in the output as long as the data in the context region is identical. Prior to that point, however, several hierarchical manipulations (including scaffolding and clustering) are done to build a hierarchy that is conducive to rapid context calculation.

Scaffolding (except for flat scaffolding) typically adds structures to the input data.

Leaf Scaffolding

The first step in hierarchy management is to create only two types of cells: those that contain only graphical elements (graphics cells), and those that contain only references to other cells (holder cells). Holder cells do not need to be corrected.

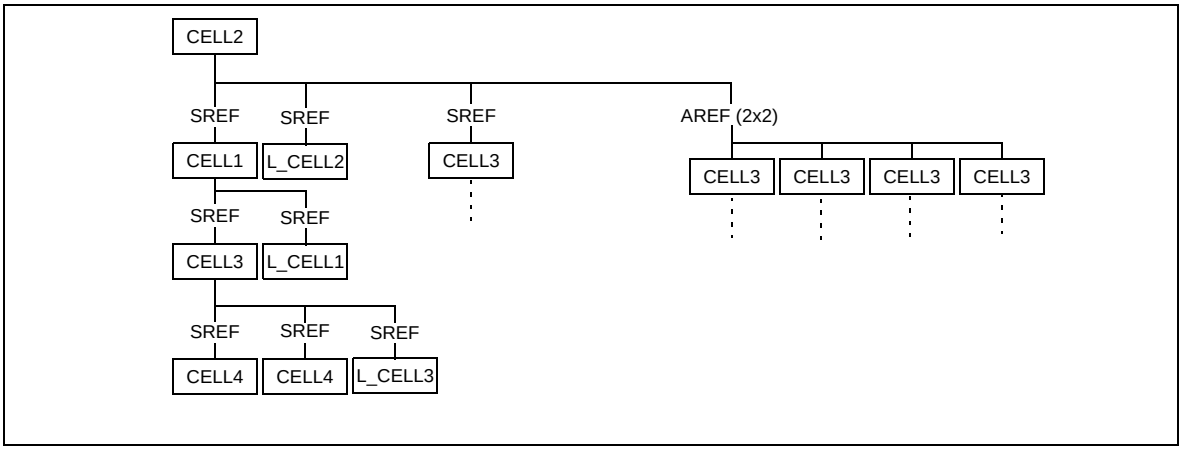


Figure 27 Layout Hierarchy after Leaf Scaffolding

In this example, the structures CELL1, CELL2, and CELL3 all contain both graphical elements and references to other structures, while CELL4 contains only graphical elements. Leaf scaffolding adds virtual cells to the hierarchy as shown in Figure 27. The new virtual cells are labeled L_CELL1, L_CELL2, and so forth. The cells are virtual because they might not be part of the output, depending on subsequent processing steps.

Graphics Scaffolding

Providing effective distributed processing during correction depends in part on balancing the size of the correction templates. In order to do this, a maximum size is enforced on cells holding only graphics information.

Graphics scaffolding slices up large graphics cells into approximately equal-sized bins no larger than `MAX_CLUSTER` in size. This is beneficial because:

- smaller cells will not overwhelm subsequent processing with too much data.
- hierarchy compression could happen in the smaller cells.

As shown in Figure 25 on page 123, only the new graphics cell L_CELL2 is large enough to be sliced up; it results in two additional cells, G1_CELL2 and

G2_CELL2, shown in Figure 28.

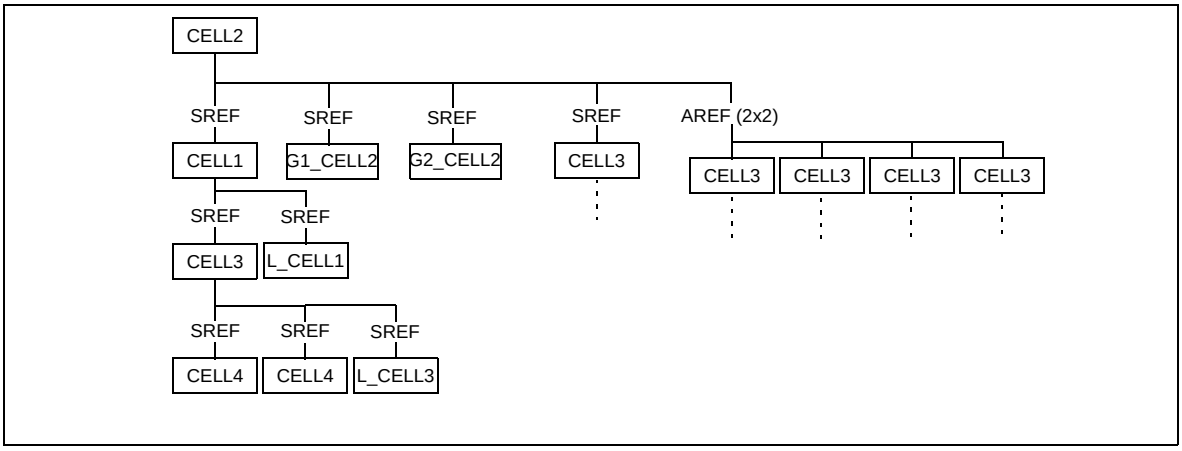


Figure 28 Layout Hierarchy after Graphics Scaffolding

Multiple threads each process distinct cells, in parallel. Because graphics scaffolding uses multiple CPUs, the reported CPU time might significantly exceed the wallclock time. You should measure performance based on "Elapsed time" rather than "User + Sys" in the log file.

AREF and SREF Scaffolding

Optimizing overall correction runtime can be driven by

- instance count: the more instances there are, the longer it takes to do context analysis.
- template size: very small templates have a large amount of context data in relation to their content data, and so are inefficient to correct.

AREF and SREF scaffolding attempts to create minimum-sized templates from groups of repeated structures. Arrays of references (AREFs) are always inspected and subject to subdivision and effective flattening, while large groups of single references (SREFs) are subdivided and effectively flattened if the number of instances in the design exceeds the job control parameter `SREF_SCAFFOLD`.

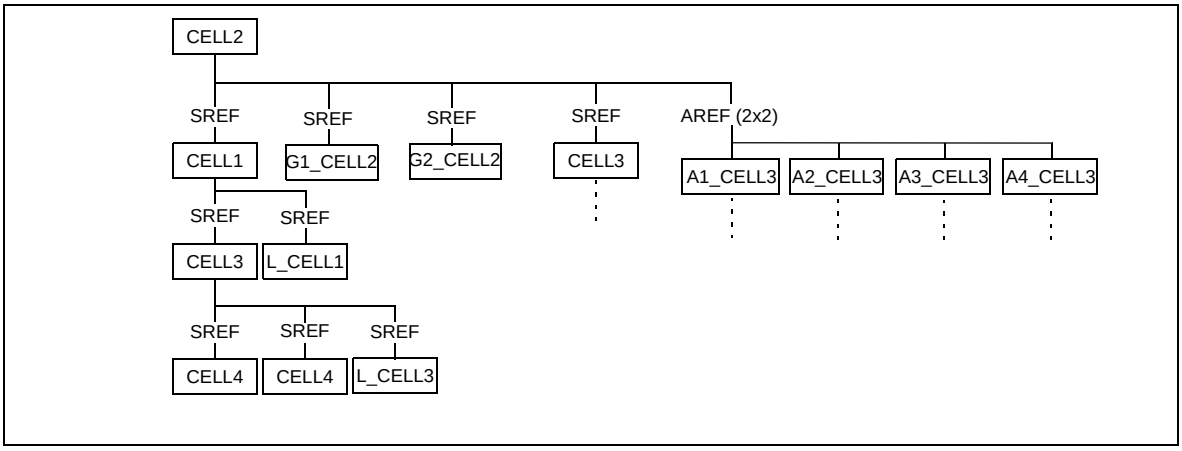


Figure 29 Layout Hierarchy after AREF Scaffolding

In the example shown in Figure 26 on page 124, the only AREF is the 2x2 array of the structure CELL3. If the physical size of an AREF is larger than `MAX_CLUSTER` in either dimension, this AREF is subdivided into approximately `SIZE_CLUSTER`-sized bins, starting in the center of the array. This allows the maximum hierarchical compression within the array, because context differences are more likely to occur on the edges of the array. AREF scaffolding uses `SIZE_CLUSTER` as its initial size estimate for the interior child AREF cells. This value is then adjusted to find an optimal match such that the entire AREF is efficiently scaffolded. The optimization goal is to minimize the exterior AREF cells.

Because the size of the array is approximately twice the `SIZE_CLUSTER`, four new cells (A1_CELL3, and so forth) are created to replace the AREF, as shown in Figure 29.

SREF scaffolding operates much like AREF scaffolding, except that bins are formed from the bottom left corner of the field of the holder cell. Child cells are included in a bin if their centerpoint falls into the bin. To force SREF scaffolding, set `SREF_SCAFFOLD` to 0. SREF scaffolding is not generally recommended due to significant loss of hierarchy compression, although it is useful if the design is very flat.

Clustering

Once the hierarchical manipulations are complete, the hierarchy manager examines the design and tries again to enforce a minimum-sized correction template. The clustering process looks for hierarchical cells whose bounding

box is smaller than `MAX_CLUSTER` and builds a template by effectively flattening the hierarchical cell. Every template built is smaller than the value of `MAX_CLUSTER`.

In our example, only CELL3 has an extent smaller than `MAX_CLUSTER`, and so the final hierarchy for context calculation is as shown in Figure 30; the instances of CELL3 are effectively flattened, thus eliminating references to the original CELL4.

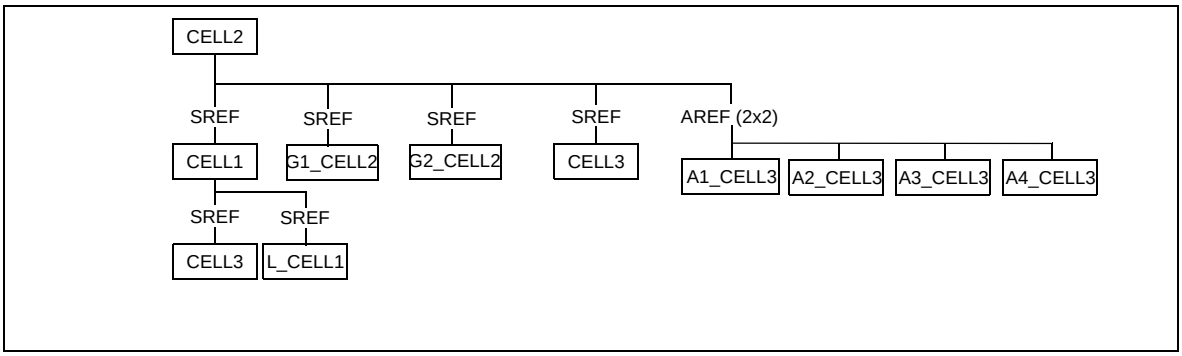


Figure 30 Layout Hierarchy after Clustering

Periodic Boundary Condition Cell Processing

Periodic boundary condition (PBC) cell processing creates arrays of highly repetitive hierarchy to optimize correction for large array processing. PBCs overcome processing limitations due to very large instance counts by:

- identifying and grouping highly repeated cells into unit cells, ensuring each cell is identical to each other cell
- forming boundary regions for orderly transitions from the array core to the periphery data
- minimizing template count and boundary mismatch for these highly repetitive cells

This optimization includes compression optimization, drawn from highly compressed memory cell placements. It also optimizes quality of results (QOR) by minimizing correction variation between different instances of the cells, and matching corrections by recognizing symmetry in repeated areas.

The Hierarchy Manager will create a PBC array from cells and instances within regions marked by a cover layer. Use the `PBC_COVER_LAYER` keyword to input a layer, or layers, of drawn shapes for use in PBC cell selection and array

creation. See PBC_COVER_LAYER on page 72.

The Hierarchy Manager will perform context analysis on all instances (PBC and non-PBC). and classify the results for each template into a type, as described in the following.

PBC Type	Description
XY	2D arrays representing the array core
X	1D horizontal array, periodic in X
Y	1D vertical array, periodic in Y
CR	Corner
NP	Not periodic, might be broken instances
None	Templates not part of the PBC region

The template type is queried using `pbc_type = info.arefPeriodicType()`. The following shows an example using `pbc_type = info.arefPeriodicType()`:

```
…
OUTLIST_XY = TEMPLATE_CALL(PBC_BLK(cor_layer), (TB_INSTANCE_NAME "XY"))
OUTLIST_X = TEMPLATE_CALL(PBC_BLK(cor_layer), (TB_INSTANCE_NAME "X"))
OUTLIST_Y = TEMPLATE_CALL(PBC_BLK(cor_layer), (TB_INSTANCE_NAME "Y"))
OUTLIST_NP = TEMPLATE_CALL(PBC_BLK(cor_layer), (TB_INSTANCE_NAME "NP"))
OUTLIST_CR = TEMPLATE_CALL(PBC_BLK(cor_layer), (TB_INSTANCE_NAME "CR"))
OUTLIST_OTHER = TEMPLATE_CALL(PBC_BLK(cor_layer), (TB_INSTANCE_NAME
"OTHER"))
…
TEMPLATE_BLOCK PBC_BLK(LAYER cor_layer)
…
pbc_type = info.arefPeriodicType()
tb_instance_name = parameter("TB_INSTANCE_NAME")
if (pbc_type == tb_instance_name):
   pbc_out = cor_layer.filter("main")
   x1,y1,x2,y2 = parameter("clipBB")
   dnt_layer = mlo.createRectangle([x1,y1], [x2,y2])
   main_ring_out = mlo.layerBoundingBox(pbc_out) -
mlo.size(mlo.layerBoundingBox(pbc_out), -10)
elif (tb_instance_name == "OTHER") and (pbc_type == None):
   other_out = cor_layer.main
…
END_TEMPLATE_BLOCK(…)
```

If the template is classified as a PBC type, the recipe can apply special processing specific to that type. For example, the current common recipe application uses four or more `TEMPLATE_CALL`s, where each `TEMPLATE_CALL` processes only one type of PBC in this order: XY, X,Y, CR, and NP. The recipe uses subsequent `TEMPLATE_CALL`s for other templates (type None). At each `TEMPLATE_CALL`, the recipe takes in the previous call's corrected results as predetermined and blends the current `TEMPLATE_CALL` results into it. This allows the recipe to correct areas of pattern in order of QOR priority.

Flat Scaffolding

For some design styles, hierarchical processing does not provide enough return on the investment in context calculation time. This can happen with

- extremely flat designs, or
- designs where there is a large gap between the smallest cells and any groups of cells.

In these cases, flat scaffolding can be used to minimize hierarchy management runtime and develop a set of similar-sized templates.

Flat scaffolding effectively flattens the entire design and applies graphics scaffolding to the result; templates are generated, starting from the lower-left corner of the design's bounding box, as approximately equal-sized bins whose dimensions are smaller than `MAX_CLUSTER`. Flat scaffolding can be enabled for all or portions of the design, depending on the value of the `CLUSTER` keyword.

Context Calculation

Once the hierarchy has been altered to make context calculation straightforward, the Hierarchy Manager determines which cell instances can be represented by the same correction template by comparing the context regions around each instance of the cell. This range of context influence, parameterized by the ambit value, is taken from

- the ambit specified in the model file, or
- the `OVERRIDE_HIER_AMBIT` job control keyword.

For a simple array with no infringing outside geometries, the expansion to correction templates can be predicted with reasonable accuracy from relative cell placements.

Consider, for example, a simple 5 x 5 array of identical 1-μm cells like the one in Figure 31. If the specified ambit is 1 μm, the Hierarchy Manager produces 9 templates: 4 corners (1 instance each), 4 unique edge templates (3 instances each), and 1 interior (9 instances).

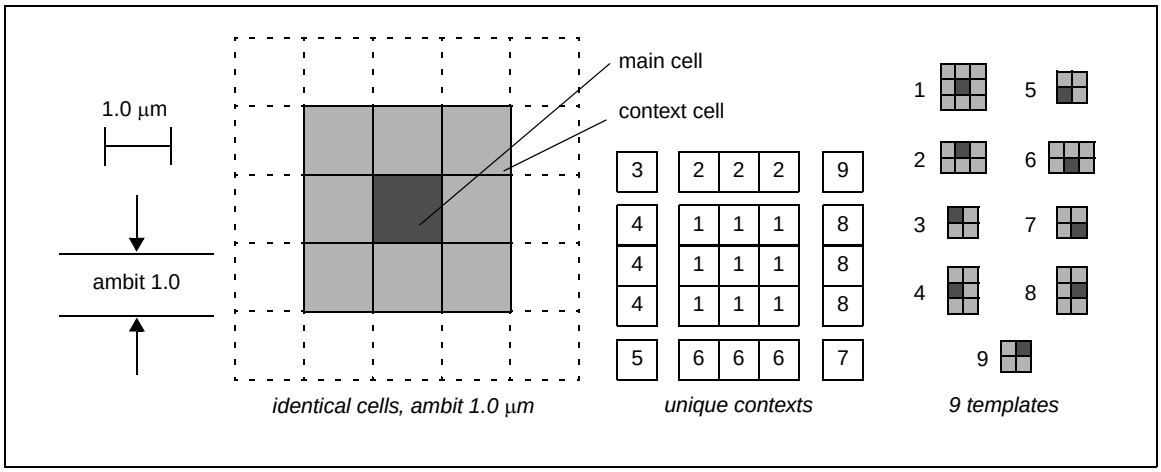


Figure 31 Correction Templates

As shown in Figure 32, increasing the size of the ambit increases the number of contexts. For instance, if the ambit increases to 1.5 µm, the number of templates increases to 25.

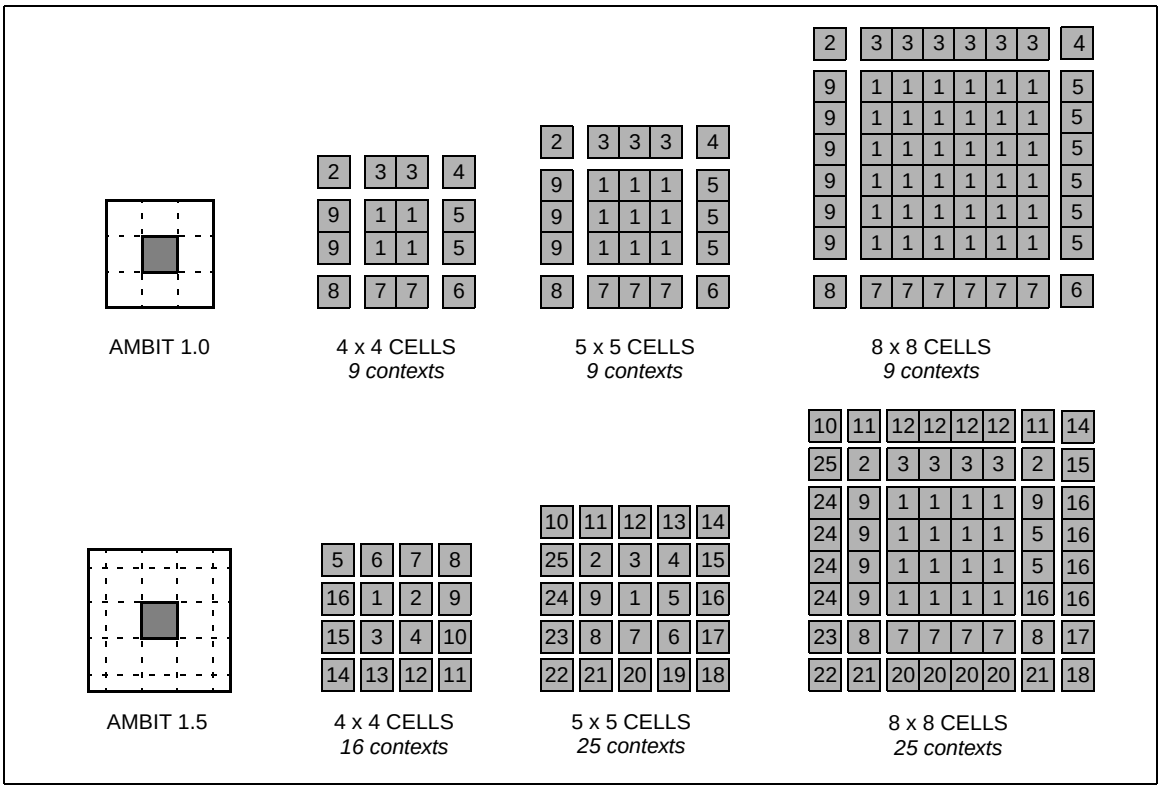


Figure 32 Predicting Unique Contexts from Relative Cell Placements

The ratio between the number of instances in a design and the resulting number of templates after context calculation is the *compression ratio*. Highly regular designs (such as memories) exhibit very high compression ratios; most templates represent many different instances of the same cell. Largely flat designs, such as those from autorouted cell-based tools, can exhibit relatively small compression ratios, as every instance of any given cell is likely to have different neighboring cells.

The hierarchy manager by default (`COMPACT_CONTEXT OFF`) considers only the placement of neighboring cells when calculating context. However, a more accurate context determination is possible by considering the geometries of neighboring cell instances. For example, add a structure that contains a single long line near the top border of a 5 x 5 array (Figure 33).

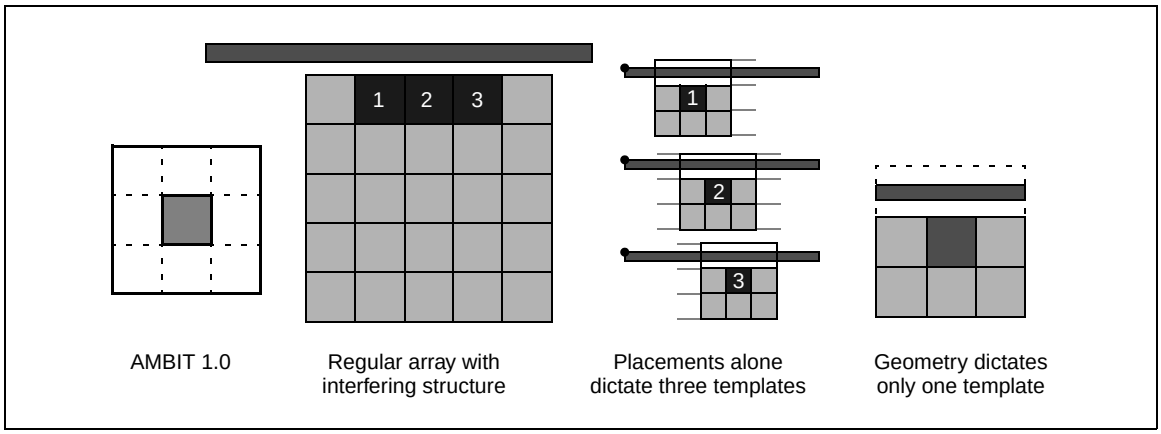


Figure 33 Influence of Geometry of Neighboring Cell Instances

Without considering the geometries in the interfering cell, the Hierarchy Manager must make each top row cell instance a unique correction template, since each cell in the top row has the neighboring cell at a different offset. This can result in some loss of compression in the design. To determine that the effect of this run on the top row is the same for every placement, the Hierarchy Manager must examine the geometry of the interfering structure.

The `CLUSTER` parameter (see CLUSTER on page 144) specifies how much effort is spent in finding context matches, in addition to defining clustering modes. The default `CLUSTER` setting looks only at the neighboring cell placements. It uses a simple bias of the bounding box of the cell graphics to define the area in which to look for other cells. More complex algorithms are used with `COMPACT_CONTEXT ON`, at the expense of runtime; using `COMPACT_CONTEXT` can require significantly more hierarchy management runtime during context analysis.

Template Generation

The final step in hierarchy management is to generate the template descriptions and write the output files. The context analysis phase determines which instances can be represented by the same template. During template generation, one arbitrary instance of those is chosen as the representative instance.

The Hierarchy Manager does not write a new layout file with the final hierarchy in it. Instead, it writes a set of files that define the hierman templates by which

cells or graphical elements from the input file make up the correction template. The files are placed in the directory specified by the `BASEPATH` job control parameter, and are prefixed with the string specified in the `JOBNAME` job control file parameter.

The Hierarchy Manager prints the area of top cell bounding boxes, the summed total area of all cluster hierman template cell bounding boxes, and the summed total area of all cluster hierman template cell ambit-biased bounding boxes. The Hierarchy Manager also prints the numbers of generated hierarchical, flat, and holder templates.

Template names (which become output cell names after correction), are uniquely named according to the following formula:

<prefix><template_number><letter>_<original_cell_name>

where:

prefix

A user-defined prefix (the `STR_PREFIX` parameter).

template_number

A unique ordinal number indicating the sequential template number. The ordinal number used in the template name is simply an index. The last template has an index one less than the number of templates created.

letter

A flag indicating which the Hierarchy Manager function generated the cell (see Table 2).

original_cell_name

As much of the original cell name as fits in the length remaining in `CELLNAME_MAX_LEN` (the default of which is 256).

When `NEW_SMART_BLOCK_COMPRESSION is ON`, parent cells created by smart block flat compression follow the naming convention:

<prefix><template_number>F_SB_cellName

These parent cell names appear in the output file as holder cells.

Flat child template names take the form:

<prefix><template_number>F_<col>_<row>_cellName

where:

col

The column number of flat bins from left to right.

row

The row number of flat bins from bottom to top.

Table 2 Template Name Flags Indicating Cell Genesis

Flag	Origin of the Cell
N	Native cell. The cell exists in the original design hierarchy.
L	The cell was created during leaf scaffolding.
G	The cell was created during graphics scaffolding.
A	The cell was created during AREF scaffolding.
S	The cell was created during SREF scaffolding.
F	The cell was created during flat scaffolding.
T	Transform cell.
SB	The cell was created by smart block flat compression.

Log Files

After hierarchy management processing, Proteus hierarchy management files are essentially a disk-based list of links back to the original input file. The Proteus tool writes data structures that are used to retrieve the data for a particular template for correction. The original GDS or OASIS is unchanged and proteus does not create any GDS or OASIS files.

Proteus log files follow the naming convention *jobname*.HMLOG, where *jobname* is the name you have given your Hierarchy Management job, controlled by the `JOBNAME` keyword, which defaults to `PROTEUS`. For log file examples, see Appendix C, Log Files.

Pattern Selection and Hierarchy Control

By default, the Hierarchy Manager treats all cells containing any of the layers defined in the `INPUT` declaration as though they contain data to be corrected. See INPUT and END_INPUT on page 54.

You can improve correction efficiency by identifying which components of the pattern must be treated as correctable data, and which components can be treated as context only.

Several optional job control statements specify which cells in the design are to be treated for creating context for other cells, and which cells in the design must be treated as though influenced by other neighboring cells. These options (`BASE_GROUP`, `BLIND`, `SUPPRESS`, `INVISIBLE`, and `ENVIRONMENT_GROUP`) use the group names included in the `INPUT` declaration, and `MARK`s. Refer to Keywords for Hierarchy Control on page 86 for more information.

Hierarchy Revision and Mark Output

The Hierarchy Manager applies marks after performing hierarchic scaffolding operations. The revised hierarchy resulting from these operations can thus contain cells that are different from those found in the original hierarchy. Rectangle mark elements are applied against this newly constructed hierarchy. The mark results are based on interactions, not only with the bounding boxes of the original cells, but also with those of the newly generated scaffold cells. Node path mark elements are applied to the original hierarchy, ignoring the effect of scaffolding operations. Additionally, the clustering operation does not permit cell instances with different marks to be clustered together. These factors should be considered in evaluating the outcome of the mark operation.

An example of marks using rectangles and node paths is as follows:

```
MARK selected
+ *
- *.myCell.*
- RECTANGLE 45900 27000 47000 28000 INTERSECT
- RECTANGLE 43000 27000 51000 28000 INTERSECT
- RECTANGLE 48000 26000 48050 26050 ENCLOSE
- RECTANGLE 46000 26900 46900 27900 INTERSECT
& *.myCell2.*
END_MARK
```

First, all instances of all cells are marked. Next, instances of cells that fall under `myCell` are removed from the mark. Then, instances of cells that interact with the four rectangles are removed from the mark, based on the `INTERSECT` or `ENCLOSE` rules as specified. Next, all instances of cells that do not fall under `myCell2` are removed from the mark.

See also MARK and END_MARK on page 94.

Owned Regions

The owned region of a template represents the area of the chip that is covered by one template and no other. You can use owned regions to enable your recipe to make decisions about placement and modifications of polygons between templates. Because the owned region is unchanging through the course of the recipe, it can provide a stable spatial reference across template calls.

An owned region consists of shapes that exactly define what area belongs to a template, and do not overlap any other template. (For `CLUSTER NONE`, the owned region is equivalent to clipBox.) Shapes may be non-rectangular; it may require multiple shapes to define the owned region of a template.

Any point on the chip has one and only one owner, which remains unchanged through the entire Proteus run, with two exceptions:

- In PBC processing, some instances may be grouped, causing some instances to be abandoned and some instances to change size. (See Periodic Boundary Condition Cell Processing on page 128.)
- If `CREATE_EMPTY_TEMPLATES` is `OFF`, when `NEW_OVERFLOW_CELL` overflows graphics to an empty space, it activates a new template to receive the overflow graphics. The next template call will have a new template and new owned region neighbors where the new template exists. (See NEW_OVERFLOW_CELL on page 167.)

The owned region layer and neighbor owned regions may be different in otherwise identical templates, due to interactions with neighboring templates. See Figure 34.

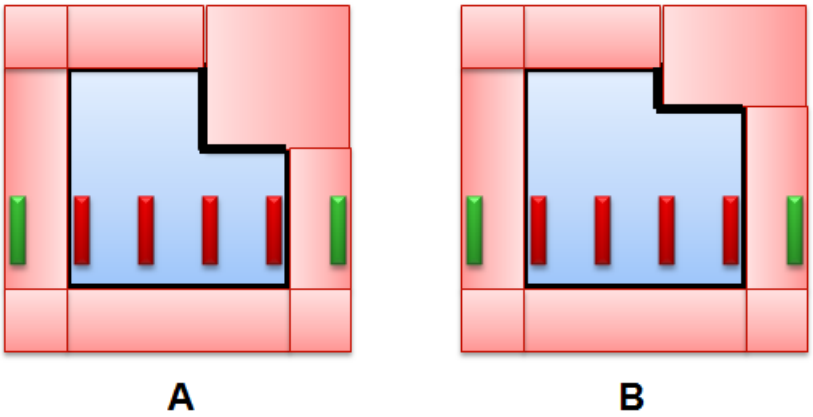


Figure 34 Owned regions with different shapes

The recipe can request the owned region as a Python layer for use in Boolean operations, MLO, LOPS, corBASIC, or output layer. Similarly, owned region neighbors can be requested as a dictionary (keyed by neighboring instance number) in the recipe. In either case, modifying the layer has no effect on the assignment of data to templates.

Figure 35 shows an owned region (the blue center) with a dictionary of eight neighbor templates (the red outer ring). The red shapes in the center are the `.main` for the current template. The green shapes in the left margin are the `.context` for the current template.

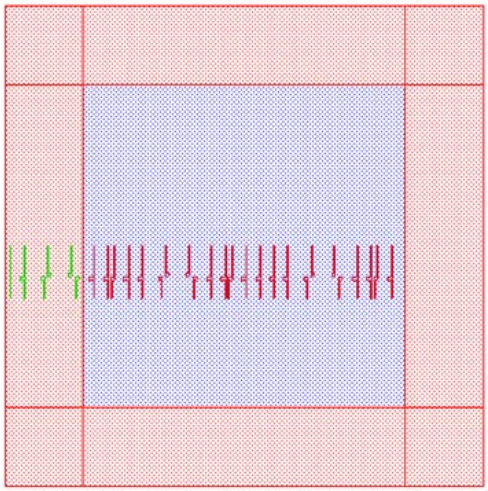


Figure 35 Owned region with eight neighboring templates

If `CREATE_EMPTY_TEMPLATES` is off in the template call, and the pattern has some empty regions of the design where no template would be created, then the parts of the templates adjacent to the empty regions will not have neighbor owned region polygons. Empty instances within `hier_ambit + MAX_OVERFLOW_GRAPHICS_EXTENSION` of source data are created, but they are not activated as templates until `NEW_OVERFLOW_CELL` needs them.

By default, owned regions will enable `CREATE_EMPTY_TEMPLATES` only for the empty instances near the source data. In other words, the empty instances described above will be activated as empty templates when owned regions is enabled (`ON`). To disable this, the recipe can define `CREATE_EMPTY_TEMPLATES OFF` either globally or as a `TEMPLATE_CALL` parameter.

Figure 36 shows seven normal templates (shown in blue) and three empty instances (not activated; shown in green). Initially, T6 is an empty (but not active) instance, so it is not seen as an owned region neighbor for its neighboring templates. If `TEMPLATE_CALL 1` creates data that overflows (via `NEW_OVERFLOW_CELL`) into T6, then T6 will be activated. On `TEMPLATE_CALL 2`, T6 (which is now active) can be seen as an owned region neighbor by its surrounding templates.

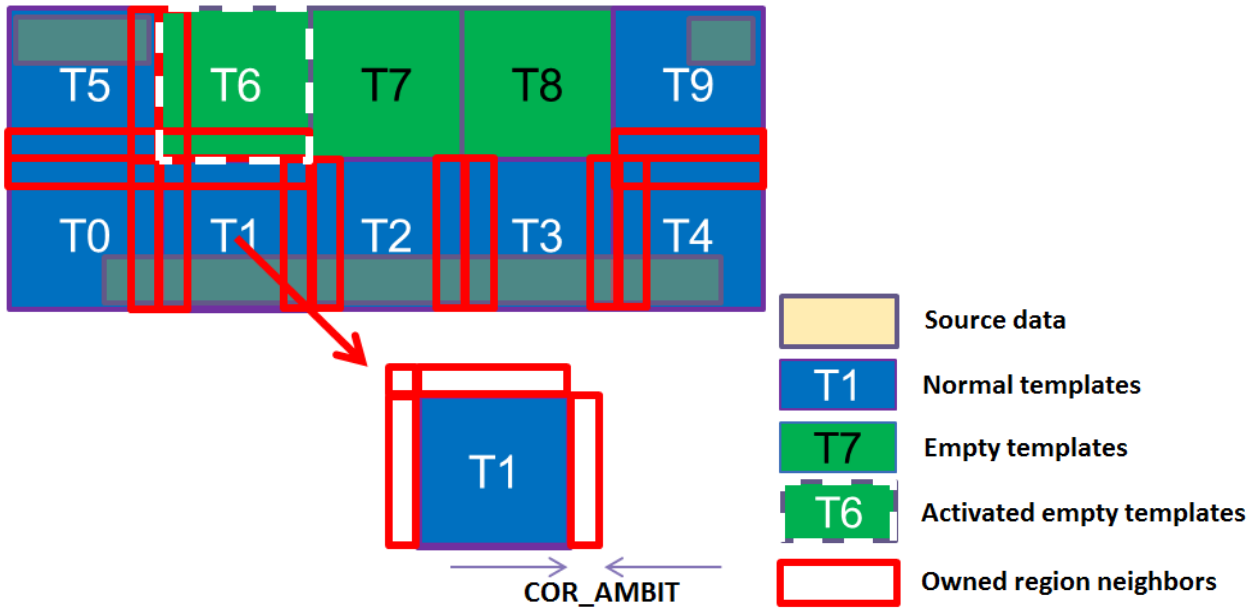


Figure 36 Owned region neighbors (for T1)

If `CREATE_EMPTY_TEMPLATES` is used in the same template call as the owned region, all templates will be surrounded with neighbor owned region polygons except at the perimeter of the chip.

When owned regions are disabled (`CREATE_OWNED_REGION OFF`), the Proteus tool looks at context within the bounding box of base_groups + `hier_ambit`. See Figure 37.

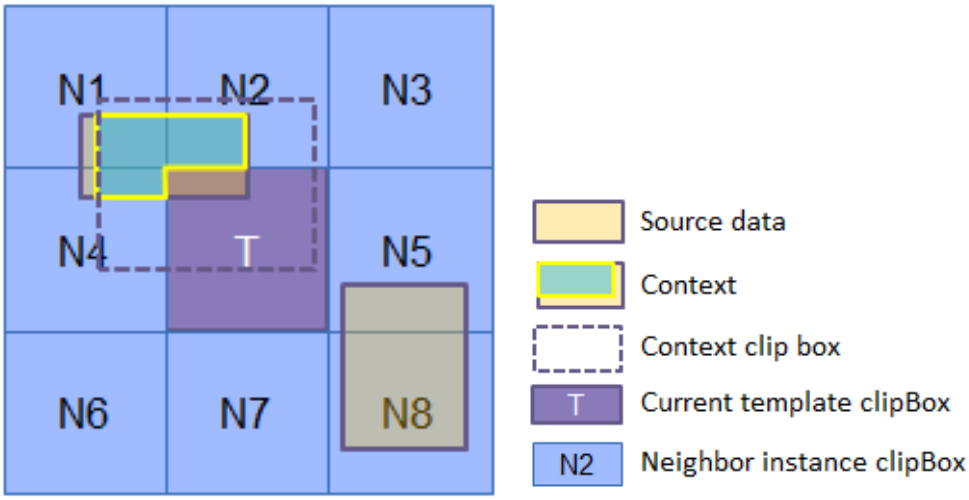


Figure 37 Context (without owned regions)

When owned regions are enabled (`CREATE_OWNED_REGION ON`), the Proteus tool looks at context within the owned region's boundbox plus + `hier_ambit`. See Figure 38.

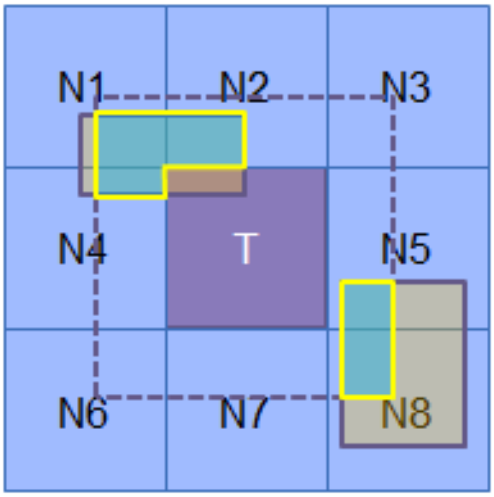


Figure 38 Context (with owned regions)

In addition, the shapes of the neighboring owned regions (within `hier_ambit`) also contribute to context. See Figure 39.

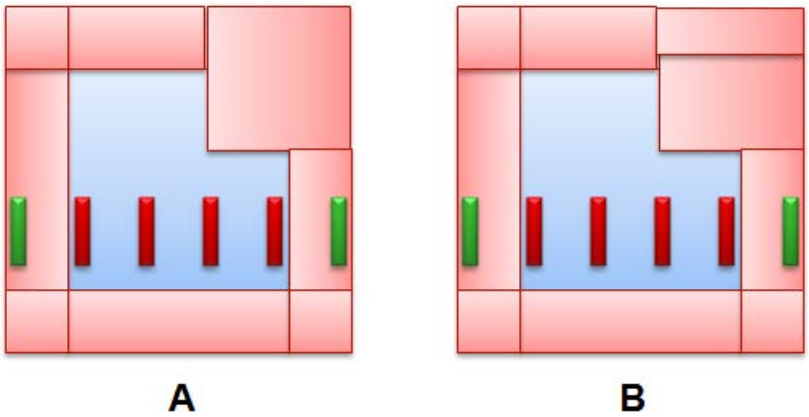


Figure 39 Neighboring owned regions with different shapes

Using owned regions may reduce compression; a larger area is contributing to the context of instances.

CREATE_OWNED_REGION

Description

When `CREATE_OWNED_REGION` is `ON`, owned region and owned region neighbor layers are available within the current `TEMPLATE_BLOCK`. This is accomplished through two Python-callable functions:

- `proteus.info.ownedRegion()` provides access to the owned region layer, which contains one or more polygons representing the owned region for the current template. The owned region shapes (within `hier_ambit`) contribute to the context of each instance, in addition to regular graphics context.
- `proteus.info.ownedRegionNeighbors()` provides access to a dictionary of polygon layers for the owned regions for each neighboring instance.

For details about these functions, see the *Python in Proteus User Guide*.

When using `CREATE_OWNED_REGION`, note the following restrictions:

- `CLUSTER FLAT` or `CLUSTER NONE` must be used so that the entire area is tiled. (`CLUSTER FLAT` is recommended.)
- Because hierarchical instances, by design, are not created as "non-overlapping", mixed-mode clustering is not allowed. (For details about mixed mode, see CLUSTER on page 144.)
- `NEW_TEMPLATE_NUMBERS` must be `ON`.
- `MARK` statements are not allowed in the recipe.
- `GRAPH_SCAFF_OVERLAP` must be 0; ragged clipping is not allowed.
- `NEW_OVERFLOW_CELL` must be `ON`; cell data must remain within clipBox.

  (Note that `NEW_OVERFLOW_CELL ON` resets the value of `RECIPE_GRAPHICS_EXTENSION` to 0.)
- `NEW_SBC_OVERLAP_ORDERING` must be `ON`.

Syntax

```
CREATE_OWNED_REGION [ON | OFF]
```

Options

`ON`

Makes the owned region layers available.

`OFF`

Does not make the owned region layers available.

See also

CLUSTER on page 144

GRAPH_SCAFF_OVERLAP on page 158

NEW_OVERFLOW_CELL on page 167

NEW_SBC_OVERLAP_ORDERING on page 168

RECIPE_GRAPHICS_EXTENSION on page 171

Clustering and Scaffolding Parameters

The Hierarchy Manager performs *clustering* and *scaffolding* to create candidate templates by merging cells together or by dividing them, rather than confine the search for correction templates solely to original cell instances.

The Hierarchy Manager commands (placed in the .xjc job control script) controlling these options are presented in the following sections.

CLUSTER

Description

`CLUSTER` specifies how cells used as candidate templates are selected.

Hierarchical clustering begins with the hierarchy in the existing layout file to determine which cells will be candidate templates before context analysis. Several hierarchical manipulations are applied to the data in an effort to generate template cells with approximately equal sizes.

Flat clustering treats the pattern as a flat expanse of polygons, using spatial bins to generate candidate template cells. Smart block compression (see NEW_SMART_BLOCK_COMPRESSION on page 169) allows for template compression among repeating flat templates. This can provide significant

advantages to all forms of flat template generation, including `CLUSTER FLAT` and mixed mode.

Mixed mode refers to having `CLUSTER FLAT` with `MARK`s followed in the next line by `CLUSTER HIER` with `MARK`s (or vice versa).

The `CLUSTER` options allow you to match appropriate hierarchy management and correction objectives to the pattern being corrected and to different regions within the pattern.

Make sure only one `CLUSTER` statement occurs in your job control file, with one of the options, `HIER`, `FLAT`, `NONE`, `AUTO`, or mixed mode.

Syntax

```
CLUSTER HIER|FLAT|NONE|AUTO
```

Options

`HIER [`*`mark1 mark2...`*`]`

`HIER` is the default mode for flash-based simulation; the layout processes hierarchically. In this mode, templates are formed based on the original hierarchy of the chip with as much compression as possible. The determination of common contexts for candidate correction template cells is based upon the shape and position of nearby graphic elements. Leaf cells in the hierarchy are clustered using the `MAX_CLUSTER` parameter. This mode works with both full hierarchy and two-level hierarchy. (See Figure 45 on page 152.)

If one or more mark arguments are defined (*`mark1 mark2`*) in the `CLUSTER` statement, the Hierarchy Manager processes the marked cells in hierarchical mode, and the remaining cells are processed flat and added to a hierarchical set of flat spatial bins (see Figure 44 on page 151).

`FLAT [`*`mark1 mark2...`*`]`

`FLAT` is the default mode for field-based simulation. In `FLAT` mode, the Hierarchy Manager divides the chip into area-based templates, looking for large repeating regions and preserving compression by creating identical flat templates in those regions. The marked cells are processed flatly (placing them in a hierarchical set of flat spatial bins) and the remaining cells hierarchically (see Figure 44 on page 151). The templates of the flattened cells are formed by grouping cells into bins approximately `MAX_CLUSTER`-sized (see Figure 42 on page 149).

Note: `CLUSTER FLAT` does not support `FORCE_TEMPLATE`.

`NONE`

In `NONE` mode, the Hierarchy Manager divides the chip into area-based regions and no compression is retained. `NONE` mode treats the pattern as a flat expanse of polygons; it is intended for fast flat processing. Hierman templates are formed by grouping graphics into bins according to the `MAX_CLUSTER` and `SIZE_CLUSTER` parameters. A two-level hierarchy is automatically imposed for `NONE` mode—any existing hierarchy is exploded into the bins. Bins might have uneven, overlapping, or interlocking edges depending on the layout of the hierarchy.

`CLUSTER NONE` does not support `MARK` statements in the recipe. If `MARK`s are used in the recipe and `CLUSTER NONE` is specified, the Hierarchy Manager ignores `MARK` statements and issues a warning. Use `CLUSTER FLAT` (without marks) to preserve any `MARK`s defined in the recipe.

Note: `CLUSTER NONE` does not support `MARK`, `ENVIRONMENT_GROUP`, or `FORCE_TEMPLATE`.

`AUTO [HIER` *`mark1 mark2...`*`] [FLAT` *`mark3 mark4...`*`]`

`AUTO` mode automatically identifies areas with a high number of repetitive cell placements (such as might be found in an SRAM area) and processes those cells hierarchically. Remaining cells are processed in flat mode and added to a hierarchical set of flat spatial bins (see Figure 44 on page 151). The `AUTO` mode is ideal in cases where it is difficult to identify deep hierarchical regions with special `MARK`s. When `NEW_SMART_BLOCK_COMPRESSION` is `OFF` (it is `ON` by default), they are added to a single set of flat spatial bins (see Figure 43 on page 150).

The `AUTO` mode can also be used in combination with user-defined `MARK`s. Two available options, `HIER` and `FLAT`, allow you to use `MARK`s to designate regions to be processed in hierarchical mode or flat mode, respectively. The user-defined regions override the autodetection results. When using the `HIER` and `FLAT` options together, user-defined flat regions override user-defined hierarchy regions if the regions conflict.

For example:

```
CLUSTER AUTO [FLAT markA markB] [HIER markC markD]
```

The cells under `markA` and `markB` are processed in `FLAT` mode, while the cells under `markC` and `markD` are processed in hierarchical mode. The unmarked cells are detected by the `AUTO` mixed-mode detection function. Any cells which have conflicting marks are processed in `FLAT` mode.

Examples

Consider the following sample design (Figure 40) showing four repeating blocks containing random logic, each with an embedded hierarchical dense array.

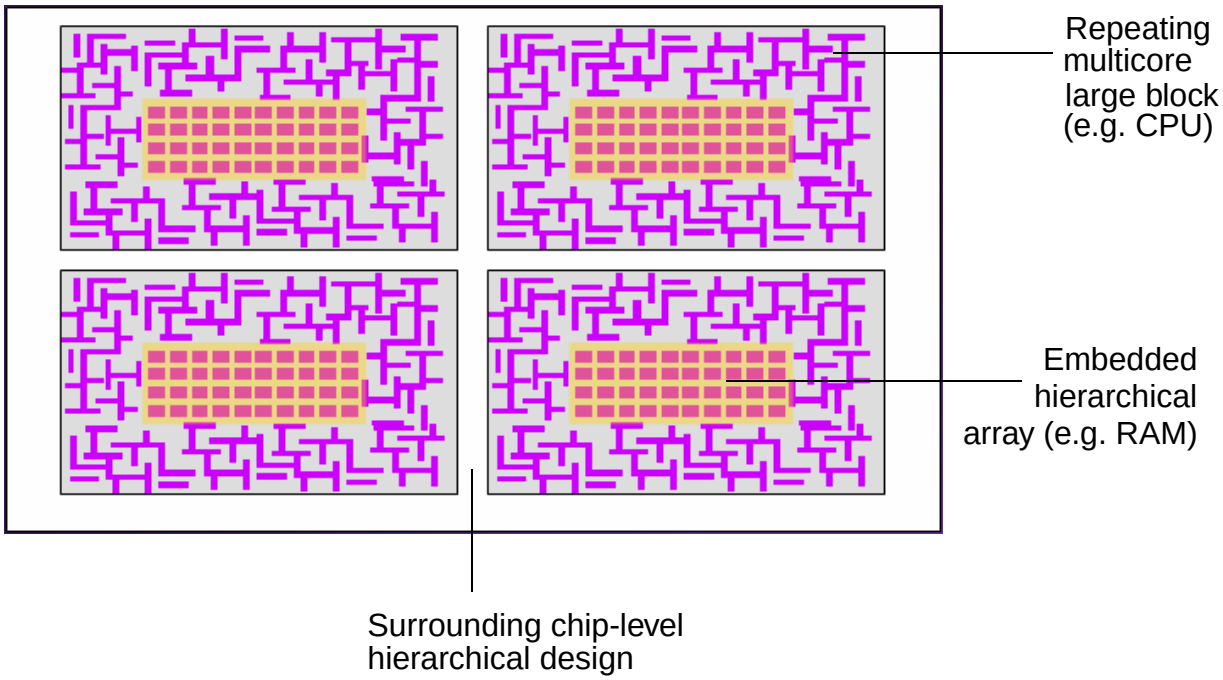


Figure 40 Sample Input: Four-Core Design with Embedded RAM

In `CLUSTER HIER` mode, the dense array benefits from the hierarchical processing, but the other surrounding pattern is better served by flat processing (especially for some metal layers) (see Figure 41).

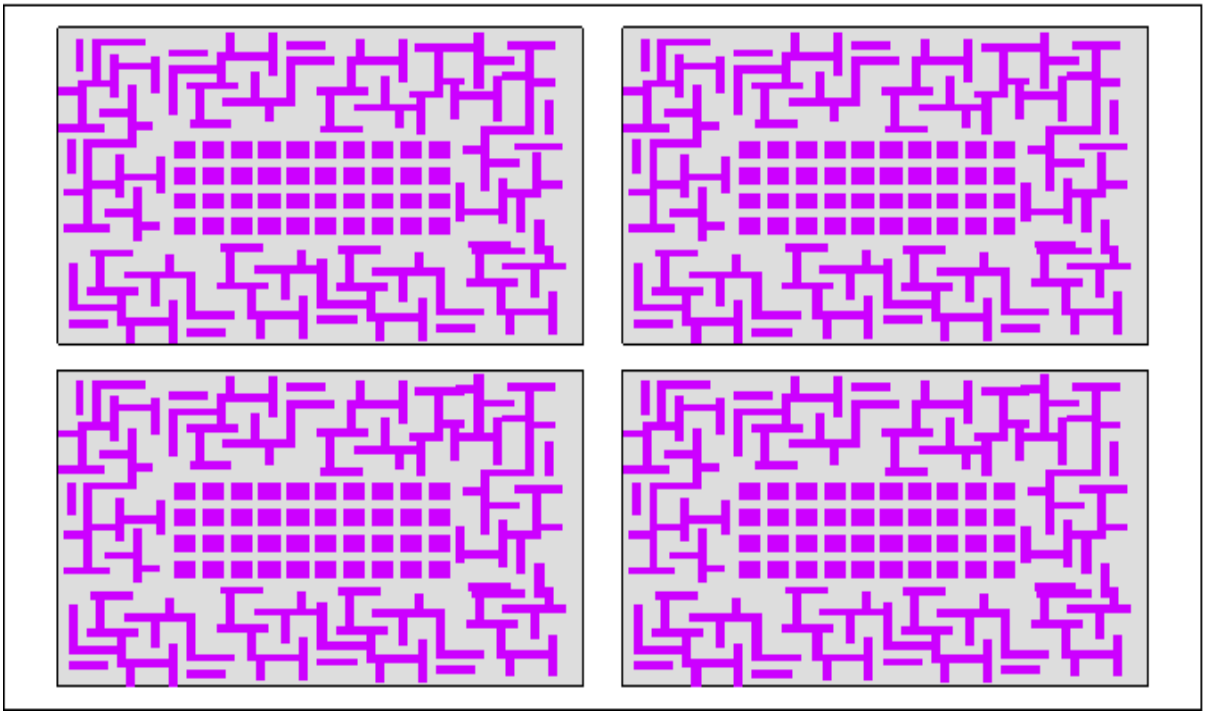


Figure 41 Correction Output with CLUSTER HIER

In `CLUSTER NONE` mode, the chip is cut up into flat templates with no detection of repeating templates (see Figure 42). Note that the bins are `MAX_CLUSTER`-sized.

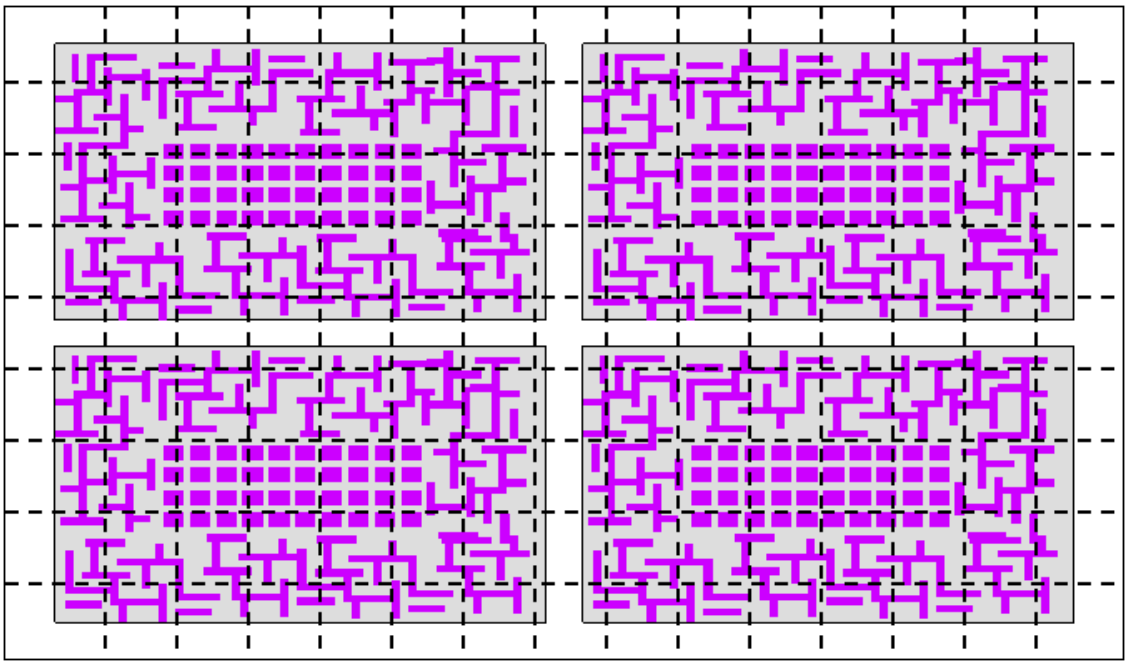


Figure 42 Correction Output with CLUSTER NONE

In `CLUSTER AUTO` mode, mixed-mode processing gives the benefit of hierarchical processing to the dense RAM arrays, but the remaining pattern is treated flat and repetition goes unrecognized (see Figure 43).

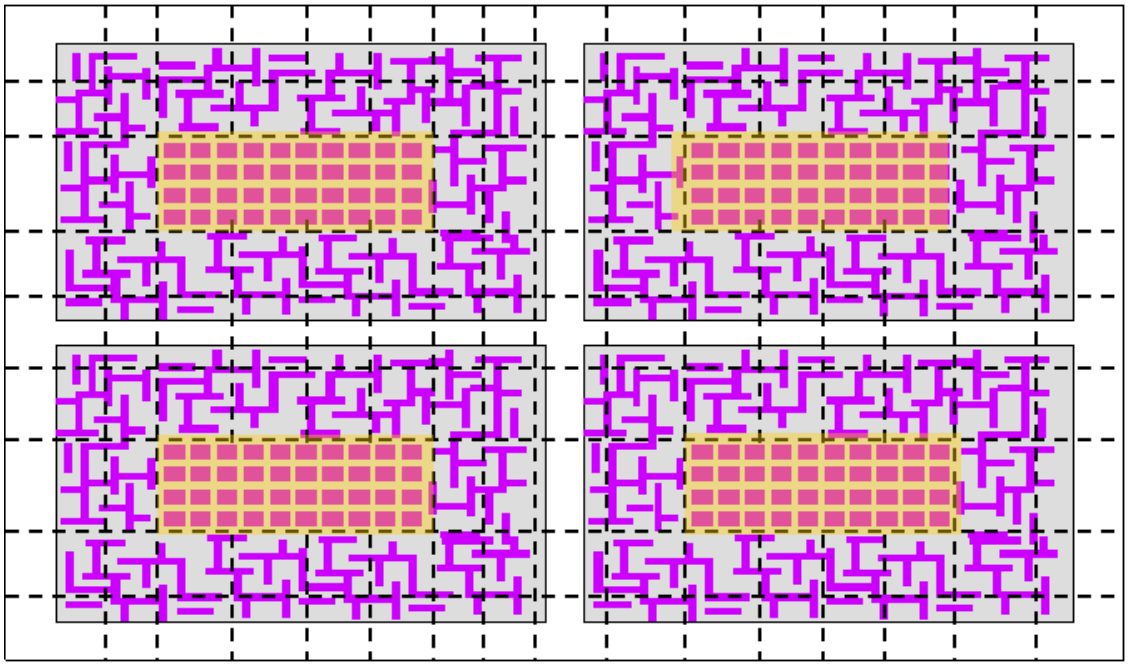
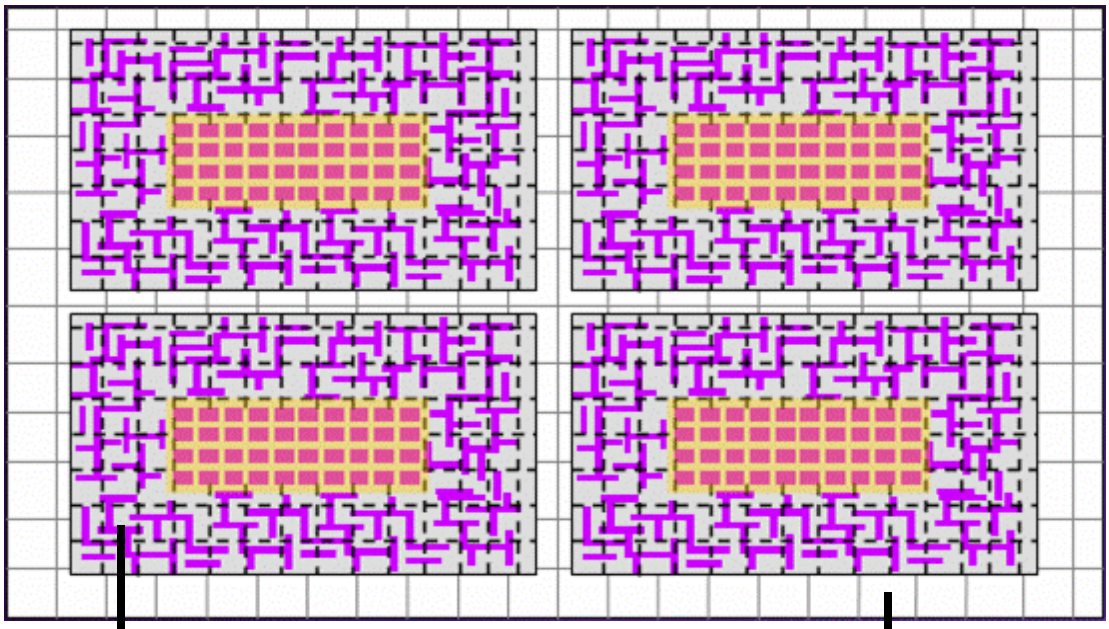


Figure 43 Correction Output with CLUSTER AUTO

When `NEW_SMART_BLOCK_COMPRESSION` is `ON` (the default), the dense arrays (for example memory regions), continue to benefit from hierarchical clustering, but at the same time repeating blocks, such as in multicore designs, are identified and processed uniformly in flat mode. Smart block compression thus results in greater template clustering and the best correction efficiency for `FLAT` regions (see Figure 44).



Repeating smart compression block (e.g. core). Each block has a set of identical flat template cells.

Single flat block region for the top-level chip logic, which lies outside the repeating cores.

Figure 44 Correction Output with NEW_SMART_BLOCK_COMPRESSION ON

See also

COMPACT_CONTEXT on page 151

NEW_SMART_BLOCK_COMPRESSION on page 169

SREF_SCAFFOLD on page 173

COMPACT_CONTEXT

Description

Note: `COMPACT_CONTEXT` is not supported with `PIPELINE_STRATEGY`, `TEMPLATE_HASH_VERIFICATION`, or periodic boundary correction (PBC).

When the `COMPACT_CONTEXT` keyword is `ON`, the Hierarchy Manager performs a more detailed examination of context during the creation of templates from cells processed in hierarchical mode. The layout, biased by `OVERRIDE_HIER_AMBIT`, determines the extent of the region used to determine context; normally, the biased bounding box of the cell is used. With `COMPACT_CONTEXT ON`, the Hierarchy Manager performs additional checks of the cell graphics to reduce the template count.

A low level of compression means the Hierarchy Manager makes little or no effort to find common contexts for correction template candidates. This can speed up the hierarchy management phase, but can also slow the correction phase by failing to identify common-context instances—identical cells that also have identical context, and which therefore only need to be corrected once.

Syntax

```
COMPACT_CONTEXT ON|OFF
```

Options

`ON`

Turns on additional checks of cell graphics.

`OFF`

Turns off additional checks of cell graphics. This is the default.

Example

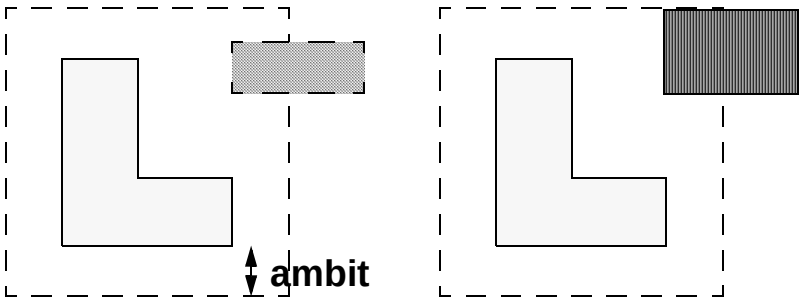


Figure 45 Typical Context Compression Case

Figure 45 shows a cell placed twice (light gray fill). Typically, the biased bounding box of the cell (dashed line) is used to determine context. In this case, the cell generates two templates because there are polygons from

different cells (medium gray fill and dark gray fill) inside the biased bounding boxes.

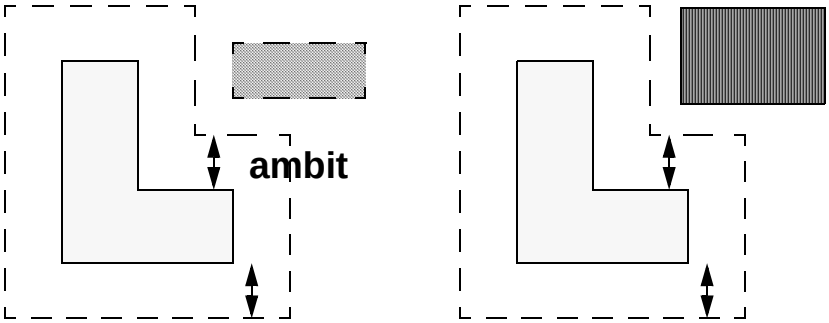


Figure 46 COMPACT_CONTEXT Compression Case

If `COMPACT_CONTEXT` is used (as in Figure 46), instead of using the bounding box, an oversize of the polygons in the cell is used for context determination. This oversize is equal to the Hierarchy Manager ambit value, `OVERRIDE_HIER_AMBIT`. In Figure 46, the two instances contain identical context polygons (none), thus only one template is generated. This illustrates how `COMPACT_CONTEXT` can reduce the number of correction templates, at the expense of additional hierarchy management runtime.

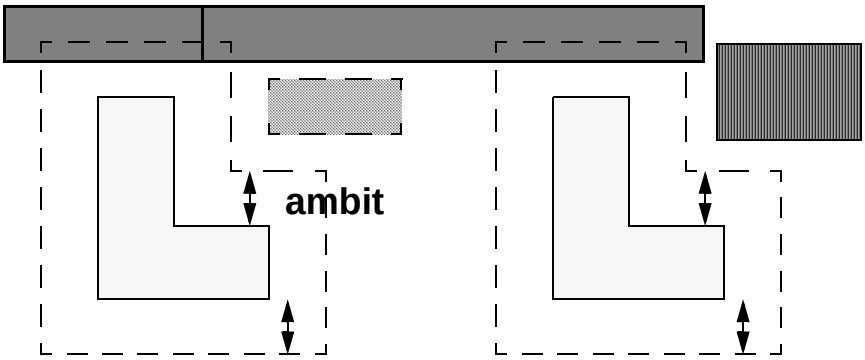


Figure 47 COMPACT_CONTEXT Flattening Case

`COMPACT_CONTEXT` also flattens polygons in the context region before doing a context comparison. In Figure 47, when `COMPACT_CONTEXT` is `OFF`, the two polygons across the top of the two cell instances are considered different context, and two templates are generated. With `COMPACT_CONTEXT ON`, the

polygons are flattened such that each instance sees the same polygons in context, and thus only a single template is generated.

See also

CLUSTER on page 144

SREF_SCAFFOLD on page 173

CONTEXT_INSTANCE_DIAMOND_GRID

Description

When the `CONTEXT_INSTANCE_DIAMOND_GRID` keyword is `ON`, the Hierarchy manager adds the additional criteria of an instance's origin on the global diamond grid for differentiating templates. Note that the cell's origin is calculated based on the Proteus tool internal DB units, 1x chip coordinate/ `DBU_PROC`.

With `CONTEXT_INSTANCE_DIAMOND_GRID`, all placements of a template are consistent with the global diamond grid, so that when the template info is queried, decisions can be made in the recipe about edge or AF placements that will be maintained when all instances are placed globally. `CONTEXT_INSTANCE_DIAMOND_GRID` enables your recipe to solve issues related to global placements of cells having 45-degree edges; for example, cases where the edges in cell coordinates are on one diamond grid, but, when placed globally, they fall onto the other diamond grid. `CONTEXT_INSTANCE_DIAMOND_GRID` allows recipe algorithms to compensate for the global placement.

Using this keyword can result in some increase in template count.

Note: If `CONTEXT_INSTANCE_DIAMOND_GRID` occurs inside of a `TEMPLATE_CALL` section, an argument is required.

Syntax

```
CONTEXT_INSTANCE_DIAMOND_GRID ON|OFF|DEFAULT
```

Options

`ON`

Turns on check for an instance's position on the global diamond grid.

`OFF`

Turns off check for an instance's position on the global diamond grid. This is the default.

`DEFAULT`

Sets the keyword value to the current default.

Example

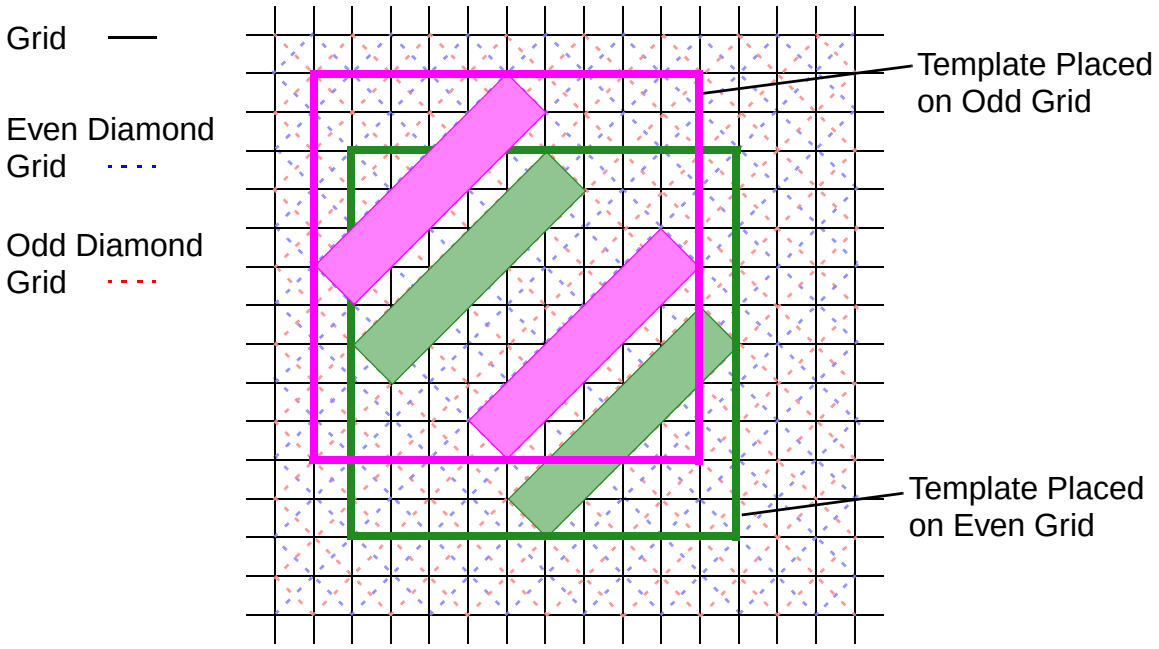


Figure 48 Global Diamond Grid and Even/Odd Template Placements

As an example of how to use `CONTEXT_INSTANCE_DIAMOND_GRID` in the recipe, the following code queries the x/y placement of the first instance of the

current template. If that location is on the global even diamond-grid, `this_diamond` will be set to `"EVEN"`, otherwise it will be set to `"ODD"`.

```
from proteus import info
t_info = info.templateInfo(2)
...
ins_id = t_info.placements.keys()[0]
 xloc = t_info.placements[ins_id]["xform"]["x"]
 yloc = t_info.placements[ins_id]["xform"]["y"]
 if xloc%2 == yloc%2:
   this_diamond = "EVEN"
 else:
   this_diamond = "ODD"
```

Subsequent processing that occurs in the template coordinate system can account for the global placement offset and place 45-degree segments selectively on the local even or odd diamond-grid. In other words, if the current template was found to have instances placed on the global even diamond-grid, then place the 45-degree segments on the local even diamond-grid. But if the current template has instances on the odd diamond-grid, place the 45-degree segments on the local odd diamond-grid. At the top-level cell, all of these 45-degree segments will reside solely on the even-diamond grid.

Note: Due to a current limitation in functionality, you must have `SCALE_IN = 1` in your recipe when using `templateInfo`.

CREATE_EMPTY_TEMPLATES

Description

When `CREATE_EMPTY_TEMPLATES` is `ON`, empty instances and templates for all regions of the chip are generated during processing.

An *empty cell* is a cell that contains no graphics on any base layers. An empty cell has a well defined graphics scaffold clip bounding box (clipBB) representing the extent of this cell.

An *empty instance* is a placement of an empty cell. An instance is placed on the chip with at a given offset, rotation, and mirroring. The clipBB is transformed by this placement and represents that actual placement of the empty cell. Empty instances are subject to `MARK`s during subsequent processing.

An *empty template* represents an empty instance to the correction engine as a template. An empty template does not contain base layer graphics. It might contain environment layer graphics. An empty template is not blind to visible context from nearby non-empty instances; it sees as context the graphics from any nearby visible instances.

To obtain the coordinates for an empty template, use the Python function `parameter("clipBB")` or `parameter("clipBB_global")`.

`CREATE_EMPTY_TEMPLATES` can be used as a `TEMPLATE_CALL` override parameter to create empty templates only for specific stages; for example, if you want only the MLO stage to generate empty templates and not the remaining stages (such as OPC).

Syntax

```
CREATE_EMPTY_TEMPLATES ON|OFF
```

Options

`ON`

Turns on creation of empty templates.

`OFF`

Turns off creation of empty templates. This is the default.

FLAT_ABSORB_HIERARCHY

Description

When `FLAT_ABSORB_HIERARCHY` is `ON`, hierarchical instances are absorbed by the flat-bin templates covering them, if any. This eliminates the need for context analysis and correction on those hierarchical instances, thereby reducing runtime.

At the boundary between a hierarchical template and a flat-bin template there can be hierarchical instances that are covered by a flat-bin template. When `FLAT_ABSORB_HIERARCHY` is `ON`, during flat bin creation the Hierarchy Manager determines whether a hierarchical instance is fully contained in a flat bin. If so, it is added to the flat-bin template and removed as a hierarchical template.

To be absorbed by a flat-bin template, the instance of the hierarchical template must have the same `MARK` as the flat-bin template. This will always be true if the hierarchical template was created using `CLUSTER AUTO`.

`FORCE_TEMPLATE` is ignored when `FLAT_ABSORB_HIERARCHY` is `ON`.

Syntax

```
FLAT_ABSORB_HIERARCHY ON|OFF
```

Options

`ON`

Turns on flat-bin template absorption of hierarchical instances. This is the default.

`OFF`

Turns off flat-bin template absorption of hierarchical instances.

See also

CLUSTER on page 144

FORCE_TEMPLATE and END_FORCE_TEMPLATE on page 92

GRAPH_SCAFF_OVERLAP

Description

`GRAPH_SCAFF_OVERLAP` controls how much a polygon extends beyond graphics scaffold boundary that will not be clipped.

Syntax

```
GRAPH_SCAFF_OVERLAP value
```

Options

`value`

The overlap value that will not be clipped during graphics scaffolding, in nanometers. By default, this value is set to `MIN_FEATURE`.

See also

MIN_FEATURE on page 194

HIERARCHY_INDEPENDENT_TILE_EQUIVALENCY

Description

When `HIERARCHY_INDEPENDENT_TILE_EQUIVALENCY` is `ON`, the tool obtains better compression by ignoring any hierarchy and structural differences in graphics while performing tile equivalency. In many cases, with this keyword

ON, you will notice fewer templates for correction. However, in some cases you might experience a significant increase in hierman runtime, depending on design hierarchy, job parameters, and the compute environment.

Syntax

`HIERARCHY_INDEPENDENT_TILE_EQUIVALENCY ON|OFF`

Options

`ON`

This setting will ignore hierarchy or other structural differences in graphics.

`OFF`

This setting takes into account hierarchy or other structural differences in graphics. This is the default.

See also

MERGE_CONTEXT_POLYGONS on page 163.

HIERARCHY_SKELETON_OUTPUT

Description

This keyword instructs the tool to generate a hierarchy hint file with the file name and location you specify. This hint file helps improve smart block compression (SBC) by providing a description of the original file's hierarchy. This feature is available only when using `CLUSTER FLAT` mode. If you do not specify *`filename`*, the tool will use `BASEPATH/JOBNAME.HSF`.

Note: When no SBC compressible hierarchy is found, a top cell with bounding box is written to the hierarchy skeleton file.

Syntax

`HIERARCHY_SKELETON_OUTPUT [`*`filename`*`]`

Options

`filename`

The name (and location) for the output hierarchy hint file.

See also

CLUSTER on page 144.

HIERARCHY_SKELETON on page 54.

NEW_SMART_BLOCK_COMPRESSION on page 169.

MAX_CLUSTER

Description

This specifies the maximum size (in nanometers) for a clustered template in either X or Y. The default value for flash-based simulation is 40000.

Note: The field-based engine determines the `MAX_CLUSTER` value slightly differently, so it is best not to set the `MAX_CLUSTER` value manually when using FBS.

The `MAX_CLUSTER` keyword can recognize one or more optional `MARK` names. `MARK` functionality can thus be used to specify various chip regions or hierarchies where an alternate `MAX_CLUSTER` value is to be applied.

`MAX_CLUSTER` can use `MARK`s with cover layers. Typically, this usage is on the RAM part of a design, with `COVER_LAYER` defined on the bit cell of the RAM. To reduce the number of rectangles, an `over_under` operation with a value of `SIZE_CLUSTER` is done automatically. See MARK and END_MARK on page 94.

The `MAX_CLUSTER` override functionality's primary application is to adjust cluster size during instance generation. For example, if a design has an embedded RAM with a deep hierarchy, you can override a `MAX_CLUSTER` with a smaller value to cause smaller cluster templates to be generated, thereby increasing overall template compression prior to correction.

The rule used to decide the template size under the mark is as follows:

1. height + width < 2 * `MAX_CLUSTER` override
2. each side >= `OVERRIDE_HIER_AMBIT`
3. If a suitable size to match 1and 2 is not found, the regular `MAX_CLUSTER` value is used to determine template size.

Note: Remember to include the `COVER_LAYER` graphics when measuring cell width and height before setting the `MAX_CLUSTER` override and `MAX_CLUSTER_OVERRIDE_MIN_SIZE`.

Note: RECTANGLE- and COVER_LAYER-based MARKs are recommended to obtain the best overall performance using the MAX_CLUSTER override. *node_path*-based MARKs run slower.

Syntax

MAX_CLUSTER *size* [[!]*mark1* [[!]*mark2*]]...]

Options

size

Specifies the maximum size (in nanometers) for a clustered template in either X or Y. This must be a positive value or an error message is generated. Additionally, *size* must be greater than height and greater than width. The default value is 2 * SIZE_CLUSTER in flash-based processing.

A warning is issued when this value is less than 500.

When using FBS models, the allowable size of the graphics, including the context region, is limited. The allowable size depends on the model and the settings of FIELD_MATRIX_SIZE, FIELD_INTERPOLATOR, FIELD_SAMPLE_SPACING, and possibly others. When running FBS models, the Hierarchy Manager internally adjusts the specified MAX_CLUSTER to accommodate for this, attempting to maximize the efficiency of the allowable template size while not exceeding the allowable size. This can be worked around by setting a MAX_CLUSTER override, so caution must be exercised when adjusting this limit for FBS models. When using flash models, there is no hard limit and MAX_CLUSTER specifies the maximum size of the template.

mark1, mark2

Must match existing mark names or an error message is generated.

!

Only affects other MARKs within the same MAX_CLUSTER statement. Thus, the following statement has no effect:

```
MAX_CLUSTER 15000 !mark2
```

... because there is no other MARK within the MAX_CLUSTER statement that mark2 can affect.

The ! operator can be specified to indicate the case where a MARK is *not* applied to an instance. For example,

```
MAX_CLUSTER 15000 mark1 !mark2
```

In this case, instances get a MAX_CLUSTER override value of 15,000 nanometers in all areas marked by mark1 with the exception of areas also marked by mark2. This is not a Boolean operation. Areas outside of mark2 and not in mark1 are not affected by the !mark2 statement. The !mark2 statement only has the effect of negating marked areas in mark1.

The following additional behaviors apply:

- Additional MAX_CLUSTER values with various mark information can be specified, for a total of up to 16 additional MAX_CLUSTER override lines.
- Multiple mark names for the same keyword line use an AND behavior on a given line. For example:

  ```
  MAX_CLUSTER 16000 markA markB !markD
  ```

  applies the MAX_CLUSTER value of 16,000 nanometers whenever markA AND markB AND NOT markD are active.
- Each specification of the MAX_CLUSTER keyword with MARK names is applied in order of declaration. In other words, as the values are being searched, the first item that matches a mark specification is used. This provides the equivalent of an OR behavior by allowing the first match to be applied.
- When using MARKs with a MAX_CLUSTER override statement, only MARKs containing "+" operators are currently supported. This operator allows multiple RECTANGLEs to be specified within a single MARK and name aggregation. If it is necessary to use the "-" operator, use a second MARK and then use the "!" operator with this MARK on the MAX_CLUSTER override statement.

See also

MARK and END_MARK on page 94

SIZE_CLUSTER on page 172

MAX_CLUSTER_OVERRIDE_MIN_SIZE

Description

This keyword sets a minimum clustered template size for a MAX_CLUSTER override to take place.

Syntax

MAX_CLUSTER_OVERRIDE_MIN_SIZE *size*

Options

size

Specifies the minimum size (in nanometers) for a MAX_CLUSTER override to take place. This must be a positive value or an error message is generated.

See also

MAX_CLUSTER on page 160

MAX_OVERFLOW_GRAPHICS_EXTENSION

Description

MAX_OVERFLOW_GRAPHICS_EXTENSION is used in conjunction with NEW_OVERFLOW_CELL to specify the measurement (in nanometers) of the pad around the periphery of the design, all smart blocks, and marked cells. Templatization occurs after the design has been padded by this amount. This results in an empty area on the border of all instances on the periphery, all smart blocks, and marked cells to receive graphics migrating from populated areas of the template.

MAX_OVERFLOW_GRAPHICS_EXTENSION is ignored if NEW_OVERFLOW_CELL is not ON.

Syntax

MAX_OVERFLOW_GRAPHICS_EXTENSION *value*

Options

value

Specifies the size (in nanometers) of the pad around the periphery of the design, all smart blocks, and marked cells. The default is the highest OVERRIDE_COR_AMBIT value in all TEMPLATE_BLOCKs in the recipe.

See also

NEW_OVERFLOW_CELL on page 167

RECIPE_GRAPHICS_EXTENSION on page 171

MERGE_CONTEXT_POLYGONS

Description

MERGE_CONTEXT_POLYGONS obtains better compression by merging adjacent graphics and removing overlaps, which helps determine equivalencies during

context analysis. If you set `TEMPLATE_HASH_VERIFICATION ON`, the tool ignores the `MERGE_CONTEXT_POLYGONS` keyword and issues a warning indicating such.

Syntax

`MERGE_CONTEXT_POLYGONS` *`ON|OFF`*

Options

`ON`

Turns on merging of graphics.

`OFF`

Turns off merging of adjacent graphics. This is the default.

See also

HIERARCHY_INDEPENDENT_TILE_EQUIVALENCY on page 158

NEW_DISABLE_TC_SNAP_SIZING

Description

When `NEW_DISABLE_TC_SNAP_SIZING` is `ON`, hierman reserves a snapping region for the first template call only; thereby removing the accumulative effect of successive CORGRID values to the `MAX_CLUSTER` size. When `OFF`, hierman reserves a snapping region for each template call, thereby growing `MAX_CLUSTER` by one CORGRID for each template call.

Syntax

`NEW_DISABLE_TC_SNAP_SIZING ON|OFF`

Options

`ON`

Reserves a snapping region for the first template call. This is the default.

`OFF`

Reserves a new snapping region for each template call.

Example

With this keyword ON, the following example shows the PJF template size summary with MAX_CLUSTER not accumulating.

```
PJF Template Size Summary:
      MAX_CLUSTER + 2(    RGE   ) = Extended Graphics + 2(COR_AMBIT) +
Snapping = Template Graphics
TB1:     48431.00 + 2(     0.00) =         48431.00 + 2(     3000) +
1.00 =          54432.00*
TB2:     48431.00 + 2(     0.00) =         48431.00 + 2(     3000) +
1.00 =          54432.00*
TB3:     48431.00 + 2(     0.00) =         48431.00 + 2(     3000) +
1.00 =          54432.00*
Notes:
(*) Template Graphics size is equal to Largest Field Graphics size
```

See also

MAX_CLUSTER on page 160

NEW_FLAT_LARGE_GRAPH_SCAFF

Note: NEW_FLAT_LARGE_GRAPH_SCAFF is available only in CLUSTER FLAT and CLUSTER NONE hierarchy modes. Other modes, including mixed-mode hierarchy, ignore this keyword.

Description

When NEW_FLAT_LARGE_GRAPH_SCAFF is ON (it is OFF by default), graphics scaffolding increases the size of the scaffolded cells, both for the minimum original cell size that graphics scaffolding divides and for the size of graphics-scaffolded cells that are produced. The size for both of these is set at 5X the size that would be used if the keyword were not present. This is possible because in CLUSTER FLAT and CLUSTER NONE the flat tiling algorithms do an additional graphics-scaffold-like ragged clip of the cell instances that fall into the flat tiles, thus providing the necessary template-size control.

This keyword can cause significant decreases in TAT for some designs where there are a very large number of moderate-sized graphics scaffold cells. However, this feature can also can interact with smart block compression (SBC) and cause some increase in TAT in some cases.

Syntax

```
NEW_FLAT_LARGE_GRAPH_SCAFF ON|OFF
```

Options

`ON`

Turns on increased size of scaffolded cells during graphics scaffolding.

`OFF`

Turns off increased size of scaffolded cells during graphics scaffolding. This is the default.

NEW_OFC_CONSOLIDATION_FILE

Note: `NEW_OFC_CONSOLIDATION_FILE` is only supported with `CLUSTER FLAT` or `CLUSTER AUTO`. It does not run with `CLUSTER NONE`.

Description

When `NEW_OFC_CONSOLIDATION_FILE` is `ON` (the default), the tool writes one or more overflow cell consolidation files after each fragment file is scanned. Overflow cell processing reads the consolidated overflow graphics from the new files, resulting in improved performance during polygon-based overflow cell processing (`CLUSTER FLAT` with SBC).

Note: Multiple overflow cell consolidation files are created when `NEW_SCAN_FRAGMENTS_TBB` is `ON` (that is, when scanning fragment files using multiple CPU/cores). The number of consolidation files is determined by `TBB_THREAD_COUNT`.

Syntax

```
NEW_OFC_CONSOLIDATION_FILE [ON | OFF]
```

Options

`ON`

Writes one or more overflow cell consolidation files after each fragment file is scanned. This is the default.

`OFF`

Does not write overflow cell consolidation files.

See also

NEW_SCAN_FRAGMENTS_TBB on page 202

TBB_THREAD_COUNT on page 218

NEW_OPTIMIZE_SBC_OVERLAP

Description

`NEW_OPTIMIZE_SBC_OVERLAP` improves smart block compression performance by removing several types of overlapping blocks, including similarly-sized overlapping blocks and single placements of an overlapping block from an alternate hierarchy. This optimization also eliminates thin templates that can occur between the edges of adjacent flat blocks.

Syntax

```
NEW_OPTIMIZE_SBC_OVERLAP ON|OFF
```

Options

`ON`

Turns on improved smart block compression performance. This is the default.

`OFF`

Turns off improved smart block compression performance.

See also

CLUSTER on page 144

NEW_SMART_BLOCK_COMPRESSION on page 169

NEW_OVERFLOW_CELL

Note: `NEW_OVERFLOW_CELL` is only supported with `CLUSTER NONE` or `CLUSTER FLAT`.

Description

Use `NEW_OVERFLOW_CELL` to maintain optimal template size. During correction, template sizes can expand due to the creation of new graphics (such as AFs) and edge movement, particularly when using multiblock recipes with multiple `TEMPLATE_CALL`s that cause cumulative expansion. When `NEW_OVERFLOW_CELL` is `ON`, graphics are partitioned between the template undergoing correction and an overflow cell. Later, during the templatization phase between `TEMPLATE_CALL`s, graphics from the overflow cells are

distributed to ideal neighboring instances in such a way as to prevent instances from growing overly large and still maintain correction performance.

Note: When a template call uses DPT (double-patterning technology) and `NEW_OVERFLOW_CELL` is `ON`, the Proteus tool automatically turns off `NEW_OVERFLOW_CELL` for that template call. Overflow processing must be off for DPT recipes to preserve consistent polygon numbers; otherwise, an error message is produced.

`NEW_OVERFLOW_CELL ON` resets the value of `RECIPE_GRAPHICS_EXTENSION` to 0.

Syntax

```
NEW_OVERFLOW_CELL ON|OFF
```

Options

`ON`

Turns on use of overflow cell.

`OFF`

Turns off use of overflow cell. This is the default.

See also

MAX_OVERFLOW_GRAPHICS_EXTENSION on page 163

RECIPE_GRAPHICS_EXTENSION on page 171

NEW_SBC_OVERLAP_ORDERING

Description

When running `NEW_SMART_BLOCK_COMPRESSION` with `NEW_SBC_OVERLAP_ORDERING ON`, the Proteus tool improves the processing of overlapping large cells ("smart blocks") with the goal of fewer overlapping instances and better compression.

Note: If `NEW_SMART_BLOCK_COMPRESSION` is `OFF`, this keyword has no effect.

Syntax

```
NEW_SBC_OVERLAP_ORDERING ON|OFF
```

Options

`ON`

Turns on this behavior when running `NEW_SMART_BLOCK_COMPRESSION`. This is the default.

`OFF`

Turns off this behavior when running `NEW_SMART_BLOCK_COMPRESSION`.

See also

NEW_SMART_BLOCK_COMPRESSION on page 169

NEW_SBC_PLACEMENT_MATH

Description

When `NEW_SBC_PLACEMENT_MATH` is `ON`, the Proteus tool uses improved algorithms for determining whether a flat block should be kept for smart block compression. This results in improved template counts and reduced TAT.

Syntax

```
NEW_SBC_PLACEMENT_MATH ON|OFF
```

Options

`ON`

Turns on improved smart block compression placement math.

`OFF`

Turns off improved smart block compression placement math. This is the default.

See also

CLUSTER on page 144

NEW_SMART_BLOCK_COMPRESSION on page 169

NEW_SMART_BLOCK_COMPRESSION

Description

During `CLUSTER` processing, this keyword controls whether or not repeating flat block cells are detected and uniformly processed during flat template generation.

Syntax

```
NEW_SMART_BLOCK_COMPRESSION ON|OFF
```

Options

`ON`

Turns on smart block compression. This is the default.

`OFF`

Turns off smart block compression.

See also

CLUSTER on page 144

OVERRIDE_HIER_AMBIT

Description

During hierarchy management, this command instructs the Hierarchy Manager to use the value of `OVERRIDE_HIER_AMBIT` for context analysis, instead of ambit from the recipe or model file. `OVERRIDE_HIER_AMBIT` itself only increases the amount of context beyond .main to be considered in context analysis.

In practice, `OVERRIDE_COR_AMBIT` should be set equal to `OVERRIDE_HIER_AMBIT`. `OVERRIDE_COR_AMBIT` determines which graphics are in the .context of the template, and the same graphics should be used for context analysis (set by `OVERRIDE_HIER_AMBIT`). During hierman template generation, only cell instances that have identical context data within `OVERRIDE_HIER_AMBIT` distance away are grouped in to a single template If the data is different, a separate template is created for each different case.

During processing, it is possible to include more or less context data than originally specified by `OVERRIDE_HIER_AMBIT`. This is done by setting `OVERRIDE_COR_AMBIT`. This value can be safely set to values less than or equal to the value of `OVERRIDE_HIER_AMBIT`. It is possible to set `OVERRIDE_COR_AMBIT` greater than the `OVERRIDE_HIER_AMBIT` setting. However, the extra data will be based on one distinct instance of the template. In most situations, this is undesirable. If unsure, please consult your Synopsys representative.

For recipes using flash models (CMDL), template.main is not changed by either `OVERRIDE_HIER_AMBIT` or `OVERRIDE_COR_AMBIT`. The size of template.main is determined in the clustering and scaffolding steps before

context analysis, and affected by `SIZE_CLUSTER`, `MAX_CLUSTER`, and other parameters.

For recipes using field models (XMDL), template.main becomes smaller as the value of `OVERRIDE_COR_AMBIT` increases. This is because the usable spatial size is fixed for these field models.

Syntax

`OVERRIDE_HIER_AMBIT` *`value`*

Options

`value`

The new ambit value (in nanometers). A setting of 1000 ensures that 1000 nanometers of extra data is included in the template.

See also

OVERRIDE_COR_AMBIT on page 236

RECIPE_GRAPHICS_EXTENSION

Description

When using field-based simulation (FBS), there is a firm outer limit on graphics size, resulting in an error message if model evaluation is performed outside the limit. The `RECIPE_GRAPHICS_EXTENSION` keyword determines the additional buffer amount (in nanometers) to use when creating templates. This is necessary to account for growth of the template boundary box during correction due to auxiliary feature creation or edge movements.

Growth phases during a multiple-`TEMPLATE_BLOCK` recipe can cause later `TEMPLATE_BLOCK`s to issue the `RECIPE_GRAPHICS_EXTENSION` limit message, even though previous templates actually exceeded the limit but did not warn. If this is suspected, it could be tested using `UPDATE_GRAPHICS` and model evaluation at the very end of a `TEMPLATE_BLOCK`.

Note: `NEW_OVERFLOW_CELL ON` resets the value of `RECIPE_GRAPHICS_EXTENSION` to 0. This is because with `NEW_OVERFLOW_CELL`, you need no longer provide for a template's expanding graphics. Instead, polygons are allowed to migrate between adjoining templates.

Syntax

`RECIPE_GRAPHICS_EXTENSION` *`value`*

Options

value

Maximum buffer amount in nanometers. The default is 50.

See also

MAX_OVERFLOW_GRAPHICS_EXTENSION on page 163

NEW_OVERFLOW_CELL on page 167

SBC_SELECTION_CRITERIA

Description

When using `NEW_SMART_BLOCK_COMPRESSION`, the new `SBC_SELECTION_CRITERIA` keyword allows you to modify the SBC cell selection algorithm.

Syntax

```
SBC_SELECTION_CRITERIA [1|2]
```

Options

1

Allows up to 50 SBC cells to be selected. This is the default.

2

A more aggressive selection algorithm will select additional SBC cells, resulting in potentially better SBC compression and reduced template counts.

SIZE_CLUSTER

Description

This specifies the nominal cluster size (in nanometers) used for partial flattening of the input pattern prior to correction. This parameter is used to optimize tradeoffs between hierarchy management time and correction processing work.

Syntax

`SIZE_CLUSTER` *size*

Options

size

The nominal cluster size. It is recommended that SIZE_CLUSTER values lie in the range of 10,000 to 50,000 nanometers. This value should never be larger than the MAX_CLUSTER value.

In flash-based processing, the default value of SIZE_CLUSTER is 20000. In FBS, the default is the value of MAX_CLUSTER, which is calculated based on properties of FBS models.

A warning is issued when this value is less than 500.

See also

MAX_CLUSTER on page 160

SKIP_NON_ORTHOGONAL_COVER_LAYER

Description

When SKIP_NON_ORTHOGONAL_COVER_LAYER is in the job control file, the Hierarchy Manager skips all nonorthogonal polygons when creating a MARK COVER_LAYER.

Syntax

SKIP_NON_ORTHOGONAL_COVER_LAYER

See also

MARK and END_MARK on page 94

SREF_SCAFFOLD

Description

This triggers SREF scaffolding when the instance count exceeds the specified positive integer (the count is interpreted as a threshold). When this statement is not present, SREF scaffolding is triggered at 20,000,000. If a count of 0 is specified, SREF scaffolding is always performed. The maximum legal value for this command is 2,147,483,647 (or 2^{31}-1).

Syntax

SREF_SCAFFOLD *count*

Options

`count`

Threshold instance count.

Input File and Global Job Control Parameters

Patterns can be selected for correction by layer number and datatype, or structure name and hierarchy path name. The following sections list keywords related to file I/O and pattern filtering.

Note: The maximum number of vertices per polygon for a GDSII input file is set to 8191. OASIS files have no limit on the maximum number of vertices.

For information on DBU_IN, DBU_PROC, and SCALE_IN, see Pattern Scaling with Fractional Grids on page 104.

ALLOW_MISSING_REFS

Description

When `ALLOW_MISSING_REFS` is present, the Hierarchy Manager allows an input file with a cell name that does not exist to be loaded. When missing references are loaded, a warning message is printed to the parent xterm window. The names of the missing cells are written to the *job*.HMLOG file.

The default behavior is a fatal error if a reference to a missing cell is present.

Syntax

```
ALLOW_MISSING_REFS
```

BASEPATH

Description

This specifies the directory in which temporary and hierarchy files are placed. The default path is current directory. The maximum string length is 1,000 characters.

When the directory is not followed by a forward slash (/) BASEPATH treats the text string that follows the last forward slash as a prefix for both temporary files and Hierarchy Manager files.

Syntax

BASEPATH *path*

Options

path

The path to the directory in which temporary and hierarchy files are placed. The path takes the following form:

```
BASEPATH ./TMP/
```

See also

TEMPPATH on page 219

CELL_COORDINATE_LIMIT

Description

The default value limit of the polygon vertex coordinate is:

$$MIN\left(67108812 \cdot \frac{\text{DBU\_PROC}}{\text{SCALE\_IN} \cdot \text{DBUNIT}}, \frac{2^{23}}{\text{SCALE\_IN} \cdot \text{DBUNIT}}\right)$$

CELL_COORDINATE_LIMIT overrides the value limit of the polygon vertex coordinate computed by the Hierarchy Manager. If set, the Hierarchy Manager triggers the processing method that handles full 32-bit coordinates or the internally calculated limit, whichever is smaller.

Syntax

CELL_COORDINATE_LIMIT *value*

Options

value

The user-defined value limit of the polygon vertex coordinate.

DESIGN_READER_CACHE_PATH

Description

`DESIGN_READER_CACHE_PATH` can be used with `NEW_DESIGN_READER_HIERMAN_SCAN` to set your preferred location for the cache created by the design reader. The specified location and the standard Proteus WorkBench cache locations are searched for an existing cache. If none exists, the cache is created according to the options described as follows.

In absence of this keyword, the cache is created in standard Proteus WorkBench cache locations.

Syntax

`DESIGN_READER_CACHE_PATH {NONE|LAZY|STRICT}` *`path`*

Options

`path`

The reader first tries to create the cache in the user-defined path, but if that fails, it tries the standard Proteus WorkBench cache path options.

`NONE`

Does not create or read a cache from disk. If you specify a *path*, it is ignored.

`LAZY`

Reads a cache if it exists in normal locations, but does not create a cache if one does not already exist. You may optionally specify a *path* to check.

`STRICT`

Only checks the user-defined *path*. Failure to create or read a cache from this path raises an error.

See also

NEW_DESIGN_READER_HIERMAN_SCAN on page 196

ERROR_WITH_MULTI_TOPCELL

Description

Typically, the Hierarchy Manager reports all top cell names. If `ERROR_WITH_MULTI_TOPCELL` is in the recipe, only the first two top cell names are reported; after that, the Hierarchy Manager prints the following error message and exits:

```
--- An error has occurred. ---

Error: Multiple Topcells found in input file.

Please specify TOPCELL_IN.
```

Syntax

`ERROR_WITH_MULTI_TOPCELL`

EXTEND_GDSII_AREF_COLROW

Description

This extends the possible number of columns and rows in an AREF from 0~32767 to 0~65535. (The default is OFF). The extension of the AREF format applies only to GDS input files, since OASIS files natively support larger AREFs.

Syntax

`EXTEND_GDSII_AREF_COLROW`

FIELD_CONVOLVER

Description

The optional `FIELD_CONVOLVER` keyword allows you to adjust the convolution routine used by the field-based modeling engine. Adjusting the value of `FIELD_CONVOLVER` causes tiny numerical differences in results and affects FBS modeling performance.

Syntax

`FIELD_CONVOLVER` *`value`*

Options

`value`

Specifies the convolver version. Can be one of the following values:

- 1.0
  The original convolver for FBS. `FIELD_CONVOLVER 1.0` exists only for backward compatibility. It should be used only when you want to reproduce exactly the signal values from previous versions.

- 2.0
  This is the default value. `FIELD_CONVOLVER 2.0` is significantly faster than 1.0 and should always be used. Results vary slightly between the two in the range of numeric noise, but both have equal accuracy.

  The speed improvement seen with this value depends on the type of mask and the type of kernels. In particular, imaginary or complex masks/kernels do not see a speed improvement.

FIELD_EXTREMA_SEARCHING_MODE

Description

The optional `FIELD_EXTREMA_SEARCHING_MODE` keyword allows you to select the algorithm used in calls to the corBASIC function `GET_LOCAL_EXTREMA`.

Syntax

`FIELD_EXTREMA_SEARCHING_MODE [`*`mode`*`]`

Options

`mode`

Optional. Specifies the algorithm to use in calls to `GET_LOCAL_EXTREMA`. Can be one of the following values:

- `ON`
  Use the enhanced extrema searching algorithm.
- `OFF`
  Use the pre-G-2012.09 extrema searching algorithm.
- `DEFAULT`
  This is the default, which behaves like the `ON` mode and uses the enhanced extrema searching algorithm.

FIELD_INTERPOLATOR

Description

The optional `FIELD_INTERPOLATOR` keyword allows you to adjust the version of the FBS interpolator. The FBS interpolator is used for discovering the values of a POINT program at locations between a field's sample points.

If you specify an invalid mode (an alphanumeric string or number outside the list that follows), the Proteus tool issues an error message and the program exits at job control parse time. The field interpolator setting is global for the entire recipe; you cannot set it at the individual `TEMPLATE_BLOCK`/ `TEMPLATE_CALL` level.

Syntax

`FIELD_INTERPOLATOR` *`value`*

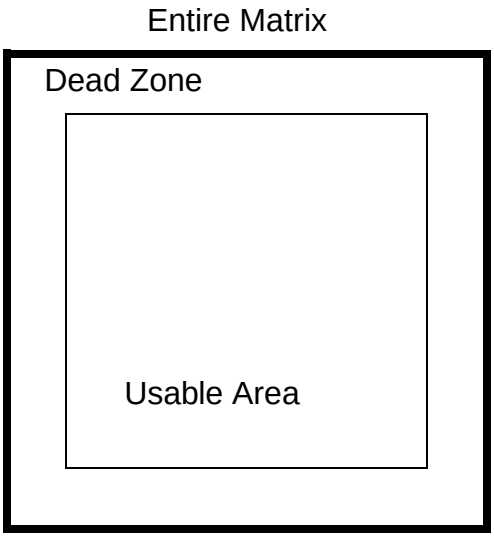
Options

`value`

Specifies the interpolator version. Can be one of the following values:

- 1.0
  The original interpolator for FBS.
- 1.1
  The 1.1 interpolator increases accuracy but creates slightly different templates than 1.0. The oversampling ratio of 2 for autogenerating the `FIELD_SAMPLE_SPACING` remains the same between interpolator versions 1.0 and 1.1.
- 2.0
  The 2.0 interpolator allows a lower oversampling of 1.45 for autogenerating the `FIELD_SAMPLE_SPACING`. This provides optimal accuracy when using automatic sample spacing. This increase in accuracy allows an increase in the automatically calculated sample spacing value (used when `FIELD_SAMPLE_SPACING` is not specified in the recipe or is set to 0). The reduced oversampling also increases FBS efficiency by allowing larger templates to be generated by the Hierarchy Manager (which might increase TAT (turnaround time) in some high-density areas).

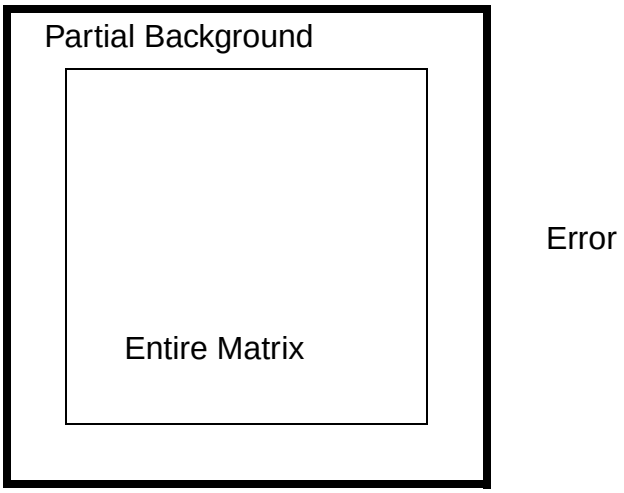
  In this mode, there is a usable area of the entire matrix minus the dead zone (which is a 6-element ring on each side where the pixels are not valid). Any evaluation made within the dead zone or outside of the field raises an error. (See Figure 49.)



Entire matrix = 25*2048 = 51200
Usable area = 51200 - 6*2*25 = 50900
(when FIELD_SAMPLE_SPACING=25, FIELD_MATRIX_SIZE=2048, Both_sides=2, Elements_ring=6)

Figure 49 FIELD_INTERPOLATOR 2.0

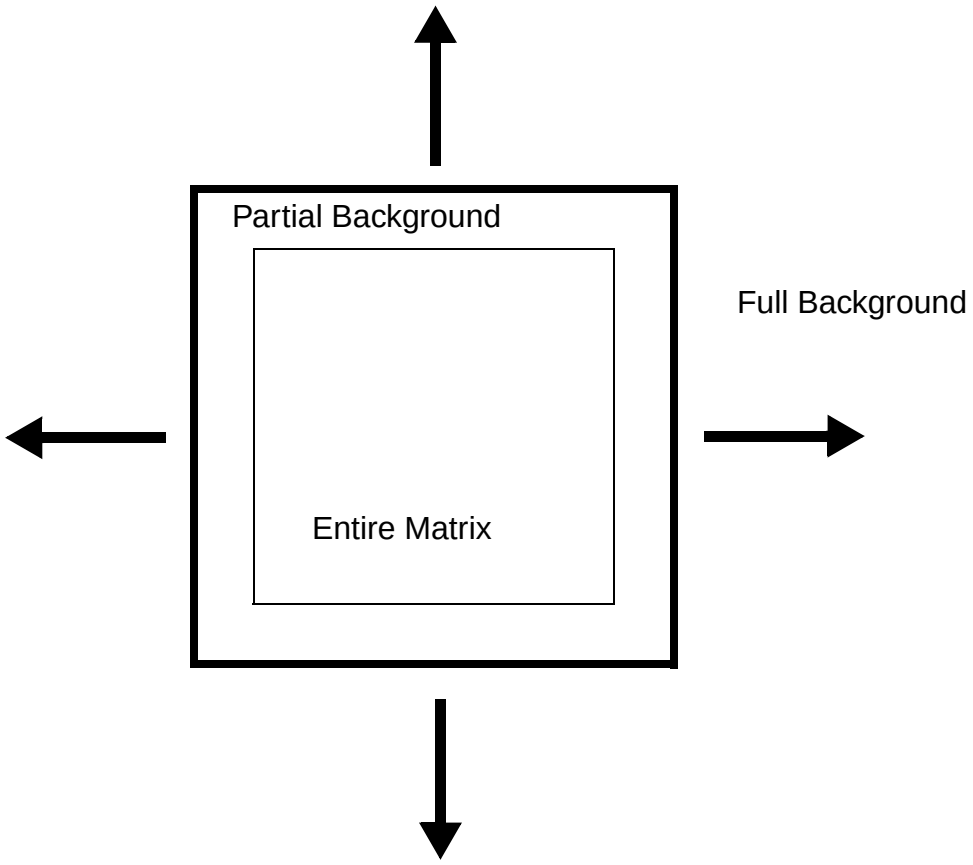
- 2.1
  This is the default value. The 2.1 interpolator removes the dead zone and increases the usable main area so that the entire field is available, allowing for some off-field evaluation. The tool generates an error if none of the samples used in the evaluation are within the field. Thus, it has a usable area of the entire matrix plus a valid element zone (11 elements on each side where any pixel is still valid). (See Figure 50.)



Entire matrix = 25*2048 = 51200
Usable area = 51200 + 11*2*25 = 51750
(when FIELD_SAMPLE_SPACING=25, FIELD_MATRIX_SIZE=2048, Both_sides=2, Element_ring=11)

Figure 50 FIELD_INTERPOLATOR 2.1

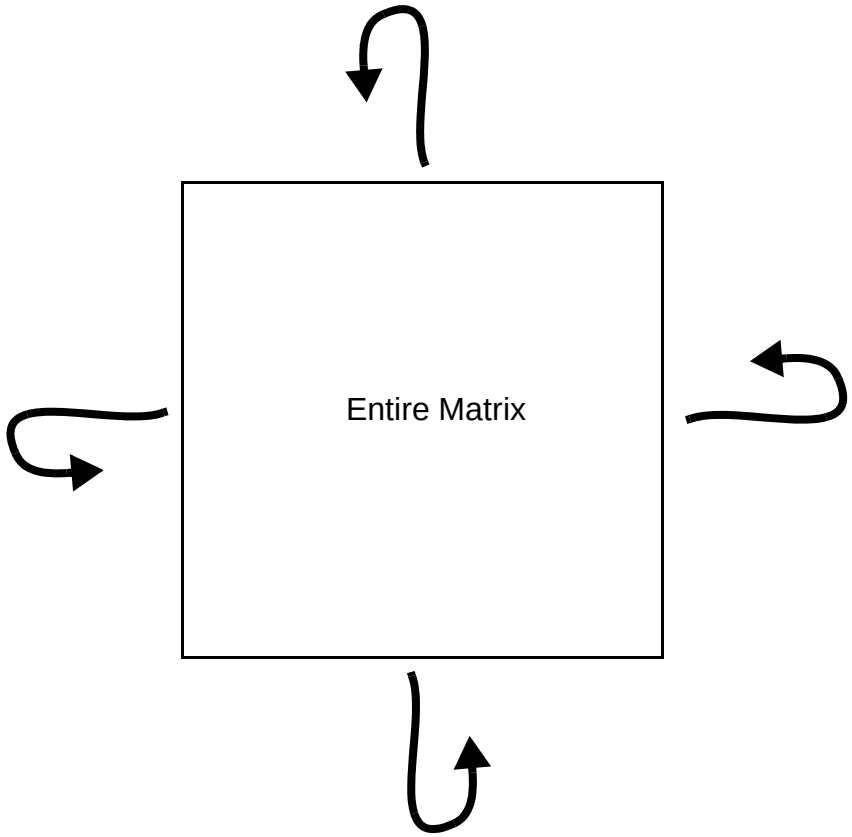
- 2.2
  The 2.2 interpolator mode is similar to mode 2.1, except that the tool never generates an error. Instead the evaluation value phases in the background value and there is an infinite usable area. (See Figure 51.)



Entire matrix = 25*2048 = 51200
Usable area = infinite
(when FIELD_SAMPLE_SPACING=25, FIELD_MATRIX_SIZE=2048, Both_sides=2, Element_ring=11)

Figure 51 FIELD_INTERPOLATOR 2.2

- 2.3
  The 2.3 interpolator mode is similar to mode 2.2, except that evaluation values are cyclical (infinitely periodically repeating) using the input pattern, and the background is not used. Thus, there is a repeating usable area equivalent to the entire matrix. The rasterizer does not change, only the interpolator changes. (See Figure 52.)



Entire matrix = 25*2048 = 51200
Usable area = 51200 (repeating)
(when FIELD_SAMPLE_SPACING=25, FIELD_MATRIX_SIZE=2048)

Figure 52 FIELD_INTERPOLATOR 2.3

See also

FIELD_SAMPLE_SPACING on page 186

FIELD_MATRIX_SIZE

Description

The optional `FIELD_MATRIX_SIZE` keyword determines the dimensions of the field to be used by the field-based modeling engine. A smaller field requires

less memory than a larger sized field, and permits the model to use more kernels within the same memory. The field matrix size can also be specified with the optional argument *field_matrix_size* to the FIELD_MODEL keyword. If both FIELD_MATRIX_SIZE and the *field_matrix_size* argument to the FIELD_MODEL keyword are specified, the latter value is used.

Syntax

FIELD_MATRIX_SIZE *value*

Options

value

Specifies the size of the field (in units of number of pixels, or sampling points). A larger field size translates into larger (and fewer) templates. A smaller field size translates into more available kernels using the same memory space. If this value is unspecified, the field engine chooses a default field dimension of 2048 x 2048 pixels (2kx2k). Can be one of the following values:

- 4kx4k specifies a 4096-pixel by 4096-pixel field.
- 2kx2k specifies a 2048-pixel by 2048-pixel field. This is the default value.
- 1kx1k specifies a 1024-pixel by 1024-pixel field.
- 512x512 specifies a 512-pixel by 512-pixel field.

See also

FIELD_MODEL on page 184

FIELD_MODEL

Description

The FIELD_MODEL keyword creates an instance of the corBASIC type model_name and associates it with the model found at *xmdl_path*. It inspects the referenced .xmdl file for wellformedness and creates a table containing all the POINT programs in the XMDL model.

Syntax

FIELD_MODEL *model_name xmdl_path* [*field_matrix_size*] [*field_pitch*]

Options

`model_name`

The corBASIC model_name type. Implemented as a constant. Can be used within the recipe to evaluate POINT programs contained within the XMDL model.

`xmdl_path`

The path to the model. The Proteus tool executables search for the model file in the following locations:

- The absolute or relative path specified in the `FIELD_MODEL` statement.
- The base name of the specified path, within the current working directory.
- The specified path, except with the `PROTEUS_BASIS_PREFIX` environment variable prepended.

`field_matrix_size`

Optional. Specifies the size of the field (in units of number of pixels, or sampling points). A larger field size translates into larger (and fewer) templates. A smaller field size translates into more available kernels using the same memory space. If this value is unspecified, the field engine chooses a default field dimension of 2048 x 2048 pixels (2kx2k). Can be one of the following values:

- 4kx4k specifies a 4096-pixel by 4096-pixel field.
- 2kx2k specifies a 2048-pixel by 2048-pixel field. This is the default value.
- 1kx1k specifies a 1024-pixel by 1024-pixel field.
- 512x512 specifies a 512-pixel by 512-pixel field.

`field_pitch`

Optional. The distance in nanometers between points in a field. If this value is unspecified and the `FIELD_SAMPLE_SPACING` keyword is not being used in the recipe, the field engine automatically sets the optimal sample spacing based on the model. If this value is specified, *`field_matrix_size`* must also be specified.

FIELD_SAMPLE_SPACING

Description

This specifies the spacing (in nanometers) between sampling points in a field. The field sampling pitch can also be specified with the optional argument *field_pitch* to the FIELD_MODEL keyword. If both FIELD_SAMPLE_SPACING and the *field_pitch* argument to the FIELD_MODEL keyword are specified, the latter value is used.

The optimal value to use for FIELD_SAMPLE_SPACING is physically related to the Nyquist pitch of the XMDL model. The Nyquist pitch can be calculated from the maximum spatial frequency (MSF) using the following equation:

$$Nyquist = 1000nm \cdot (pi)/(MSF)$$

The field engine will provide good results with FIELD_INTERPOLATOR 1.0 and FIELD_INTERPOLATOR 1.1 at an oversampling factor of 2.0 (FIELD_SAMPLE_SPACING equal to Nyquist/2.0). The oversampling factor is reduced to 1.45 when using FIELD_INTERPOLATOR 2.0. Automatic FIELD_SAMPLE_SPACING should be used in all cases to use these optimal oversampling factors.

Note: The MSF is an attribute of the model, and is typically output in the .xmdl model script by ProGen, along with the other model parameters like sigma and NA. If the MSF is not in the .xmdl file, the formula for calculating it is in the *ProGen Template Programming Guide*.

Syntax

FIELD_SAMPLE_SPACING *value*

Options

value

The distance in nanometers between points in a field. If this value is unspecified or set to 0, the field engine chooses the optimal spacing value for that particular model and FIELD_INTERPOLATOR version.

The maximum value allowed varies depending on the basis files being used. A warning is issued if you manually set FIELD_SAMPLE_SPACING to too large a value.

See also

FIELD_MODEL on page 184

FIELD_INTERPOLATOR on page 178

FILTER_PYTHON_WARNING

Description

This keyword allows warnings to be filtered and specific actions to be taken according to the warning message, category, and module. Empty arguments match all values; trailing empty arguments can be omitted.

The keyword can be specified multiple times and the filters are accumulated. Each additional occurrence is prepended to the warning filter list, such that the last matching occurrence takes precedence over previous occurrences.

`FILTER_PYTHON_WARNING` can also be provided on the command line when invoking proteus.

The `FILTER_PYTHON_WARNING` keyword interacts in the following ways with `SUPPRESS_SYMMETRY_WARNING`:

- If `SUPPRESS_SYMMETRY_WARNING ON` is in the `TEMPLATE_CALL`, it results in the suppression of all symmetry warnings, regardless of the global setting of `FILTER_PYTHON_WARNING`.
- If `SUPPRESS_SYMMETRY_WARNING OFF` is in the `TEMPLATE_CALL`, it causes proteus to ignore the global setting of `FILTER_PYTHON_WARNING`. It also results in proteus behaving as if `FILTER_PYTHON_WARNING default::proteus.SymmetryWarning` was set globally. (This usually means that one warning is issued for each unique occurrence of a symmetry warning).
- If `SUPPRESS_SYMMETRY_WARNING` is not set in the `TEMPLATE_CALL`, the behavior is as before. In other words, `SUPPRESS_SYMMETRY_WARNING` is `OFF` by default and has no effect on `FILTER_PYTHON_WARNING`, unless `SUPPRESS_SYMMETRY_WARNING` is specifically set to `ON` at the global level, in which case it turns off symmetry warnings in `TEMPLATE_CALL`s that do not have an explicit symmetry option set.

Syntax

```
FILTER_PYTHON_WARNING
   action[:message[:category[:module]]]
```

Options

`action`

An action must be specified for each warning. The *`action`* is applied to each warning that matches the remaining arguments.

The possible values for *`action`* are:

- `ignore`: Causes the associated warning category to be suppressed, and no messages written to the Proteus log files, to STDERR, or to STDOUT.
- `default`: Causes the warning messages for associated warning category to be written to Proteus log files and to STDOUT once per location per template.

`message`

Optional. The *`message`* argument is a string containing a regular expression that matches the start of the warning message to be filtered; this match is case-insensitive.

`category`

Optional. The *`category`* argument matches the warning category to be filtered. This must be a Proteus Warning class name; the match tests whether the actual warning category of the message is a subclass of the specified warning category. The full class name must be given. If you do not specify a category, the specified action is applied to all warning categories.

The *`category`* filter supports these Proteus tool categories:

- `proteus.DeprecationWarning`
- `proteus.PendingDeprecationWarning`
- `proteus.SymmetryWarning`
- `proteus.ArgumentWarning`
- `proteus.ArgumentDeprecationWarning`
- `proteus.CommonStyleWarning`
- `proteus.LayerWarning`

The category implicitly refers to warnings from the proteus.exceptions module. For example, to filter the tool deprecation warning, specify:

```
FILTER_PYTHON_WARNING ignore::proteus.DeprecationWarning
```

The double colon is necessary in this example because the optional *`message`* argument is omitted.

You can also specify any standard warnings class. For example:

```
FILTER_PYTHON_WARNING  ignore::SyntaxWarning
```

`module`

Optional. The *`module`* argument matches the (fully-qualified) module name; this match is case-sensitive. This allows you to control warnings within a large category according to the module where they are generated.

The module must match the name of a Proteus module. If not specified, the associated action is applied to all warnings within the specified category.

For example:

```
FILTER_PYTHON_WARNING
default::proteus.PendingDeprecationWarning:proteus.mlo
```

Note the double colon, necessary because the optional *`message`* argument is omitted in this example.

INPUT_FILE

Description

This specifies the UNIX path (full or relative) to the file containing the OASIS or GDSII layout to be corrected. The maximum string length of the file specification is 2000 characters. The input file must be accessible during both hierarchy management and correction processing.

This is optional for a `PROTEUS_JOB_FLOW` recipe. If both `INPUT_FILE` and `INPUT` are specified in a `PROTEUS_JOB_FLOW` recipe, `INPUT` is used.

Note: The tool checks the input format of the file or files and if the actual format is different from the format specified in the recipe, the tool uses the actual format and issues a warning message.

Syntax

`INPUT_FILE` *`input_filename`*

Options

`input_filename`

The UNIX path (full or relative) to the file containing the OASIS or GDSII layout to be corrected. The input file can also be a zip file.

INPUT_FORMAT

Description

If using the `INPUT_FILE` keyword, `INPUT_FORMAT` specifies the type of input file that is present. If this keyword is omitted, the format of the input file is detected automatically, using the initial bytes in the file. If `INPUT_FORMAT` is present, omission of the input format argument is an error.

If `INPUT_FORMAT` is present, and the `INPUT_FILE` type does not match the type specified by `INPUT_FORMAT`, an error is issued.

This is optional for a `PROTEUS_JOB_FLOW` recipe. If both `INPUT_FORMAT` and format in the `INPUT` declaration are specified in a `PROTEUS_JOB_FLOW` recipe, the format declared in `INPUT` is used.

Syntax

```
INPUT_FORMAT GDSII|OASIS
```

Options

`GDSII`

Indicates a GDSII input file.

`OASIS`

Indicates an OASIS input file.

INPUT_GDS_VALIDATION

Description

The `INPUT_GDS_VALIDATION` keyword checks your GDSII input file for illegal syntax. Results of the check are written to the *job*.HMLOG file. This feature is not available for OASIS input.

`INPUT_GDS_VALIDATION` checks to make sure that:

- The record type and datatype are consistent with the GDSII standard.
- Record length is consistent with the various record types. (The record length, record type and datatype comprise the first four bytes of a GDS record).
- `BGNSTR` is followed by an `ENDSTR` before another `BGNSTR` occurs.
- An occurrence of `BOUNDARY`, `PATH`, `SREF`, `AREF`, `TEXT`, `NODE`, or `BOX` is followed by an `ENDEL` before one of these records is repeated.
- XY records that are associated with a `BOUNDARY` have identical first and last vertices.

Errors resulting from `INPUT_GDS_VALIDATION` are fatal and cause the Hierarchy Manager to terminate (after scanning and reporting on the entire file). Warnings are written to the *job*.HMLOG file, but do not halt hierarchy processing.

Syntax

```
INPUT_GDS_VALIDATION 0|1
```

Options

`0`

No checking is done. This is equivalent to not including the keyword in the recipe.

`1`

Initiates GDSII syntax checking.

JOBNAME

Description

This specifies a name for the current job. `JOBNAME` is used to construct the base portion of all Hierarchy Manager temporary file names. The correction processor assigns a standard extension to each file name. The maximum length is 34 characters. The name should be restricted to alphanumeric characters and the underscore(_).

Syntax

`JOBNAME` *`jobname`*

Options

jobname

The user-defined name of the current job. The default is PROTEUS.

LOCAL_PYTHON_PATH

Description

This keyword allows you to cache Python modules and packages on a local disk partition to reduce file system load and improve data locality in Python. To specify the local path:

```
LOCAL_PYTHON_PATH /tmp/my_job_name
```

The Python files are cached in /tmp/*my_job_name** on the local disk partition.

Alternatively, you can use the shared memory partition to cache the Python files in RAM:

```
LOCAL_PYTHON_PATH /dev/shm/my_job_name
```

The new cache is available in dpserver only, and not in other Proteus tools such as corexec or rdebug. All dpservers on the same host share a single cache, which is removed after all dpservers exit.

Note: If you have reason to revert the new cache, set REVERT_PYTHON_MODULE_INFO_CACHE ON.

Syntax

LOCAL_PYTHON_PATH *path_name*

Options

path_name

The partition where you want to cache Python modules and packages.

LOG_VERBOSITY

Description

This determines the verbosity level of messages sent to the logfile (*job*.HMLOG).

Syntax

`LOG_VERBOSITY` *n*

Options

n

A number from 0 to 3 indicating the verbosity level of messages. 0 indicates nearly silent messages, 3 indicates the most verbose. The default is 0.

MASK3D_IGNORE_OVERLAPPING_POLYGON_EDGES

Description

Use the `MASK3D_IGNORE_OVERLAPPING_POLYGON_EDGES` keyword to avoid the hole-without-parents error during the Mask3D ripple paint step.

Mask3D ripple paint simulation has a strict requirement that the incoming shapes should not have any point touches or overlaps. In the event of multi-layer models, this requirement holds true even across the multiple layers, meaning the geometries on main feature and assist feature layers should again not have any point touches, abutting edges, or overlaps. Failure to satisfy the above requirement results in a fatal hole-without-parents error which halts the simulation.

For example, during Proteus ILT level set optimization, the mask evolves freely with aggressive topography changes, which can result in mask contours violating the above constraint. Another possibility where this can happen is during the mask contour simplification step, which is sometimes used to reduce the number of vertices and speed the Mask3D ripple paint operation. In such cases, the `MASK3D_IGNORE_OVERLAPPING_POLYGON_EDGES` keyword allows the simulation to proceed (instead of erroring out) by ignoring the troublesome edges during the Mask3D ripple paint step.

Syntax

`MASK3D_IGNORE_OVERLAPPING_POLYGON_EDGES [ON|OFF]`

Options

`ON`

During Mask3D ripple paint simulation, ignore any point touches or overlaps and proceed with simulation.

`OFF`

During Mask3D ripple paint simulation, when geometries on main feature and assist feature layers have any point touches, abutting edges, or overlaps, terminate simulation with a fatal hole-without-parents error. This is the default.

MIN_FEATURE

Description

This specifies the minimum feature size at a 1x scale in nanometers. The acceptable range is 1 to 1,000 nanometers, with a default of 200 nanometers. This parameter provides the Proteus tool with an estimated width of the correction features, If you specify a value that is outside a reasonable range, a warning/error message appears, notifying you of a possible error. (Values less than 1 result in an error.)

The `MIN_FEATURE` value specified in the job control file is truncated to an integer.

Syntax

`MIN_FEATURE` *`dimension`*

Options

`dimension`

The minimum feature size, in nanometers. Must be an integer. Non-integer values are truncated to an integer before being used by the tool. The default is 200.

See also

DBU_PROC on page 229

MMAP_LENGTH_LIMIT

Description

This specifies the limit in bytes of file size that the Proteus tool attempts to memory-map during hierarchy management. This parameter is ignored by the correction executables corexec and dpserver.

The limit of this keyword is (2^{63})-1 bytes. Inputs between 2^{53} and (2^{63})-1 bytes are supported, but are rounded to fit into a double-precision floating point value.

Files larger than the `MMAP_LENGTH_LIMIT`, or larger than the available address space, are not memory-mapped and are instead accessed with buffered I/O. On 32-bit machines (Linux IA32), address space is limited to approximately 3.8GB.

Note: In corexec and hierman, user-defined values for `MMAP_LENGTH_LIMIT` are not used, but are reported by `REPORT_PARAMETERS` instead of the actual values used by the application.

Syntax

`MMAP_LENGTH_LIMIT` *`length_limit`*

Options

`length_limit`

The user-defined limit in bytes of file size that the tool attempts to memory-map.

The default values for this parameter vary from application to application, and are listed in Table 3. These values are identical across all platforms and operating systems on which the tool is supported. Explicitly setting `MMAP_LENGTH_LIMIT` in the recipe forces that value to be the same for all applications.

Table 3 Default Values for MMAP_LENGTH_LIMIT

Application	Default in bytes
proteus	1,800,000,000
hierman	1,800,000,000
Celltool	4,000,000

NEW_CONTEXT_ANALYSIS_TBB

Description

This keyword enables you to perform the Hierman context analysis step using multiple CPU/cores (multi-thread).

Syntax

`NEW_CONTEXT_ANALYSIS_TBB [ON | OFF]`

Options

`ON`

Perform the Hierman context analysis step using multiple CPU/cores. The CPU/core count is determined by `TBB_THREAD_COUNT`. This is the default.

`OFF`

Perform the Hierman context analysis step using a single CPU/core.

See also

TBB_THREAD_COUNT on page 218

NEW_CREATE_BUFFER_SPEED

Description

Improves the runtime of the “Create OASIS decompression buffer file” step of hierarchy management.

Syntax

```
NEW_CREATE_BUFFER_SPEED ON|OFF
```

Options

`ON`

Turns on runtime improvements for the “Create OASIS decompression buffer file” step. This is the default.

`OFF`

Turns off runtime improvements for the “Create OASIS decompression buffer file” step.

NEW_DESIGN_READER_HIERMAN_SCAN

Description

`NEW_DESIGN_READER_HIERMAN_SCAN` decreases the elapsed runtime of the hierman scans of native patterns. The reader creates a cache of information that it uses the next time it is asked to read the same design. This cache is shared with other tools such as the Proteus WorkBench application. When the cache already exists, the speed improvement can be substantial.

Note: You might encounter an issue due to these cache files exceeding disk space limit set for your home directory. To prevent this, you can specify where hierman puts the layout cache by adding `default layout_cache_dir <path>` to the following:

```
$HOME/.synopsys/pwb/layout_rc.mac
```

Alternatively, you can specify the location for the cache file for an individual job in your .pjx using `DESIGN_READER_CACHE_PATH`.

Note: `INPUT_GDS_VALIDATION` is not supported with `NEW_DESIGN_READER_HIERMAN_SCAN`.

Syntax

```
NEW_DESIGN_READER_HIERMAN_SCAN ON|OFF
```

Options

`ON`

Turns on improved elapsed runtime for hierman scans.

`OFF`

Turns off improved elapsed runtime for hierman scans. This is the default.

See also

DESIGN_READER_CACHE_PATH on page 176

NEW_FLUSH_LAYOUT_OUTPUT

Description

This keyword determines whether to flush data in memory to layout output files when CTRL-C is pressed.

Syntax

```
NEW_FLUSH_LAYOUT_OUTPUT ON|OFF
```

Options

`ON`

When CTRL-C is pressed, flushes data to output files (for OASIS format) and then closes the output files safely (for both GDS and OASIS formats).

`OFF`

Does not flush data to output files when CTRL-C is pressed.

NEW_NORMALIZED_BIPOLAR_SYMMETRY

Description

When this keyword is ON (the default), under `SYMMETRY BIPOLAR`, the Proteus tool loads templates using a normalized symmetrically-equivalent orientation. Various rotations of symmetrically-equivalent instances all load using the same orientation. For example, a 180-degree representative instance loads at 0 degrees, and a 270-degree mirrored representative instance loads at 90 degrees non-mirrored.

In previous releases, `SYMMETRY_BIPOLAR` loaded the template using the representative instance's current orientation. For example, a 180-degree representative instance loaded at 180 degrees, while a 270-degree mirrored representative instance loaded at 270 degrees mirrored.

Note: When using this keyword, exercise caution if your recipe is converting from template-to-chip coordinates. Since the template now loads in the new symmetrically-equivalent orientation, any conversion logic from template-to-chip coordinates must be recoded. To facilitate this step, you can use the python API to query the state of the keyword:

```
info.parameter("NEW_NORMALIZED_BIPOLAR_SYMMETRY
")
```

If needed, you can use this value to control template-to-chip conversion logic.

Syntax

`NEW_NORMALIZED_BIPOLAR_SYMMETRY`

See also

SYMMETRY on page 215

NEW_ORPHAN_SIMULATION_PROCESS

Description

Improves TAT by performing efficient convolutions on the fields that are unpairable with each mask ("orphans") or split into non-packed fields before field operations.

Syntax

```
NEW_ORPHAN_SIMULATION_PROCESS ON|OFF
```

Options

`ON`

Turns on improved TAT. This is the default.

`OFF`

Turns off improved TAT.

NEW_OVERFLOW_CELL_TBB

Description

This keyword enables a new threading model to significantly improve overflow processing performance.

Syntax

```
NEW_OVERFLOW_CELL_TBB [ON | OFF]
```

Options

`ON`

Enable the new threading model for improved performance, using multiple CPU/cores. The CPU/core count is determined by `TBB_THREAD_COUNT`. This is the default.

`OFF`

Use the previous algorithm for overflow processing.

See also

TBB_THREAD_COUNT on page 218

NEW_REAL_IMAG_FILTER_ORDER

Description

Reorders the storage of internal filters in FBS to improve memory usage and TAT. There can be very small differences in results.

Syntax

```
NEW_REAL_IMAG_FILTER_ORDER ON|OFF
```

Options

`ON`

Turns on reordering of FBS filters. This is the default.

`OFF`

Turns off reordering of FBS filters.

NEW_REMOVE_CELL_OVERLAP

Description

The Proteus tool performs the correction on each template cell. Each template cell represents a group of instance cells (real placements of the template cell) that have the same geometric graphics and the same environment (context cells). Due to the cell overlap in the input data pattern, some geometric artifacts might exist in the correction output data. As such, it is desirable to remove any geometric overlaps between template cells before the correction process starts.

As the corrector collects the template graphics and its context graphics, it checks whether the template cell has any overlaps with its context cells. If any overlap exists, either template cell graphics or any of its context cell's graphics are cut before they are sent to correction.

Because `NEW_REMOVE_CELL_OVERLAP` comes into effect when the graphics are loaded into a dpserver for processing, it is not `TEMPLATE_BLOCK`-specific. `NEW_REMOVE_CELL_OVERLAP` behaves the same regardless of which `TEMPLATE_BLOCK` of a `PROTEUS_JOB_FLOW` recipe is being processed.

Syntax

```
NEW_REMOVE_CELL_OVERLAP ON|OFF
```

Options

`ON`

Turns on cell overlap removal. This is the default.

`OFF`

Turns off cell overlap removal.

Example

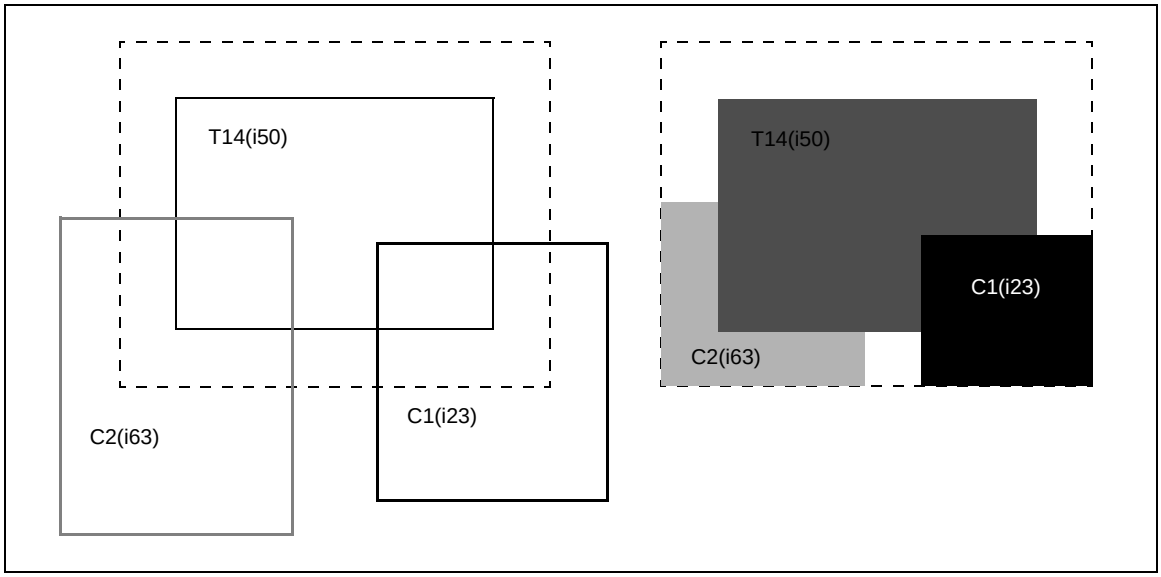


Figure 53 Removing Cell Overlap

In Figure 53, template cell 14 is the cell to be corrected. It has two context cells C1 and C2. Each template cell and context cell has an associated instance cell index. The cell with the lower instance cell index will cut the cell with the higher instance cell index. The graphics shown in the right figure are the ones to be corrected. T14 cuts C2 and is cut by C1. Using the instance cell index to determine the order of cell overlap cut assures consistent overlap removal on all template cells.

NEW_REPORT_INPUT_SCAN_STATS

Description

Use `NEW_REPORT_INPUT_SCAN_STATS` to instruct hierman to print out the polygon count for each defined layer alongside the layer name as it is defined in the `INPUT` and `END_INPUT` sections of a recipe. The results are printed to the log file (`./BASEPATH/JOBNAME.HMLOG`). This requires `NEW_DESIGN_READER_HIERMAN_SCAN ON`.

For example, with `NEW_REPORT_INPUT_SCAN_STATS ON`, the following lines:

```
INPUT ./small_coordinates.gds
 cor_layer_1     = 51:1
```

would produce the following polygon count information in the log file.

```
source layers:
 graphics layers:
     car_layer_1 -count 201
```

Note: If a polygon is in a cell that is placed multiple times, the polygon is counted only once, not multiple times.

Syntax

`NEW_REPORT_INPUT_SCAN_STATS ON|OFF`

Options

`ON`

Instructs hierman to print out the polygon count for each defined layer.

`OFF`

Will not include the polygon count with each defined layer in the log file. This is the default.

See also

NEW_DESIGN_READER_HIERMAN_SCAN on page 196

NEW_SCAN_FRAGMENTS_TBB

Description

This keyword enables you to perform the scanning cell graphics step (one of the fragment file validations between template calls) using multiple CPU/cores (multi-thread). This keyword is available with `PIPELINE_STRATEGY OFF` or `SINGLETONS`.

Syntax

`NEW_SCAN_FRAGMENTS_TBB [ON | OFF]`

Options

`ON`

Perform the scanning cell graphics step using multiple CPU/cores. The CPU/core count is determined by `TBB_THREAD_COUNT`. This is the default.

`OFF`

Perform the scanning cell graphics step using a single CPU/core.

See also

NEW_OFC_CONSOLIDATION_FILE on page 166

TBB_THREAD_COUNT on page 218

NEW_TEMPLATE_NUMBERS

Description

When `NEW_TEMPLATE_NUMBERS` is `ON`, the Proteus tool uses an advanced method of numbering templates that supports newer features such as concurrent pipeline, double patterning technology (DPT), and owned regions.

Syntax

```
NEW_TEMPLATE_NUMBERS ON|OFF
```

Options

`ON`

Uses advanced template numbering.

`OFF`

Does not use advanced template numbering. This is the default.

See also

CREATE_OWNED_REGION on page 143

NEW_OVERFLOW_CELL on page 167

PIPELINE_STRATEGY on page 261

NO_QUERY

Description

Each time you run proteus or hierman, the application checks whether there are files remaining from the previous run. For example, if you are trying to rerun proteus but the corrections have not been completed, or there is an output file, a warning is issued. Without `NO_QUERY`, the application waits for confirmation from you, as follows:

```
If you wish to ignore this warning, type Y.
Typing anything else will terminate.
```

If NO_QUERY is in the job control file, the application gives the warning and proceeds.

When the NO_QUERY keyword is assigned a timeout value, the Proteus tool waits for the user's response until the timeout expires. After the timeout expires, the tool will continue the job.

Syntax

NO_QUERY ON|OFF|*timeout_value*

Options

ON

The application gives the warning and proceeds.

OFF

The application queries and waits for user input before proceeding.

timeout_value

The application queries and waits for user input for the specified amount of time, then proceeds. The *timeout_value* can be between 1 and 86400 (inclusive), in units of seconds. The maximum timeout is 86400 seconds, or 24 hours.

POLYGON_FILL_RULE

Description

Certain polygon configurations, particularly self-intersecting polygons or paths, can often cause EVEN_ODD_RULE polygon fills to give an unexpected result. If you suspect you have a situation where this can be used, contact opc_help@synopsys.com for details.

Syntax

POLYGON_FILL_RULE EVEN_ODD_RULE | WINDING_RULE

Options

EVEN_ODD_RULE

Enclosure at a point is defined by the number of edges crossed by a ray from the point to another point an infinite distance away. If the number of edges crossed is odd, the point is inside the polygon; if the number of edges is even, the point is outside the polygon.

`WINDING_RULE`

Enclosure at a point is defined by the direction of the edges crossing a ray from the point to another point an infinite distance away. Edges crossing the ray from right to left increment the winding count, while edges crossing from left to right decrement the winding count. If the winding count is greater than 0, the point is inside the polygon. This is the default.

Example

Figure 54 is an illustration of how a self-intersecting polygon is handled for each rule.

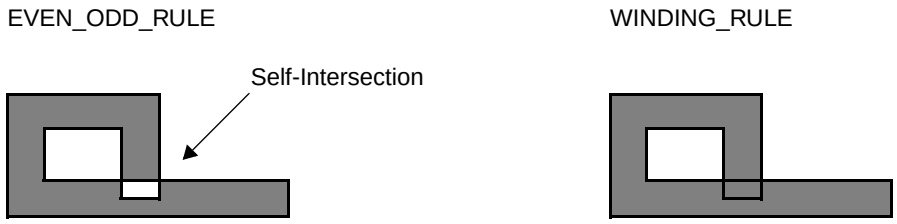


Figure 54 Polygon Fill Rules

POOL_MEMORY_INIT_SIZE

Description

`POOL_MEMORY_INIT_SIZE` specifies the initial size of memory pools at creation. This can be useful when using `POOL_MEMORY_STRATEGY LOCAL` because the size of a pool affects how soon it is reclaimed by the operating system upon its release. For the one-pool-per-object approach, some pools might be small compared with the centralized pool used when `POOL_MEMORY_STRATEGY` is set to `GLOBAL`. Setting the initial size of the memory pools sufficiently large ensures that they are reclaimed by the operating system immediately upon their release, thus reducing the memory footprint.

However, having too many of the increased size of pools active at the same time can increase the memory footprint. Thus, `POOL_MEMORY_INIT_SIZE` allows you to fine-tune your recipe to help reduce the memory watermark.

Syntax

`POOL_MEMORY_INIT_SIZE` *`size`*

Options

size

Specifies the initial size of a memory pool, in kilobytes. The default is 0. The recommended value is 128 so the allocated pool can be freed and claimed back by the operating system.

See also

POOL_MEMORY_STRATEGY on page 206

POOL_MEMORY_STRATEGY

Description

Basic Boolean operations, such as AND, OR, NOT, and sizing operations, as well as MLO layer operations, use a Synopsys geometry library whose memory is pool-based. Pool-based memory management allows the entire pool to be freed at once without the overhead of freeing each element individually or the need to keep a pointer to each element to free it later.

A problem with the centralized pools is that as the number of objects in these pools grows, the size of the pools could grow very big, very fast. This could result in a large memory footprint, especially for a “flat” recipe where thousands of operations are carried out in one run.

One solution to this is to replace the centralized memory pool with distributed local pools associated with objects, although there can be runtime and memory overhead from having too many local pools. Use `POOL_MEMORY_STRATEGY` to switch between the centralized pool and the fully distributed pool approach, according to which best meets the needs for your recipe.

The initial size of the memory pool can be specified using the keyword `POOL_MEMORY_INIT_SIZE`.

Syntax

```
POOL_MEMORY_STRATEGY GLOBAL|LOCAL
```

Options

`GLOBAL`

The centralized memory pool is used.

`LOCAL`

The one-pool-per-object approach is used. This is the default.

See also

POOL_MEMORY_INIT_SIZE on page 205

PROFILE_COMMAND

Description

This controls whether to output profiling data, such as runtime and memory information, for individual commands in a recipe.

Syntax

```
PROFILE_COMMAND DEFAULT|MLO
```

Options

`DEFAULT`

Profiling information is not included in the output. This is the default.

`MLO`

Profiling information for each MLO function will be in the output stdout of each dpserver or corexec, which will go to the server logs.

Example

Assume a recipe like the one shown here (with line numbers added):

```
26   TEMPLATE_BLOCK PROFILE_MLO_TEST(LAYER layer1, LAYER layer2)
27
28   from proteus.constraint import Constraint as Cst
29   from proteus import mlo2 as mlo
30
31   CHECKOUT_LICENSE "MLO"
32   PYTHON_BOOLEAN_OPS ON
33   layer2e = mlo.polygonToEdge(layer2)
34   layer4 = mlo.enclosing(layer1, layer2)
35   layer5 = mlo.enclosing(layer1, layer2e)
36   layer6 = mlo.edgeExpand(mlo.lengthEdge(layer2, value = 38
<= Cst.arg <= 38, connect = 'corner_connect'), outside = 5)
37
38   END_TEMPLATE_BLOCK (
```

The profiling report for the previous recipe is as follows:

```
TEMPLATE_BLOCK_PROFILE_MLO_TEST at line number :33
polygonToEdge complete, 0:00:00.000635  211028 kB/74176780 kB
TEMPLATE_BLOCK_PROFILE_MLO_TEST at line number :34     enclosing
complete, 0:00:00.000377      211028 kB/74176780 kB
TEMPLATE_BLOCK_PROFILE_MLO_TEST at line number :35     enclosing
complete, 0:00:00.000268      211028 kB/74176780 kB
TEMPLATE_BLOCK_PROFILE_MLO_TEST at line number :36    lengthEdge
complete, 0:00:00.000337     211028 kB/74176780 kB
TEMPLATE_BLOCK_PROFILE_MLO_TEST at line number :36    edgeExpand
complete, 0:00:00.000262     211028 kB/74176780 kB
```

RAM_LENGTH_LIMIT

Description

`RAM_LENGTH_LIMIT` is ignored by the correction executables corexec and dpserver.

This specifies the RAM length limit. The length limit is the maximum size in bytes of a Hierarchy Manager temporary file to attempt to load into RAM during hierarchy management. If the file is smaller than or equal to `RAM_LENGTH_LIMIT`, the file is loaded into RAM. If the file is larger than `RAM_LENGTH_LIMIT`, a cache is created for the file. The length of the cache is set using the keyword `CACHE_LENGTH`, which defaults to 512 megabytes. `CACHE_LENGTH` specifies the total amount of RAM used by all file caches.

Loading files into RAM can significantly improve the Hierarchy Manager's performance, and have an overall effect on correction performance. This limit can also be reduced to attempt to get larger jobs to run on smaller machines, such as 32-bit Linux machines.

The limit of this keyword for a 64-bit machine is a file size of (2^{63})-1 bytes. Sizes between 2^{53} and (2^{63})-1 bytes are supported, but are rounded to fit into a double-precision floating point value.

Syntax

`RAM_LENGTH_LIMIT` *`length_limit`*

Options

`length_limit`

The maximum size in bytes of a proteus temporary file to attempt to load into RAM.

The default values for this parameter vary from application to application, and are listed in Table 4. These values are identical across all platforms and operating systems on which the Proteus tool is supported. Explicitly setting `RAM_LENGTH_LIMIT` in the recipe forces that value to be the same for all applications.

Table 4 Default Values for RAM_LENGTH_LIMIT

Application	Default in bytes
proteus	2,147,483,647
hierman	2,147,483,647
Celltool	268,435,455

See also

CACHE_LENGTH on page 224

REPORT_PARAMETERS

Description

To print a parameter report, add the line `REPORT_PARAMETERS` to a `.pjx` file. The parameter report includes the current and default values for all JCL or `DPROTEUS_CFG` parameters you have set in the current job.

The text report is printed by hierman, corexec, or proteus to the regular screen output, in a format that you can insert into a `.pjx` file in the future, if desired. By copying and pasting the parameter report printed from `REPORT_PARAMETERS`, you can run a job on some future version of the Proteus tool and ensure that all JCL or `DPROTEUS_CFG` parameters keep the same values, even if the tool has changed some of the defaults from the earlier version to the now-current version.

Putting a keyword in the `.pjx` file with the single argument `DEFAULT` instructs hierman, corexec, or proteus to report on that keyword's parameter value, but not change it from the default. For example, `SIZE_CLUSTER DEFAULT` causes Proteus applications to use the current default value for `SIZE_CLUSTER` and report it in the parameter report even if you have not changed it.

The parameter report resulting from `REPORT_PARAMETERS` starts with the line `SET_PARAMETER_DEFAULTS`. The report's header indicates which version of hierman, corexec, or proteus is being executed.

As long as REPORT_PARAMETERS is present in the .pjx file, hierman and every correction run generates a report.

Syntax

REPORT_PARAMETERS

Example

The following example shows a sample parameter report. In this example, SET_PARAMETER_DEFAULTS shows that all parameters that have not been set by the .pjx file have been set to the default values for the J-2014.06 release.

The DP config file section (highlighted) appears only in the report printed by proteus (not hierman). If any DPROTEUS_CFG parameters were overridden on the proteus invocation line, the comment lines at the beginning of the highlighted section would include "modified by the command line".

```
'Begin parameter report
'proteus Release J-2014.06 Revision Proteus_J-2014.06 (64f/64m
LINUX_X86_64).
SET_PARAMETER_DEFAULTS      J-2014.06

'DP config file settings, from DPROTEUS_CFG file "/project/foo/
'dproteus.cfg":
DPSERVER_HEARTBEAT_TIME                120 ' Default was 3600
DPSERVER_HEARTBEAT_TIMEOUT            2400 ' Default was 11160
END_PORT                              2656 ' Default was 2445
KEEP_INACTIVE_SERVERS                    5 ' Default was 20
MAX_BURST_COUNT                         20 ' Default was 50
NO_LOCAL_SERVER                         ON ' Default was OFF
NO_SVR_VERSION_CHECK                    ON ' Default was OFF
START_PORT                            2356 ' Default was 2346
' End DP config file settings

BASEPATH                    ABC_BP/
INPUT_FILE                  Cel.oas
JOBNAME                     HIERMAN
MAX_CLUSTER                 20000
OUTPUT_FILE                 Out.oas
OVERRIDE_COR_AMBIT          1024
OVERRIDE_HIER_AMBIT         1024
OVERRIDE_INF_LOOP_MAX       333444555     ' Default was 20000000
PYTHON_BOOLEAN_OPS          ON            ' Default was ON
SREF_SCAFFOLD               0             ' Default was 20000000
' End parameter report
```

See also

SET_PARAMETER_DEFAULTS on page 212

RETAIN_JOB_FLOW_FILES

Description

If `RETAIN_JOB_FLOW_FILES` is included in your job control file, you will notice intermediate OASIS files for all `TEMPLATE_CALL`s prior to the last one. Each file is named for its individual template block, using the naming convention *jobname*_TB*num*_out.oas. The final output file has the name you specified in the `PROTEUS_JOB_FLOW` section. The last template block is described in a separate section in this final output file.

`RETAIN_JOB_FLOW_FILES` also retains intermediate .pjx files for each `TEMPLATE_BLOCK`.

Note: Intermediate `.pjx` files are only provided for reference purposes. Do not attempt to run them.

Syntax

```
RETAIN_JOB_FLOW_FILES ON|OFF
```

Options

`ON`

Does not retain intermediate OASIS and `.pjx` files.

`OFF`

Retains intermediate OASIS and `.pjx` files for reference purposes. This is the default.

ROTATE_IN

Description

The `ROTATE_IN` JCL keyword allows you to rotate an incoming layout around the origin, in 90 degree increments, with the rotate layout snapped to the incoming grid. The default is 0 degrees. Specify the value for `ROTATE_IN` as either positive or negatIve, and it must be a multiple of 90 degrees.

If you do not specify this keyword, the Proteus tool does not perform any rotation of the input layout.

Using this keyword will not change the existing behavior of the `SYMMETRY` setting; the orientation of every instance will be honored. If you use `SYMMETRY ALL` and a dpserver loads the template, this representative instance will ignore

the ROTATE_IN setting. This ensures SYMMETRY ALL does not produce different results when you use different rotation values.

Examples

```
ROTATE_IN 90
```

See also

ROTATE_OUT on page 111

SCALE_IN on page 106

SYMMETRY on page 215

SET_PARAMETER_DEFAULTS

Description

The parameter report resulting from REPORT_PARAMETERS starts with the line SET_PARAMETER_DEFAULTS (regardless whether SET_PARAMETER_DEFAULTS is present in the .jcl file). This shows you the Proteus tool release version and, for an application reading it in a .jcl file, sets the values for all the parameters that have not been set by the .pjx file.

SET_PARAMETER_DEFAULTS tells you that all parameters that have not been set by the .pjx file have been set to the default values for the release specified. When SET_PARAMETER_DEFAULTS is encountered in a .jcl file, the application resets all the JCL- or DPROTEUS_CFG-settable parameters to their original defaults as of the release specified. The specified release must be D-2010.06 or later; there is no provision to use SET_PARAMETER_DEFAULTS to return to earlier defaults.

The position of the SET_PARAMETER_DEFAULTS keyword in the recipe is relevant. When positioned at the top of the recipe (the suggested position), SET_PARAMETER_DEFAULTS allows any parameter changes defined below it to override the SET_PARAMETER_DEFAULTS setting with the specified values. When positioned below keywords in the recipe, SET_PARAMETER_DEFAULTS overrides preceding specified keyword values with their default values.

Syntax

SET_PARAMETER_DEFAULTS *fully_qualified_release_name*

Options

`fully_qualified_release_name`

Takes the format year.month-patch of the release; for example, 2014.06-4. Note that `SET_PARAMETER_DEFAULTS` cannot be used to set keyword defaults to any release before 2010.06.

See also

REPORT_PARAMETERS on page 209

SUPPRESS_CORBASIC_DEPRECATION_WARNING

Description

This keyword turns off warnings associated with deprecated Synopsys recipe framework functions (for example, `segActualHeadF(ftype, polygon, segment)`, `thisSegActualHeadF()`, and so forth). The deprecated recipe framework functions are declared in the `.brcp` files by a `DEPRECATED_FUNCTION` statement.

Syntax

```
SUPPRESS_CORBASIC_DEPRECATION_WARNING ON|OFF
```

Options

`ON`

Turns off warnings for deprecated recipe framework functions.

`OFF`

Warnings for deprecated recipe framework functions are issued. This is the default.

SUPPRESS_ECOSYSTEM_WARNINGS

Description

When Proteus Ecosystem is enabled, the tool suggests optimized keyword settings to help you use new Proteus features. Recommended settings are printed during recipe parsing. For example:

```
Warning: CREATE_OWNED_REGION is enabled, forcing
NEW_TEMPLATE_NUMBERS...
Warning: CREATE_OWNED_REGION is enabled, forcing
NEW_SBC_OVERLAP_ORDERING...
Warning: CREATE_OWNED_REGION is enabled, forcing
CREATE_OWNED_NEIGHBORS...
Warning: CREATE_OWNED_REGION is enabled, forcing
GRAPH_SCAFF_OVERLAP 0...
Warning: CREATE_OWNED_REGION is enabled, forbidding CLUSTER
HIER...
Warning: CREATE_OWNED_REGION is enabled, setting CLUSTER FLAT as
default...
```

This keyword enables you to suppress Ecosystem printing activity.

Syntax

`SUPPRESS_ECOSYSTEM_WARNINGS ON|OFF`

Options

`ON`

Suppresses Ecosystem printing activity.

`OFF`

Permits Ecosystem printing activity. This is the default.

See also

BOSS_CALL on page 48

CREATE_OWNED_REGION on page 143

SUPPRESS_SYMMETRY_WARNING

Description

A warning is normally issued when a direction-dependent Boolean or MLO command is used when `SYMMETRY` is set to a value other than `NONE`. This keyword suppresses the `SYMMETRY` warning at the global or `TEMPLATE_CALL` level.

The `TEMPLATE_CALL`-level setting will be in effect for the particular `TEMPLATE_CALL`, regardless of the global setting. By default, the setting is `OFF` and the warning is not suppressed.

Syntax

To add the keyword at the global level, use:

```
SUPPRESS_SYMMETRY_WARNING ON|OFF
```

In the `TEMPLATE_CALL`, use:

```
TEMPLATE_CALL(template1(), (SUPPRESS_SYMMETRY_WARNING ON|OFF))
```

Options

`ON`

Suppresses the symmetry warning.

`OFF`

The symmetry warning is not suppressed. This is the default.

See also

SYMMETRY on page 215

SYMMETRY

Description

The `SYMMETRY` keyword is primarily used to process the layout to match the symmetry of the model being used for correction. This keyword sets limits on the set of cell placements that can be represented by a single correction template. It affects template generation within the Hierarchy Manager.

Note: Changing the `SYMMETRY` value can cause small differences in output for cells that have different transformations. This is due to the increase or decrease in the number of the representative templates for a given repeating cell based on the change in the `SYMMETRY` value.

For example, consider two cells that are placed in identical environments, but one is placed with a different rotation. When `SYMMETRY` is `NONE`, these two placements are two separate templates and processed separately. If `SYMMETRY` is changed to `ALL`, the two cells are represented by a single template. These different `SYMMETRY` settings can result in small XOR differences against the original pattern. This can be due to one cell's template placement being different during the template processing, which can result in slight differences in simulation and/or vertex and edge snapping.

Also, be sure you set `OVERRIDE_COR_AMBIT` and `OVERRIDE_HIER_AMBIT` to identical values so that differences due to hierarchical placement are observed when creating templates.

Syntax

```
SYMMETRY ALL | BIPOLAR | NONE
```

Options

`ALL`

A single hierman template can represent a set of instances of an input cell placed with any combination of GDSII REFLECTION and GDSII ANGLE properties (mirror and rotation), provided that their contexts are identical. This is the default setting.

Note: Whenever the illumination source is asymmetric (for example, a dipole), `SYMMETRY` should not be set to `ALL`.

`BIPOLAR`

A single correction template can represent only a set of instances of an input cell placed with values of REFLECTION or ANGLE properties (mirror and rotation) that differ by 180 degrees, provided that their contexts are identical. Input placements that differ by other ANGLE are represented by different correction templates.

Note: The Proteus tool loads templates using a normalized symmetrically-equivalent orientation. Various rotations of symmetrically-equivalent instances all load using the same orientation. For example, a 180-degree representative instance loads at 0 degrees, and a 270-degree mirrored representative instance loads at 90 degrees non-mirrored.

`NONE`

A single hierman template can represent only a set of instances of an input cell placed with identical values of REFLECTION and ANGLE properties (mirror and rotation), provided that their context is identical. Input placements with different REFLECTION or ANGLE properties will be represented by separate correction templates.

Example

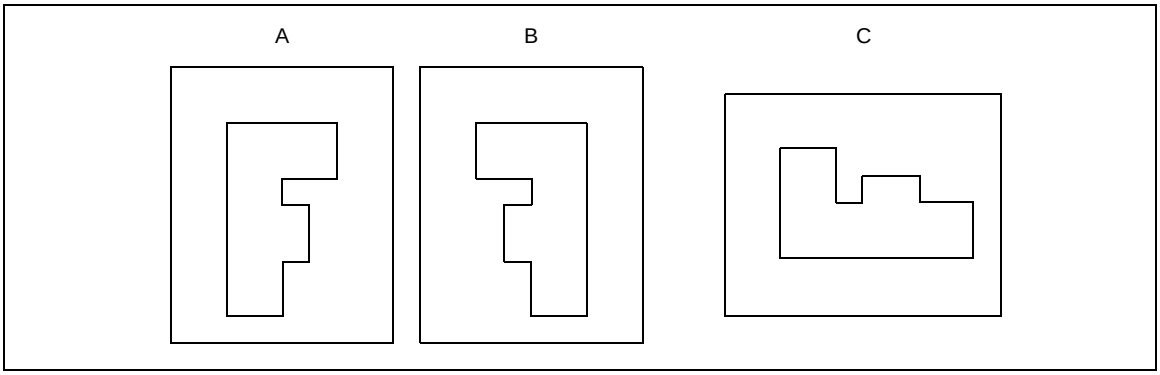


Figure 55 Cell Placements

In Figure 55, assume that the placements of A, B, and C (instances of the same cell) are sufficiently distant from other cells that their correction shapes are unaffected by their neighbors. The different `SYMMETRY` arguments provide the following results:

`ALL`

One correction template is created and placed three times.

`BIPOLAR`

Two correction templates are generated, one to be placed in locations A and B, and the other for C.

`NONE`

Three correction templates are generated, one for each unique orientation.

In addition to this hierarchy expansion, when `SYMMETRY` is `NONE` or `BIPOLAR`, the data to be used for correction is transformed to a representative instance rotation and mirror when data extraction occurs (prior to `PROTEUS_JOB_FLOW`). This ensures the correct interpretation of data by nonsymmetrical models, such as bipolar stepper configurations.

See also

NEW_NORMALIZED_BIPOLAR_SYMMETRY on page 198

TBB_THREAD_COUNT

Description

This keyword specifies the thread count of the TBB thread pool. It determines the number of CPU/cores used in `NEW_CONTEXT_ANALYSIS_TBB` and other `_TBB` keywords.

Syntax

`TBB_THREAD_COUNT` *`float`*

Options

`float`

When *`float`* > 1, specifies the actual thread count.

When 0 < *`float`* <= 1, specifies the thread count as the fraction of the total CPU/cores on the host.

When *`float`* = 0, uses the default thread count (4).

If *`float`* < 0, issues an error.

For a 16-CPU/core host, the thread count would be determined as follows:

TBB_THREAD_COUNT	Thread count
0	4
0.5	8
1	16
4 (default)	4
16	16

See also

NEW_CONTEXT_ANALYSIS_TBB on page 195

NEW_OFC_CONSOLIDATION_FILE on page 166

NEW_SCAN_FRAGMENTS_TBB on page 202

TEMPPATH

Description

This specifies the directory in which temporary correction files are stored. The default is `BASEPATH`.

When `TEMPPATH` is specified, uncompressed OASIS files are written to this directory. For this reason, if you explicitly specify a directory on a local file system for `TEMPPATH`, you might see improved performance of the graphics scaffolding operation.

When the directory is not followed by a forward slash (/), `TEMPPATH` treats the text string that follows the last forward slash as a prefix for both temporary files and Hierarchy Manager files.

Syntax

`TEMPPATH` *`path`*

Options

`path`

The path to the directory in which temporary correction files are stored. The path can take one of the following forms:

```
TEMPPATH ./TMP/
TEMPPATH ./TMP
```

See also

BASEPATH on page 174

TOPCELL_IN

Description

This overrides the correction processor's automatic determination of the top cells in the input pattern. If more than one cell in the input library has no references from any other cell in the library, the correction processor treats them all as top cells referenced from the absolute origin by default. If the data in multiple top cells overlap, the correction processor treats any overlapping data as valid environment data between top cells by default.

Use `TOPCELL_IN` to choose one or more top cells. `TOPCELL_IN` can also specify a cell that is not a real top cell.

Note: All other hierarchy not referenced by the declared top cell(s) is ignored. This can be used to correct portions of a pattern; for example, only the SRAM.

Syntax

`TOPCELL_IN` *`topcell1`* `[`*`topcell2 topcell3 ...`*`]`

Options

`topcell`

The user-defined top cell(s). A cell name cannot exceed 127 characters.

See also

TOPCELL_OUT on page 113

5

Correction

Provides job control keywords associated with the correction recipe.

Correction Recipe Job Control Keywords

The following keywords provide options for the correction recipe in the job control file. These keywords initiate prior to starting template correction.

ALL_ANGLE_OUTPUT

Description

This bypasses routine snapping of non-45-degree data to 45-degree data. To preserve non-45 degree data, use `ALL_ANGLE_OUTPUT`. The keyword will not be able to preserve the non-45-degree data in a corBASIC flow.

Note: `ALL_ANGLE_OUTPUT ON` results in a warning that non-45-degree data is not preserved in a corBASIC flow.

Syntax

```
ALL_ANGLE_OUTPUT ON|OFF
```

Options

`ON`

Non-45-degree data is preserved and not snapped to 45-degree data.

`OFF`

Non-45-degree data is snapped to 45-degree data. This is the default.

ALLOW_HOLE_WITHOUT_PARENT

Description

When this keyword is `OFF` (the default), proteus issues an error message when a hole without parent (a negative-area polygon) is found during processing. Adding `ALLOW_HOLE_WITHOUT_PARENT` to the recipe suppresses such error messages.

Syntax

```
ALLOW_HOLE_WITHOUT_PARENT ON|OFF
```

Options

`ON`

Hole without parent errors are allowed.

`OFF`

Hole without parent errors are not allowed. This is the default.

See also

WARN_HOLE_WITHOUT_PARENT on page 241

APPROXIMATE_COMPARE_TOLERANCE

Description

When the global job control keyword `USE_APPROXIMATION_FOR_EQUALITY_OPS` is `ON`, comparison operations in corBASIC run in the approximate mode. (See the *corBASIC Reference Manual* for details.) Use `APPROXIMATE_COMPARE_TOLERANCE` to define how close two numbers must be in order to be regarded as equal in the approximate mode.

Syntax

`APPROXIMATE_COMPARE_TOLERANCE` *`value`*

Options

`value`

A value defining ULP (Units of Least Precision). In the Proteus tool, 1 ULP is about $2*10^{-16}$ relative to the number. The default is 10.

See also

USE_APPROXIMATION_FOR_EQUALITY_OPS on page 239

APPS_CURRENT_ITER

Description

In a multiblock recipe, each template block can go through multiple iterations ("ITER") as OPC is performed, yet some functions operate only on a certain iteration. `APPS_CURRENT_ITER` is used within a `TEMPLATE_CALL` as a reference point for the current iteration. For example, if 2 iterations have already been performed, this value would be set to 3.

This keyword is overridable in the `TEMPLATE_BLOCK`. For example, if you have a recipe with two `TEMPLATE_BLOCK`s having four OPC iterations each, you could add `APPS_CURRENT_ITER 5` to the second `TEMPLATE_BLOCK` to reset the current iteration to 5 instead of 0 (the default) at the start of the second block.

Syntax

`APPS_CURRENT_ITER` *`current_iter`*

Options

`current_iter`

The current iteration. The default is 0.

APPS_NUM_ITERS

Description

In a multiblock recipe, each template block might go through multiple iterations ("ITER") as OPC is performed, yet some functions operate only on a certain iteration. `APPS_NUM_ITERS` is used within a `TEMPLATE_CALL` to set the number of iterations that should be performed within a `TEMPLATE_BLOCK`.

This keyword is overridable in the `TEMPLATE_BLOCK`. For example, if `APPS_NUM_ITERS` is set to 1 in your multiblock recipe, but you would prefer that the second `TEMPLATE_BLOCK` undergo 3 iterations, you can add `APPS_NUM_ITERS 3` to the second `TEMPLATE_BLOCK`.

Syntax

`APPS_NUM_ITERS` *`num_iters`*

Options

num_iters

The number of iterations to perform on the current TEMPLATE_BLOCK. The default is 1.

BANDWIDTH

Description

This specifies the graphics database granularity parameter. Decreasing *band_size* can marginally decrease correction processing time, but consumes more memory resources. The default value for *band_size* is the correction recipe ambit value.

Syntax

BANDWIDTH *band_size*

Options

band_size

The user-defined bandwidth.

CACHE_LENGTH

Description

This determines the length (or amount) of RAM allocated to caching hierman files and graphics from the input layout file within the correction executables (corexec and dpserver). This parameter has no effect in hierman. The default is 768 megabytes (805306368 bytes).

Syntax

CACHE_LENGTH *value*

Options

value

The RAM amount.

See also

RAM_LENGTH_LIMIT on page 208

CORGRID

Description

During the hierman phase, `CORGRID` specifies the snapping increment (in nanometers) to apply to corrected figures. This number usually corresponds to the mask grid on which the final, properly scaled mask pattern is to be generated.

Coordinates are first snapped to the value used by `UPDATE_GRAPHICS` and then to `DBU_PROC`; exact half values of `DBU_PROC` are rounded up.

Syntax

`CORGRID` *`grid`*

Options

`grid`

The snapping increment. Can contain a floating point value. The default value is 1 nanometer.

See also

DBU_PROC on page 229

CORGRID on page 104

CORRECTION_ORDER

Description

This specifies the order in which templates are corrected. The default value is `REVERSE`. The optional second value defaults to `NORMAL`.

The proteus command-line option for correction order (`-r`) overrides the `CORRECTION_ORDER` keyword. See proteus on page 296.

Note: `CORRECTION_ORDER` is not supported when `PIPELINE_STRATEGY` is in the recipe unless `CORRECTION_ORDER` is set to `ROWS`.

Syntax

```
CORRECTION_ORDER
    AREA|LL|NORMAL|RANDOM|REVERSE
    COLUMNS[NORMAL|REVERSE]
    ROWS[NORMAL|REVERSE]
```

Options

`AREA`

In `AREA` mode, the template list is reordered according to the bounding box area of the cell, and proteus processes the templates from largest to smallest.

`COLUMNS`

In `COLUMNS` mode, templates are corrected in column order.

If you specify `COLUMNS NORMAL` (the default), correction begins with the leftmost column first then ends with the right-most column. If you specify `COLUMNS REVERSE`, correction begins with the right-most column and ends with the left-most column. The width of each column is the value of the `SPATIAL_CORRECTION_BIN_SIZE` keyword, which defaults to 10 times `MAX_CLUSTER`.

`LL`

In `LL` (“lower left”) mode, templates in bins on the left and bottom of the chip are corrected first. (Within each bin the templates are sent for correction according to their template number, in ascending order.) The correction then proceeds in row order with the bottom-most row of uncorrected bins first and proceeding to the top-most row.

Correcting the left and bottom of the chip first allows the Proteus-CATS exchange API to compute the corrected coordinate of the lower-left corner as early as possible so that Proteus-CATS exchange API clients (who need such information) can be started while OPC correction is still going on.

`NORMAL`

In `NORMAL` mode, templates are corrected in normal order; that is, templates are corrected in increasing template number.

`RANDOM`

In `RANDOM` mode, `CORRECTION_ORDER` causes a random template order during distributed processing.

`REVERSE`

`REVERSE` mode forces proteus to process the template list in reverse (n-0) order. This is the default.

`ROWS`

In `ROWS` mode, templates are corrected in row order.

If you specify `ROWS NORMAL` (the default), correction begins with the bottom-most row and ends with the topmost row.

If you specify `ROWS REVERSE`, correction begins with the topmost row and ends with the bottom-most row. The height of each row is the value of the `SPATIAL_CORRECTION_BIN_SIZE` keyword, which defaults to 10 times `MAX_CLUSTER`.

Examples

For the purpose of spatial correction orders, such as columns or rows, the chip is divided into bins, where the height and width of each bin is specified by the value of `SPATIAL_CORRECTION_BIN_SIZE`. Each bin is referred to in the log file by a number that is dependent on the setting of `CORRECTION_ORDER`, as shown in the following figures.

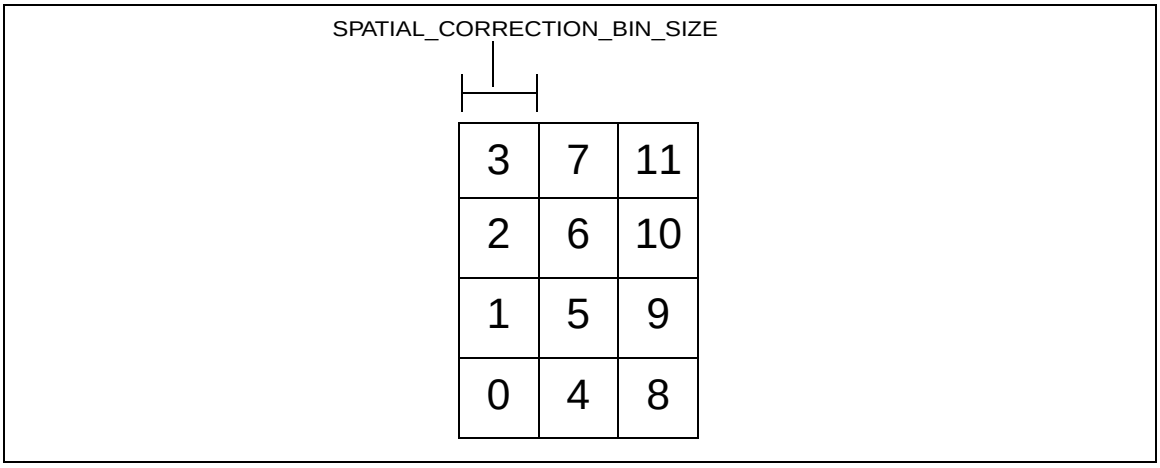


Figure 56 CORRECTION_ORDER COLUMNS

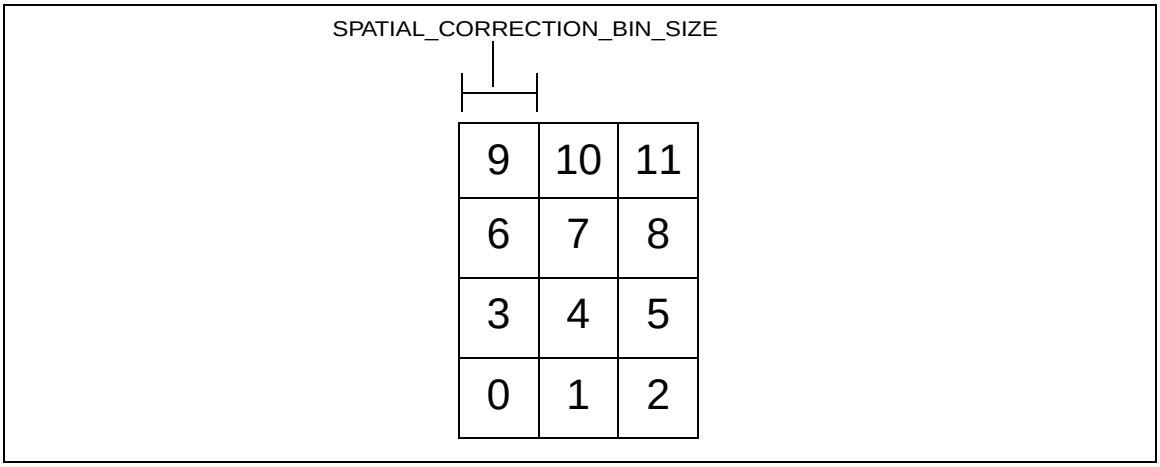


Figure 57 CORRECTION_ORDER ROWS

The templates in each bin are also referred to in the log file by number, which does not change even when the bin number does when using `COLUMNS` versus `ROWS`. In other words, the bottom middle bin in the examples above always has templates 3 and 4, whether the bin is named bin 1 or bin 4.

For example, assume that `CORRECTION_ORDER` is set to `COLUMNS` and that there are two templates in every bin (in other words, template 1 and template 2 are in bin 0, template 3 and template 4 are in bin 1, and so forth). The log file for this job shows the template number first, then the bin number, and might read as follows:

```
2 done in bin 0
1 done in bin 0
4 done in bin 1
3 done in bin 1
...
10 done in bin 4
9 done in bin 4
12 done in bin 5
11 done in bin 5
```

The log file for the same job but with `CORRECTION_ORDER` set to `ROWS` might read as follows:

```
2 done in bin 0
1 done in bin 0
8 done in bin 1
7 done in bin 1
...
6 done in bin 4
5 done in bin 4
12 done in bin 5
11 done in bin 5
```

For the purpose of `CORRECTION_ORDER AREA`, all templates are sorted by their area, from largest to smallest. (If two templates have the same area, the one with the higher template id number comes first.) The Proteus tool generates a certain number of bins based on the number of cluster instances. The tool then puts the sorted cluster templates into these bins.

For example, if there are 9 templates, there are 3 bins. Assume the sorted template id list is 4,6,7,3,1,9,8,5,2, where 4 has the largest area and 2 has the smallest.

The tool puts templates 4,6,and 7 in bin 1; templates 3,1,and 9 in bin 2; and templates 8,5, and 2 in bin 3. The log file for the job might read as follows:

```
4 done, bin 1
6 done, bin 1
...
5 done, bin 3
2 done, bin 3
```

For the purpose of `CORRECTION_ORDER RANDOM`, the tool generates a certain number of bins based on the number of cluster instances. The tool then puts each cluster template into a random bin, so the number of templates in every bin could be different.

See also

SPATIAL_CORRECTION_BIN_SIZE on page 239

DBU_PROC

Description

`DBU_PROC` is the database unit used for all internal processing in the Proteus tool. It is user-configurable and can help minimize snapping errors. The default

DBU_PROC value is 1.0 nanometers. This is overridable in the TEMPLATE_BLOCK.

Syntax

DBU_PROC *n* | *n m*

Options

n

If only *n* is supplied, this is the DBU_PROC value.

m

If both *n* and *m* are supplied, the DBU_PROC value is *n* divided by *m*.

Example

```
DBU_PROC 1 4
```

See also

CORGRID on page 225

DBU_OUT on page 108

SCALE_OUT on page 112.

ERROR_ON_POLYNO_OVERFLOW

Description

With the keyword ERROR_ON_POLYNO_OVERFLOW you can control whether an error message is reported (and the program stops execution) when the polygon number requested through the SET_POLY_NO function exceeds the number of available polygons for the current FTYPE.

Syntax

ERROR_ON_POLYNO_OVERFLOW ON|OFF

Options

ON

An error is reported if the polygon number given to SET_POLY_NO is greater than the number of polygons available.

OFF

No error is reported if the polygon number given to SET_POLY_NO is greater than the number of polygons available. This is the default.

GRAPHICS

Description

When set to `OFF`, this deactivates the default X-window correction monitor used by the shape processor. Screen graphics processing can consume a significant amount of time so `GRAPHICS OFF` is normally used for big jobs in a production environment. The `GRAPHICS` setting is `ON` by default.

Options

`ON`

Turns on the default X-window correction monitor.

`OFF`

Turns off the default X-window correction monitor.

Syntax

```
GRAPHICS ON|OFF
```

LOGFILE

Description

This specifies the base name of an optional data log file in which user-specified information can be recorded during correction. The `LOGFILE` is specified without a path, and is placed in the temporary data file directory declared with `BASEPATH`. See LOGDATA and PRINTLINE in the *corBASIC Reference Manual* to see how data is written to the `LOGFILE` during correction.

Syntax

`LOGFILE` *`log_filename`*

Options

`log_filename`

The base name of the data log file.

See also

BASEPATH on page 174

MAX_CALL_DEPTH

Description

`MAX_CALL_DEPTH` controls the maximum corBASIC call stack depth. Note that exceeding the maximum corBASIC call stack depth is usually the result of a corBASIC coding error, in which case raising the maximum call depth will only result in seeing the same "call depth exceeded" error once again, with a deeper stack.

Syntax

`MAX_CALL_DEPTH` *`number`*

Options

`number`

The call depth. Defaults to 1024.

MIN_N45_L

Description

This specifies the granularity of corrected approximations to non-45-degree polygon edges. All correction segments are generated on 45- or 90-degree orientations.

The crimp length defaults to `MIN_FEATURE`.

Syntax

`MIN_N45_L` *`crimp_length`*

Options

`crimp_length`

The user-defined granularity of corrected approximations to non-45-degree polygon edges.

Example

Figure 58 illustrates the crimping approximation with *`crimp_length`*.

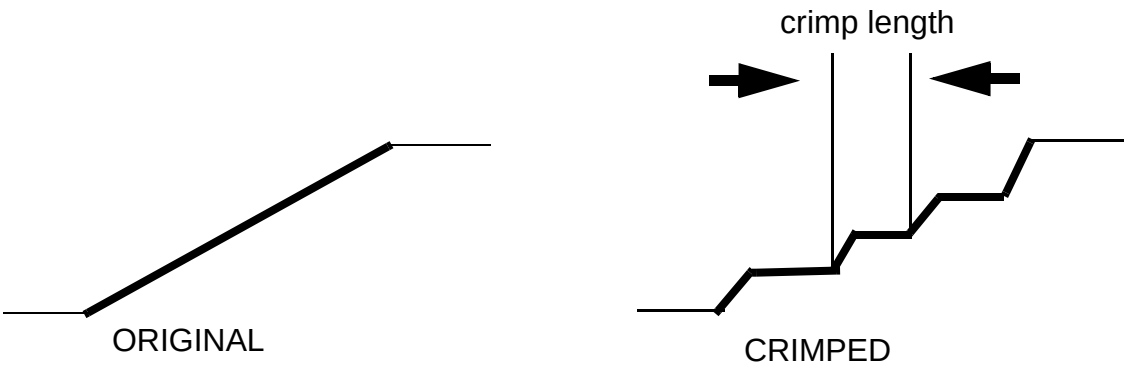


Figure 58 Crimp Length

See also

MIN_FEATURE on page 194

Crimping Near Small Angles

When crimping is applied to adjacent edges that meet to form angles smaller than 45 degrees, there is a chance that the crimped edges could intersect, as shown in Figure 59A. If this happens, the algorithm automatically identifies the intersection point that is farthest from the original vertex, and then removes the original vertex along with all vertices between the intersection point and the original vertex. This prevents the crimp algorithm from creating self-intersecting polygons from adjacent crimped edges. The result is shown in Figure 59B.

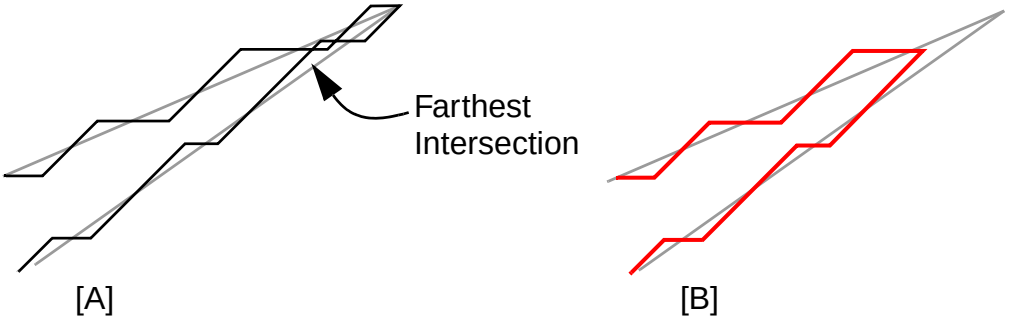


Figure 59 Crimping Near a Small Angle

In rare cases it is possible for crimping to introduce intersections between nonadjacent edges. This case is *not* currently addressed by the crimping algorithm.

MODEL_REMOVE_OVERLAP

Description

In some situations, an overlap can occur during biasing, either during correction or in precorrection. If the correction recipe might produce some overlaps or underlaps on a layer, setting `MODEL_REMOVE_OVERLAP ON` causes the model to be computed as if the overlap/underlap on a given layer had been cleaned up prior to the model being run.

Syntax

```
MODEL_REMOVE_OVERLAP ON|OFF
```

Options

`ON`

Turns on removal of overlap.

`OFF`

Turns off removal of overlap. This is the default.

Example

Ignores the overlap of polygons A and B on the left, producing simulation results similar to polygon C on the right.

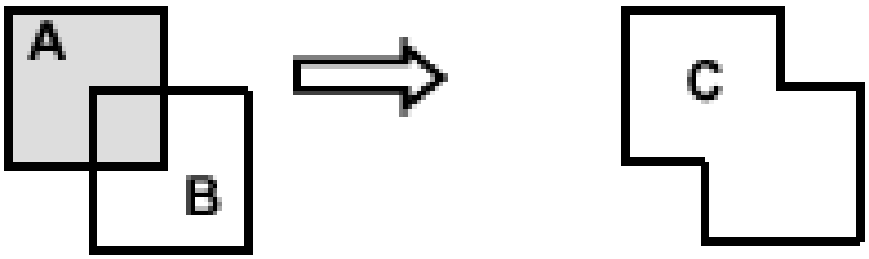


Figure 60 MODEL_REMOVE_OVERLAP ON

NEW_ITERATE_REFIN

Description

`NEW_ITERATE_REFIN` (`ON` by default) allows you to iterate over REF layers.

Syntax

```
NEW_ITERATE_REFIN ON|OFF
```

Options

`ON`

Turns on ability to iterate over REF layers.

`OFF`

Turns off ability to iterate over REF layers. This is the default.

Example

The following example shows a corBASIC function using `NEW_ITERATE_REFIN`. When `NEW_ITERATE_REFIN` is `ON` (the default), you are able to set `FTYPE` to 2 and iterate over all the polygons in the REF layer.

```
FUNCTION runRef()
SET_FTYPE(2)
IF N_POLY(2) == 0 THEN RETURN 0
FOR poly = 0 TO N_POLY(2)-1
  SET_POLY_NO(poly)
  FOR seg = 0 TO N_SEG(2,poly)-1
    SET_SEG_NO(seg)
    IF TYPE == 2 THEN
      IF LOGVAL(3) == 0 THEN
        PRINT("info.txt","%d %d %d %d %d %d %d %d\
        n",FTYPE,POLY_NO,SEG_NO,DIR,TLEN,THD,XCOORD(),YCOORD())
        makeBox(1,1,-1,1,8)
      ENDIF
    ENDIF
  NEXT seg
NEXT poly
RETURN 0
```

NEW_SNAP_EDGE_BASED

Description

`NEW_SNAP_EDGE_BASED` (`OFF` by default) turns on an improved snapping method, particularly for 45-degree edges.

Syntax

`NEW_SNAP_EDGE_BASED ON|OFF`

Options

`ON`

Turns on improved snapping method.

`OFF`

Turns off improved snapping method. This is the default.

OVERRIDE_COR_AMBIT

Description

In the correction engine, this overrides the ambit value (defined by the recipe or model file) used to determine which template .context graphics are seen by the corrector.

In practice, `OVERRIDE_COR_AMBIT` should be set equal to `OVERRIDE_HIER_AMBIT`. `OVERRIDE_COR_AMBIT` determines which graphics are in the .context of the template, and the same graphics should be used for context analysis (set by `OVERRIDE_HIER_AMBIT`). During hierman template generation, only cell instances that have identical context data within `OVERRIDE_HIER_AMBIT` distance away are grouped in to a single template. If the data is different, a separate template is created for each different case.

During processing, it is possible to include more or less context data than originally specified by `OVERRIDE_HIER_AMBIT`. This is done by setting `OVERRIDE_COR_AMBIT`. This value can be safely set to values less than or equal to the value of `OVERRIDE_HIER_AMBIT`. It is possible to set `OVERRIDE_COR_AMBIT` greater than the `OVERRIDE_HIER_AMBIT` setting. However, the extra data will be based on one distinct instance of the template. In most situations, this is undesirable. If unsure, please consult your Synopsys representative.

Note: `COR_AMBIT` is a legacy keyword that is obsolete and cannot be set in your job control file. It is an abbreviation for the correction ambit defined by the recipe or model file. `OVERRIDE_COR_AMBIT` should be used in place of the obsolete keyword `COR_AMBIT`.

Syntax

`OVERRIDE_COR_AMBIT` *`value`*

Options

`value`

The user-defined ambit value.

See also

OVERRIDE_HIER_AMBIT on page 170

OVERRIDE_INF_LOOP_MAX

Description

Use this keyword to control the infinite loop detection limit in corBASIC. The internal counter is incremented with every `NEXT` statement traversed in the recipe, which could result in nested `FOR` loops returning an infinite loop error slightly earlier than expected.

If your recipe is approaching the infinite loop detection limit default of 20,000,000 and you think this is the correct behavior of your recipe, you can increase this value. The maximum it can be set to is 2,147,000,000.

Syntax

`OVERRIDE_INF_LOOP_MAX` *n*

Options

n

The user-defined infinite loop detection limit. The default value is 20,000,000.

REMOVE_MKAUX_OVERLAPS

Description

Specifies that overlapping AUX polygons are removed during `UPDATE_GRAPHICS`. This is useful for commands such as `LOGVAL`, which could be affected if AUX polygons are created at the same exact location.

Syntax

`REMOVE_MKAUX_OVERLAPS ON|OFF`

Options

`ON`

Turns on removal of overlapping AUX polygons.

`OFF`

Turns off removal of overlapping AUX polygons. This is the default.

REMOVE_REFIN_CUTLINE

Description

Specifies that cutlines present in the `REF_IN` layer should be removed. This keyword is overridable in a `TEMPLATE_CALL`.

Syntax

```
REMOVE_REFIN_CUTLINE ON|OFF
```

Options

`ON`

Turns on removal of cutlines.

`OFF`

Turns off removal of cutlines. This is the default.

SNAP_45

Description

To help prevent the formation of non-45-degree edges upon snapping to the correction grid, the Proteus tool restricts endpoint locations of 45-degree edges to a fixed diamond grid. Specifying `DOUBLE` in this option uses two interlaced diamond grids and forces both vertices on diagonal lines to be snapped to the same diamond grid. This minimizes non-45-degree snapping errors in angled edges.

This function defaults to `SNAP_45 DOUBLE`.

Syntax

```
SNAP_45 SINGLE | DOUBLE
```

Options

`SINGLE`

Uses a single grid.

`DOUBLE`

Uses two interlaced diamond grids.

SPATIAL_CORRECTION_BIN_SIZE

Description

This specifies the height of the rows and width of the columns referred in `CORRECTION_ORDER`. This is applicable whenever `CORRECTION_ORDER` is set to a value that specifies correction in any spatially ordered manner such as rows or columns.

Syntax

`SPATIAL_CORRECTION_BIN_SIZE` *`size`*

Options

`size`

The height of rows or width of columns in nanometers. The default is the value of `MAX_CLUSTER` multiplied by 10.

See also

CORRECTION_ORDER on page 225

MAX_CLUSTER on page 160.

USE_APPROXIMATION_FOR_EQUALITY_OPS

Description

In corBASIC, comparison of two numbers can sometimes return unexpected results due to numerical constraints. For example, sqrt(X * X) can be unequal to X. To avoid this problem, set the global job control keyword `USE_APPROXIMATION_FOR_EQUALITY_OPS` to make all plain exact comparisons run in the approximate mode.

For example, with this keyword `ON`, the behavior of every == becomes ~==, every < becomes ~<, and so on.

You can also explicitly use the approximate version of a comparison operators in a comparison statement in corBASIC by prepending a tilde (~) to the operator. For example, ~==. See the *corBASIC Reference Manual* for details.

Syntax

`USE_APPROXIMATION_FOR_EQUALITY_OPS ON|OFF`

Options

`ON`

Causes corBASIC globally to treat all comparison operators as approximate.

`OFF`

corBASIC uses exact comparison operators. This is the default.

See also

APPROXIMATE_COMPARE_TOLERANCE on page 222

VISIBLE_CONTOUR_GRID

Description

Controls the sampling grid used by the visible model.

When `VISIBLE_CONTOUR_GRID` is not set, the default for the model is used instead, which will match previous results. You are responsible for setting this keyword to match the mask grid or visible model grid pitch if that is your intent.

The ProGen tool visible model grid pitch is controlled by the `CFG.VISIBLE_PRECISION` configuration parameter. The grid pitch is 0.5^(1+N), where N is the value of `CFG.VISIBLE_PRECISION`, which can be a value from 0 to 3.

The default `CFG.VISIBLE_PRECISION` is 0. A larger value gives a smaller pitch and a more precise contour, but also results in a slower performance and higher memory usage.

When setting `VISIBLE_CONTOUR_GRID`, keep in mind that small values for `VISIBLE_CONTOUR_GRID` can increase CPU and memory consumption.

Syntax

`VISIBLE_CONTOUR_GRID` *`value`*

Options

`value`

A double value from 0.01 to 5 corresponding to the grid size in nanometers.

WARN_HOLE_WITHOUT_PARENT

Description

This forces the generation of a warning message in cases where the keyword `ALLOW_HOLE_WITHOUT_PARENT` would otherwise suppress the notification of finding a negative polygon.

By itself, this keyword has no effect because the default behavior is to generate an error message and exit. When used along with `ALLOW_HOLE_WITH_PARENT`, a warning will be generated but execution will continue.

Syntax

```
WARN_HOLE_WITHOUT_PARENT ON|OFF
```

Options

`ON`

A warning message is generated when a hole without parent is found.

`OFF`

A warning message is not generated when a hole without parent is found. This is the default.

See also

ALLOW_HOLE_WITHOUT_PARENT on page 222

X_SIZE_FRAC

Description

This specifies the size of the X-window correction monitor. The window size is taken to be *`size`* multiplied by the vertical dimension of the connected display. The default value is 0.5. Setting this value to 0.25 cuts the size of graphics window in half.

Syntax

```
X_SIZE_FRAC size
```

Options

`size`

The size of the X-window correction monitor. Should be in the range 0.1 to 1.

6

Distributed Processing

Covers distributed processing, including sections on environment variables and configuration options, program usage, and error recovery for GDS and OASIS jobs. Provides examples at the end of the chapter.

Distributed Processing

The distributed processing option uses a DP controller/DP worker model where one DP controller maintains the list of work to be done, and *n* DP workers (accessed using the dpserver executable, on one or more hosts) asynchronously request work from, and report status to, the controller. The proteus executable is the single overall job manager, whereas dpservers request templates for correction and report their status. Each individual dpserver creates a fragment file and `LOGFILE` file, which are updated by that dpserver to eliminate file access conflicts. When correction is completed, proteus assembles the fragment files into one output file.

The general process of running a distributed correction is:

1. Run proteus.
2. Start additional dpservers to connect to the proteus DP controller, if desired.

DP controllers and DP workers communicate using sockets. Sockets are a BSD UNIX mechanism for communication between networked machines. Sockets can be created between two processes on different hosts, or between two processes on the same host.

The controller generally takes very little CPU time, so it can be run on almost any machine. If the machine it is running on is extremely busy, there can be some slight delays in answering requests from DP workers.

Post-hierarchy-management correction throughput is approximately 1/N, where N is the number of hosts correcting.

When using `PRINT`, `PRINTLINE`, and `LOGDATA` in distributed mode, output logs are created for each server used in correction. Log files specified in `PRINT` commands are flushed to disk with each new template.

Note: Distributed hierman (hierman -d) is discussed in Chapter 9, Proteus Applications.

Before You Start: DP with a Default Proteus Installation

This section outlines the proper setup of a cluster of machines for running distributed processing using a Proteus installation that has not been modified. In order to run a DP job in this fashion, the following requirements are recommended.

Additional information is provided in the *Installing Synopsys Proteus* manual.

System Requirements

All systems that run proteus should have the same setup with regard to the following:

1. *Hardware platform*

   For example, all Opterons or all Xeon 64.

2. *Operating System*

   For example, all RHEL 4.6 or all SLES 9.

3. *Mounted Disks*

   All disk locations that are used in the OPC job are accessible by all hosts and mounted so the directory paths are exactly the same. The default installation of the Proteus tool assumes that all directories seen by the system running proteus will be the same for all systems used to run dpserver. This applies to the directory where the OPC job will be run and the Proteus installation.

   For example, if two hosts, compA and compB, are used for running OPC, and compA will run proteus, the directories might be set to the following:

   - Data directory as seen from each host:

```
compA:  /remote/myOPCJob/currentJob
compB:  /remote/myOPCJob/currentJob
```

- Proteus directory (PRECIM_HOME) as seen from each host:

```
compA:  /applications/OPC/Proteus_2012.09
compB:  /applications/OPC/Proteus_2012.09
```

4. *Available TCP ports on all systems between 2346 and 2445*

   The Proteus tool and dpserver use TCP sockets to communicate status information. Some UNIX administrators turn off TCP ports for security reasons. Contact your UNIX administrator to ensure that these ports are available.

   Note: The ports used can be changed in the `DPROTEUS_CFG` file (see DPROTEUS_CFG on page 256).

5. *Remote login tools*

   Use of remote login tools rlogin, rsh, and remsh must be enabled so that users can login to all systems without providing a password. This requires setup by the UNIX administrator and the user (see User Account Requirements).

   For information on using ssh (secure shell) instead of rsh or remsh, see Using ssh on page 301.

6. *Licensing*

   All hosts must also be set up such that they can communicate with the license server(s). The tool and each dpserver process checks out licenses from the SCL (Synopsys Common Licensing) servers.

   Either the path to the license file must be the same on all systems, or you should use port@host format for a networked license server. In both cases, the `SNPSLMD_LICENSE_FILE` or `LM_LICENSE_FILE` needs to be set in the same startup as the path to the Proteus executables.

   See also License Pooling on page 247, BATCH_LICENSE_COUNT on page 272, and LICENSE_SCHEME on page 279.

7. *Memory*

   Running proteus starts on a single CPU (DP controller), then continues on to multiple CPUs (DP workers). Each host running a DP worker should have at least 4GB of memory per DP worker executing on that host. For most templates, this is enough for computing OPC. In some cases, more memory might be required.

User Account Requirements

The user's account must be set up to allow the following:

1. *Proteus tool version*

   Running of the same version of the Proteus tool from any host without using any commands on the command prompt. That is, all commands used to set up the tool are contained in your startup files upon login or shell execution. For example, if you open a shell on compB from compA using `rlogin compB`, you should be able to execute any Proteus command immediately.

2. *Remote login*

   Remote login to all machines used in OPC is enabled using rsh. Once enabled by the UNIX administrator, you might need to set up a .rhosts file. This file is placed in your home directory, and contains a list of all hosts that are “trusted” for remote logins. Place all of the hosts that will be used for OPC in this file, with one host per line.

   Sometimes additional user information is needed in this file. Consult your UNIX administrator for more information.

   Once the file is configured properly, you should be able to execute `rlogin compB` from host compA and get a command prompt on compB without entering a password.

3. *Permissions*

   You must also have the proper permissions in the directories where OPC will be run on all machines.

License Pooling

It is possible to use server pooling and multiple license servers. To do this, make sure of the following:

1. The `SCL_POOL_OPTION` environment variable is `ON` (the default). This controls the pooling functionality provided by the Synopsys Common Licensing (SCL) servers. By default, pooling is enabled for all Proteus applications.
2. Set the DP configuration file keyword `LICENSE_SCHEME` to `CLIENT` to optimize network traffic by reducing license server loading. (See LICENSE_SCHEME on page 279 for more information on this keyword.)
3. Refer to the *Synopsys Common Licensing Administration Guide* for additional information on license pooling.

Default Flow for Distributed Processing

Once the system and user environments are set up properly, using DP is simple. This section details what happens when the following example is executed.

Default Flow Example

Assume that the job file is go.pjx, compA runs proteus, and a total of three dpservers will be started, with one on compA and two on compB.

Note: The maximum number of characters allowed in a job control filename is 200.

The command needed to run the correction is:

```
proteus -s compB -s compB go.pjx
```

Although the commands are very simple, many different things are happening that are not seen by a typical user. The Proteus tool manages the entire distributed correction process by performing several functions.

1. proteus checks out application-specific licenses from the license server, as well as any other licenses the job requires for the DP controller. (See also BATCH_LICENSE_COUNT on page 272 and LICENSE_SCHEME on page 279.)
2. proteus creates the GDS or OASIS header information for the corrected output file.
3. proteus finds an open TCP port to use for communications with each dpserver process. This is usually port number 2346 if no other proteus jobs are currently running on the proteus host.
4. proteus starts dpservers on all hosts specified by the `-s` arguments. By default a dpserver is also started on the host that started proteus. Each dpserver is started by running the remote_server script that is found in $PRECIM_HOME/bin/remote_server. For more details about the default script, see remote_server on page 299.
5. After each dpserver is started by proteus, it checks out an application DP license or DS license (or other licenses the job requires for the DP worker) and alerts proteus of its existence on the TCP port specified. Once that communication is started, proteus then tells each server which template or group of templates to work on.
6. Each dpserver creates its own GDS or OASIS fragment file that contains the OPC output for all templates that are processed by that server.
7. Once the server fragment file reaches the limit set by the DP configuration parameter `CHECKPOINT_BYTES` or correction reaches an automatic checkpoint for any dpserver, the GDS or OASIS fragment file is appended to the OPC output file maintained by proteus. If the dpserver must work on additional templates, a new fragment file is started.
8. After all correction fragments are collected by proteus and added to the OPC output file, some additional work is done by proteus to complete the output file, stop the dpservers, and close the TCP sockets. The correction is now complete.

Starting the DP controller using proteus

To start a DP controller using proteus:

1. Locate the host on which to run. The hosts for correction should all be NFS-mounted to the same disk space, and paths used in the job control are the same on all hosts. For example, if `BASEPATH` is set to `/home/masks/project1/tempfiles/`, that path must be accessible from all hosts. For example, set `BASEPATH` to `./tempfiles/` and run corrections from `/home/mydir/project1`.
2. Go to the directory containing the job files. You must have a .rhosts file to allow proteus and remote_server to use a remote shell command without login prompts, which would stop execution.
3. Start proteus from the command line by entering **proteus test.pjx**.

Note the following:

- Unless disabled in the configuration file (with `NO_LOCAL_SERVER`), the DP controller (proteus) automatically starts a DP worker (dpserver) on the current host using the remote_server mechanism.
- proteus must be started before any dpserver.
- The DP controller (proteus) passes remaining work to DP workers (dpservers) as long as there are dpservers connected. Additional dpservers can be added at any time.
- If more than one DP controller is to be started on the same host, the additional proteus calls must be directed to use a different port number. This can be done automatically using the `START_PORT` and `END_PORT` parameters in the configuration file.
- When all DP workers (dpservers) become disconnected, due to either normal operations or error, the DP controller (proteus) checks to see if any templates are left for correction. If any are left, proteus reports these and exits with an error. Failed templates are recorded in a file named after the job control file appended with .failed. (For example, test.pjx would generate test.pjx.failed.)
- The DP controller (proteus) assembles all fragments into the final output file using concurrent buildgds as the correction progresses.
- If successful, the fragments are removed unless the `NO_FRAG_CLEAN` keyword is present.

- You can use both `DPSERVER_HEARTBEAT_TIME` and `DPSERVER_HEARTBEAT_TIMEOUT` to define connection times and time outs for the DP controller.
- You must have a .rhosts file to allow proteus and remote_server to use a remote shell command without login prompts, which would stop execution.
- The DP controller is not responsible for processing (correcting) templates. If all the available dpservers fail, there is no way to complete the templates in the current run, and there will be failed/untouched templates.

Starting DP Workers Individually Using dpserver

There are two methods for starting DP workers individually.

To start an individual DP worker:

1. Locate a host on which to run. Hosts for correction should all be NFS-mounted to the same disk space, and paths used in the job control must be accessible from all hosts.
2. Go to the desired host by `telnet` or `rlogin`, or go to the console.
3. Go to the directory containing the job files.
4. Start dpserver with the indicated DP controller (proteus) host and port (if different from the default) such as, **dpserver -c hostA -p 2347**. Note that if the proteus run resides on the same machine, the host argument is not required.
5. Start additional dpservers in the same manner. Multiple dpservers can be started from the same terminal window, but that can make it difficult to monitor the progress.

Alternatively, you could start an individual DP worker this way:

1. Locate a host on which to run. As stated above, hosts for correction should all be NFS-mounted to the same disk space, and paths used in the job control must be accessible from all hosts.
2. Go to the host and directory from which the DP controller (proteus) was started.
3. Start the DP worker using the remote_server shell script. All arguments are required by remote_server. See remote_server on page 299 for details.

4. Start additional DP workers (dpservers) in the same manner. Multiple DP workers can be started from the same terminal window, but that can make it difficult to monitor the progress.

Starting DP Workers Individually Using Grid Controls

In addition to the methods described in the previous section, the Proteus tool provides two utilities that you can use to start DP workers as batch jobs: Load Sharing Facility (LSF) and the OpenGrid Scheduler/Grid Engine (OGS/GE).

LSF and OGS/GE can manage remote and distributed execution of large numbers of standalone, parallel, or interactive user jobs, as well as the allocation of distributed resources such as processors or disk space.

Support for LSF and OGS/GE is accomplished through a set of scripts.

Load-Sharing Facility (LSF) Scripts

This section describes how to set up and run the Load Sharing Facility (LSF) scripts `bhierman` and `bproteus`.

Setting Up the LSF Scripts

To set up the LSF scripts, follow these steps:

Note: Before attempting to use the LSF scripts, set the `lsf_cluster_name` environment variable to point to the name of your cluster or cell.

1. Copy `$PROTEUS_APPS_HOME/tools/lsfForProteus` to an executable location in your path.
2. Add this directory to your `$PATH`.
3. Be sure to source your LSF initialization file (for example, `cshrc.lsf`) in your `.cshrc`.
4. Log on to an LSF submit machine.
5. Call the desired script: `bhierman` or `bproteus`.

Note: By default, `proteus` runs the `hierman` binary. If you have already run `hierman` before running `proteus`, you can include the `bproteus -g` command line option to explicitly direct `proteus` not to rerun `hierman`.

bhierman

This script submits a batch job to run the `hierman` binary on a LSF (Load Sharing Facility) machine.

`bhierman [-`*`options`*`] `*`job_control_file`*

where:

- *`job_control_file`* is the output of xmscript.

The available command-line options are:

`-h`	Help. This option displays the text help on bhierman.
`-q` *`farm_queue`*	The name of the queue to submit the job to. Default: bnormal.
`-R` *`grid_reqst`*	Farm specifications such as memory, cpu type, etc. The default is arch==glinux.
`-V` *`level`*	A number from 0 to 5 indicating the verbosity of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. The default is 3, where all normal output is printed.

The `bhierman` script creates a log file in the current working directory, named according to the pattern "`runhierman.<process IDnum>.log`." For example:

```
runhierman.18171.log
```

If you call the script on multiple CPUs, each CPU will generate a log in the current working directory.

bproteus

This script submits a batch job to run the `proteus` binary with a specified number of dpservers on a LSF (Load Sharing Facility) machine.

`bproteus [-`*`options`*`] `*`job_control_file`*

where:

- *job_control_file* is the output of xmscript.

The available command-line options are:

`-f`	Run the distributed job in recovery mode.
`-g` *DP_config_parms*	Set DP configuration parameters. If you have run `hierman` prior to running `proteus`, you can include the option "`-g NO_RERUN_HIERMAN ON`" to prevent `proteus` from running `hierman` again.
`-h`	Help. This option displays the text help on bproteus.
`-p` *port*	Specify the port number.
`-q` *farm_queue*	The name of the queue to submit the job to. Default: bnormal.
`-R` *grid_reqst*	Farm specifications such as memory, cpu type, etc. The default is arch==glinux.
`-s` *num_servers*	The number of servers requested for the proteus distributed job.
`-t [TPL#1 TPL#2 | list_file]`	Run DP repair flow.
`-V` *level*	A number from 0 to 5 indicating the verbosity of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. The default is 3, where all normal output is printed.

The `bproteus` script creates a log file in the current working directory, named according to the pattern "`runproteus.<process IDnum>.log`." For example:

```
runproteus.18171.log
```

If you call the script on multiple CPUs, each CPU will generate a log in the current working directory.

OpenGridScheduler/Grid Engine (OGS/GE) Scripts

This section describes how to set up and run the OpenGrid Scheduler/Grid Engine (OGS/GE) scripts `qhierman` and `qproteus`.

Setting Up the OGS/GE Scripts

To set up the OGS/GE scripts, follow these steps:

Note: Before attempting to use the OGS/GE scripts, set the `sge_cluster_name` environment variable to point to the name of your cluster or cell.

1. Copy `$PROTEUS_APPS_HOME/tools/sgeForProteus` to an executable location in your path.
2. Add this directory to your `$PATH`.
3. Be sure to source your OGS/GE initialization file (for example, `cshrc.sge`) in your `.cshrc`.
4. Log on to an OGS/GE submit machine.
5. Call the desired script: `qhierman` or `qproteus`.

   Note: By default, `proteus` runs the `hierman` binary. If you have already run `hierman` before running `proteus`, you can include the `qproteus -g` command line option to explicitly direct `proteus` not to rerun `hierman`.

qhierman

This script submits a batch job to run the `hierman` binary on an OGS/GE (OpenGrid Scheduler/Grid Engine) machine.

`qhierman [-`*`options`*`] `*`job_control_file`*

where:

- *`job_control_file`* is the output of xmscript.

The available command-line options are:

`-h`	Help. This option displays the text help on bhierman.
`-l` *`grid_reqst`*	Farm specifications such as memory, cpu type, etc.
`-V` *`level`*	A number from 0 to 5 indicating the verbosity of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. The default is 3, where all normal output is printed.

The `qhierman` script creates a log file in the current working directory, named according to the pattern "`runhierman.<process IDnum>.log`." For example:

```
runhierman.18171.log
```

If you call the script on multiple CPUs, each CPU will generate a log in the current working directory.

qproteus

This script submits a batch job to run the `proteus` binary with a specified number of dpservers on an OGS/GE (OpenGrid Scheduler/Grid Engine) machine.

`qproteus [-options] job_control_file`

where:

- *`job_control_file`* is the output of xmscript.

The available command-line options are:

`-f`	Run the distributed job in recovery mode.
`-g` *`DP_config_parms`*	Set DP configuration parameters. If you have run `hierman` prior to running `proteus`, you can include the option "`-g NO_RERUN_HIERMAN ON`" to prevent `proteus` from running `hierman` again.
`-h`	Help. This option displays the text help on qproteus.
`-l` *`grid_reqst`*	Farm specifications such as memory, cpu type, etc. The default is arch==glinux 0.
`-p` *`port`*	Specify the port number.
`-s` *`num_servers`*	The number of servers requested for the proteus distributed job.
`-t [TPL#1 TPL#2 | list_file]`	Run DP repair flow.
`-V` *`level`*	A number from 0 to 5 indicating the verbosity of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. The default is 3, where all normal output is printed.

The `qproteus` script creates a log file in the current working directory, named according to the pattern "`runproteus.<process IDnum>.log`." For example:

```
runproteus.18171.log
```

If you call the script on multiple CPUs, each CPU will generate a log in the current working directory.

Environment Variables and Configuration Options

The configuration file contains a number of parameters used by the programs in the distributed environment to configure behavior. (See the *Proteus User Guide* for information on setting environment variables.)

DPROTEUS_CFG

Description

The `DPROTEUS_CFG` environment variable must be set to indicate your dproteus configuration file, such as **setenv DPROTEUS_CFG $PRECIM_HOME/bin/dproteus.cfg**. The dproteus configuration file set for the DP controller is then used for all DP workers.

However, if you are using the `-s` option with the `proteus` command, the `DPROTEUS_CFG` environment variable is not propagated to the dpservers from proteus' shell; each dpserver must have `DPROTEUS_CFG` set in its .cshrc file. In other words, when using the `-s` option, if you manually perform a `setenv` to define `DPROTEUS_CFG`, dpserver does not obtain this information automatically. The `setenv` must be in the .cshrc file (or .profile for korn shell or bourne shell). `proteus` and dpservers should read the same configuration file. (See proteus for more information on the `-s` option.)

See also

Configuration File Keywords on page 269

REPORT_PARAMETERS on page 209

JCL_PASSWD_FILE

Description

The `JCL_PASSWD_FILE` environment variable sets the path to a password file. A password file is used when generating encrypted sections using xmscript and when reading password-encrypted sections using other applications.

See also

#ENCRYPT #END_ENCRYPT on page 15

NO_RERUN_HIERMAN

Description

`NO_RERUN_HIERMAN` (`OFF` by default) detects when proteus needs to run hierman portions and when it can keep them from the previous run.

When `NO_RERUN_HIERMAN` is in the DP configuration file, proteus attempts to rerun hierman front-end (FE) and back-end (BE) processing. If it finds that hierman FE has already been done but BE has not, hierman FE will be reused and hierman BE will be executed prior to correction.

If neither hierman FE nor BE has been done, both are automatically executed.

Note: `NO_RERUN_HIERMAN` is not supported with concurrent pipelining because hierman cannot run on a `PROTEUS_JOB_FLOW` recipe if it contains one of the concurrent mode settings (`PIPELINE_STRATEGY FRONT_LOAD` or `BACK_LOAD`).

PRECIM_HOME

Description

The `PRECIM_HOME` environment variable defines the default location for .pcm, .hlp, and library files.

Example

```
setenv PRECIM_HOME install_dir
```

PROTEUS_LICENSE_QUEUE

Description

The `PROTEUS_LICENSE_QUEUE` environment variable enables license queuing. Queuing is most useful if you have all keys on a single license server.

You must also specify the environment variable `SNPSLMD_QUEUE` when using `PROTEUS_LICENSE_QUEUE`. Both variables are `OFF` by default.

Example

```
setenv PROTEUS_LICENSE_QUEUE minutes
setenv SNPSLMD_QUEUE
```

Arguments

`minutes`

The number of minutes until the application times out. The default is 10 if *`minutes`* is not specified.

See also

DPCLIENT_START_TIMEOUT on page 275

SNPSLMD_QUEUE on page 259

REMOTE_SERVER_PATH

Description

The `REMOTE_SERVER_PATH` configuration file keyword defines the location of the remote_server executable file. When defined by path only, you must write the trailing "/".

Examples

```
REMOTE_SERVER_PATH /remote/test/here/
```

or

```
REMOTE_SERVER_PATH /remote/test/here/remote_server
```

SNPSLMD_QUEUE

Description

The `SNPSLMD_QUEUE` environment variable must be specified when using the `PROTEUS_LICENSE_QUEUE` environment variable. This variable is `OFF` by default.

See also

PROTEUS_LICENSE_QUEUE on page 258

Job Control Keywords

The following keywords can be placed in your job control file to optimize your distributed processing job.

CHECKOUT_LICENSE

Description

Use this keyword to checkout a specific license before running distributed processing.

You do not need to use the `CHECKOUT_LICENSE` keyword if OPC is the only operation performed by the recipe. However, you must include the `CHECKOUT_LICENSE` keyword in each template block that uses a particular licensed feature, such as LRC, DPT, or RBAF. If you attempt to use a licensed feature without including the `CHECKOUT_LICENSE` keyword with the correct license name in the argument, an error appears. For example:

```
--- An error has occurred. ---
Program: dpserver, built Jul 22 2009
Module:  stack_execute.c, line 10720
Error: Line 32708:   CREATE_MEDIAL_AXIS(targetType,1,BYREF
skelNodeGA(),
BYREF skelBranchGA(), BYREF skelPointGA())
  Failed to verify LRC license.
```

Syntax

```
CHECKOUT_LICENSE "license_name_string1"
   ["license_name_string2" ... "license_name_stringn"]
```

Options

`license_name_string`

A valid license name, such as “OPC”, contained within quotation marks. Valid license names can vary from release to release and should be confirmed with your Synopsys account team.

Multiple license names, separated by a space, can be specified in one `CHECKOUT_LICENSE` command.

See also

The *Proteus Release Notes* and the *Installing Synopsys Proteus* manual.

NEW_TC_SPECIFIC_MODEL_LOADING

Description

When you include this keyword, the Proteus tool reads only the models that are necessary for the `TEMPLATE_BLOCK` that is being run currently. This can result in memory reduction in recipes where multiple `TEMPLATE_BLOCK`s use Simulators.

Syntax

```
NEW_TC_SPECIFIC_MODEL_LOADING [ON|OFF]
```

Options

`ON`

Read only the models that are necessary for the current `TEMPLATE_BLOCK`.

`OFF`

Read models for all `TEMPLATE_BLOCK`s. This is the default.

NO_FRAG_CLEAN

Description

Fragments are written by DP workers to a directory that is normally deleted upon completion. When `NO_FRAG_CLEAN` is present in the job control file, concurrent buildgds does not automatically delete DP worker fragments from the output file directory.

Syntax

```
NO_FRAG_CLEAN
```

See also

concurrent buildgds on page 292

PIPELINE_STRATEGY

Description

Note: Not all of the `PIPELINE_STRATEGY` options are supported with `COMPACT_CONTEXT`, `CORRECTION_ORDER`, `NEW_OVERFLOW_CELL`, `NO_RERUN_HIERMAN`, `PROTEUS_EXCHANGE`, and `TEMPLATE_HASH_VERIFICATION`. See the documentation in this manual for each of these keywords for further information.

`PIPELINE_STRATEGY` enables concurrent pipelining. Concurrent pipeline strategies allow templates to be corrected from multiple template blocks in a parallel, concurrent, and pipelined manner.

Concurrent pipeline strategy (for all modes except `SINGLETONS`) is based on the premise that the correction order will proceed in rows up the chip (these rows are also called stripes). Priority is opportunistic and will depend on a variety of factors, including resource availability, individual template processing time for a stage, variability of individual template processing times per stage, and so forth.

For log file examples showing the different `PIPELINE_STRATEGY` options, see Appendix C, Log Files.

Syntax

```
PIPELINE_STRATEGY OFF|FRONT_LOAD|BACK_LOAD|SINGLETONS
```

Options

`OFF`

Concurrency is disabled.

`FRONT_LOAD`

Concurrency is enabled allowing stages to execute concurrently, gives priority to correcting earlier stages. `FRONT_LOAD` will wait until the existing stage frees up servers to assign to a later stage.

FRONT_LOAD is intended to provide maximum utilization of the servers to reduce overall TAT. This will keep the priority on the current stage, allowing later stages to start when servers are freed up keeping resources maximally utilized.

BACK_LOAD

Concurrency is enabled allowing stages to execute concurrently, gives priority to correcting later stages. BACK_LOAD will begin processing a later stage as soon as it is available sending all available servers to the later stage.

BACK_LOAD is intended to get some partial output from the final stages as soon as possible by sending all servers to the later stage. This switching of servers may reduce utilization a bit more than FRONT_LOAD, but has the advantage of enabling final output to be produced quickly. BACK_LOAD might be well applied to verification of the final stage to get final output early and discover problems before the entire job is complete.

SINGLETONS

In this mode some templates for a stage may be quickly discovered to get servers started correcting early while full context analysis completes, extending the distributable portion of the entire job.

PROTEUS_EXCHANGE

Note: The presence of an OUTPUT CATS2 section takes precedence over the PROTEUS_EXCHANGE keyword. If the OUTPUT CATS2 section is not present and PROTEUS_EXCHANGE CATS2 is present, the older PCX2 behavior is retained.

Description

When PROTEUS_EXCHANGE is in the job control file, the Proteus-CATS interface is enabled. Detailed information about this interface is provided in the CATS reference documentation.

When using PROTEUS_EXCHANGE CATS2 (PCX2), the automatic checkpoints change to the following percentages of area: 25%, 50%, 75%, 85%, 95%, 98%, and no minimum fragment file size threshold is used at the 25%, 50%, 75%, 90%, and 100% completion marks.

When using PCX2 from internal TEMPLATE_BLOCKs or with multiple output files, the presence of an OUTPUT CATS2 section is required.

Syntax

`PROTEUS_EXCHANGE CATS2`

Options

`CATS2`

The following are enforced in order to enable CATS to work in parallel with the tool:

- `NO_FRAG_CLEAN`
- `RETRY_FAILED_IMMEDIATELY`
- `CORRECTION_ORDER`: This keyword is automatically set to `ROWS`. Attempts to specify the DP correction order in the job control, with `-r` proteus switch, will be ignored.
- `CONCURRENT_MKTOP`: The mktop utility is executed as a thread on the machine where proteus runs.

See also

CORRECTION_ORDER on page 225

OUTPUT and END_OUTPUT on page 64

SERVER_UTILIZATION_LOG

Description

Use this keyword to create an output file showing server utilization. If `SERVER_UTILIZATION_LOG` is not specified (the default), the log is not created.

Syntax

`SERVER_UTILIZATION_LOG` *`filename`*

Options

`filename`

The name of the output log.

Example

The following is an example of a server utilization log. The `SERVER_UTILIZATION_RATE` was set to 2 in this case.

```
Time            Idle   TC1   TC2   TC3   TC4
15.00000000000   0      0     0     0     0
27.00000000000   0      1     0     0     0
29.00000000000   0      2     0     0     0
31.00000000000   0      2     0     0     0
33.00000000000   0      2     0     0     0
35.00000000000   0      2     0     0     0
```

The columns in the log are:

- Time (The number of seconds since the beginning of the log.)

  The logging frequency might not exactly match the value of `SERVER_UTILIZATION_RATE` because server information is sent to the log at the specified frequency only during `TEMPLATE_CALL` operations, and during hierman back-end processing between `TEMPLATE_CALL`s, the log is not updated.

- Idle (The number of idle servers.)

  All servers are engaged and included in the log, but then stay idle during hierman back-end processing. Logging stops during this period and resumes when the next `TEMPLATE_CALL` becomes active. This is particularly noticeable for serial pipeline modes.

- TC*n* (`TEMPLATE_CALL` number)

  The number in this column indicates the number of servers working on a particular `TEMPLATE_CALL`. For example, at the time stamp for 33 seconds, there are 2 servers working on TC1, none on TC2, none on TC3, and none on TC4.

See also

SERVER_UTILIZATION_RATE on page 264

SERVER_UTILIZATION_RATE

Description

Use this keyword to change the number of entries in the log file associated with `SERVER_UTILIZATION_LOG`. If the value of *n* is low (1, for example) the log

file will be very large but the data will be high-resolution. If *n* is large (300, or 5 minutes) the data volume will be much lower but the log file will be small.

Syntax

`SERVER_UTILIZATION_RATE` *n*

Options

n

The number of seconds between each time server (DP worker) utilization information is sent to the `SERVER_UTILIZATION_LOG`. If not set, the default is 20 if `SERVER_UTILIZATION_LOG` is specified.

See also

SERVER_UTILIZATION_LOG on page 263

STRIPE_HEIGHT

Description

This keyword allows you to directly specify the height (in nanometers) of a stripe used to group templates for processing, when used with `PIPELINE_STRATEGY FRONT_LOAD` or `BACK_LOAD`.

Stripes are assigned from the bottom of the original chip extent and proceed up the chip in `STRIPE_HEIGHT` intervals. On subsequent template calls, the stripe positioning (bottom and top stripe edges) will match the extent used for the first template call.

However, in case of chip growth near the top and/or bottom of the chip, the top and bottom stripes will adjust their height as needed. Specifically, the top-most row will extend to the top of the current template call's chip height. In the same way, the bottom-most stripe will extend to the bottom of the current template call's chip extent.

As a chip moves through pipeline processing, the actual chip extent may grow if additional graphics are created. Thus the top-most and bottom-most rows may be larger than `STRIPE_HEIGHT`.

Syntax

`STRIPE_HEIGHT` *height*

Options

height

Any non-negative integer. The default is 0.

When STRIPE_HEIGHT is set to 0 or is not used, the actual number of stripes used is set to chip height divided by default stripe height (10 * MAX_CLUSTER). In each template call, the actual stripe height is recomputed using the template call's chip height divided by the computed number of rows.

The minimum computed stripe count is 3. If the computed stripe count is less than 3, the Proteus tool issues a warning message and discards the specified STRIPE_HEIGHT value.

See also

PIPELINE_STRATEGY on page 261

SYNCHRONIZE

Description

SYNCHRONIZE is one of the optional parameters to TEMPLATE_CALL. It enables you to control a portion of the concurrent pipeline, in effect forcing certain template calls to not run concurrently with each other, while the rest of the template calls continue to run concurrently.

This is accomplished by creating a checkpoint in the pipeline flow between two template calls that will not permit the second of the two template calls to begin executing until the first template call has completed.

When you use PIPELINE_STRATEGY FRONT_LOAD or BACK_LOAD, the template block that requires synchronization (all previous template block work to be completed or all future template block work to wait until it has completed), must include a SYNCHRONIZE statement.

The default is NONE.

Syntax

```
SYNCHRONIZE ALL|BEGIN|END|NONE
```

Options

ALL

Indicates that the template block cannot be concurrently pipelined. Equivalent to SYNCHRONIZE BEGIN and SYNCHRONIZE END.

`BEGIN`

Indicates that the hierman back end (and thus template processing) for this template block may not start until the previous template block is complete (so the beginning of this template call must be synchronized to the completion of the previous template block). This option is for when the template block must have all data (templates) ready before it can start.

`END`

Indicates that the next template block may not start until this template block has completed. The start of the next template block must be synchronized to the completion of this template block.

`NONE`

Indicates that there are no restrictions on the template call fully participating in the concurrent pipeline; meaning `PIPELINE_STRATEGY FRONT_LOAD` or `BACK_LOAD`. This is equivalent to not specifying a value for `SYNCHRONIZE` at all, or not including the `SYNCHRONIZE` option.

TEMPLATE_HASH_VERIFICATION

Description

When this keyword is present, template main and context graphics are validated before a job is executed.

Note: `TEMPLATE_HASH_VERIFICATION ON` and `HIERMAN` options are not supported with `COMPACT_CONTEXT` or concurrent pipelining (`PIPELINE_STRATEGY FRONT_LOAD` or `BACK_LOAD`).

During correction in either single-CPU or distributed mode, `TEMPLATE_HASH_VERIFICATION` compares the template main and context graphics defined by hierman with the main and context graphics loaded into the correction engine. If a mismatch is found in single-CPU mode, the job is terminated with an error message. If a mismatch is found in distributed mode, the DP controller (client) prints a warning message: `Client: Validation for template xxx  failed on server xxx ( it has failed for x times)...`

The template is then assigned to another server for recorrection.

The DP configuration file parameter `MAX_HASH_FAILURE` controls the maximum number of allowed mismatches before a template fails, according to

the following rules:

- If `MAX_HASH_FAILURE` is greater than `RETRY_FAILED_COUNT`, `MAX_HASH_FAILURE` is ignored and the `RETRY_FAILED_COUNT` value is used.
- If `MAX_HASH_FAILURE` is smaller than `RETRY_FAILED_COUNT`, `MAX_HASH_FAILURE` is used.
- If `MAX_HASH_FAILURE` is not set, the value of `RETRY_FAILED_COUNT` is used.
- If neither `MAX_HASH_FAILURE` or `RETRY_FAILED_COUNT` is set, `MAX_HASH_FAILURE` defaults to 5.

Note: In `CLUSTER NONE` mode, or `CLUSTER FLAT` mode with `NEW_SMART_BLOCK_COMPRESSION OFF`, context analysis is not performed, and thus hash verification can only be done by the `SERVER` option. For example, proteus issues a warning such as the following:

```
WARNING: CLUSTER FLAT is used with NEW_SMART_BLOCK_COMPRESSION
OFF. Forcing TEMPLATE_HASH_VERIFICATION SERVER.
```

Syntax

`TEMPLATE_HASH_VERIFICATION OFF|HIERMAN|SERVER|ON`

Options

`OFF`

Turns off validation of template main and context graphics. This is the default.

`HIERMAN`

Turns on validation of template main and context graphics. This method compares the hash numbers between the hierman run and the correction run. During context analysis, the hash numbers of .main and .context are stored in the .THV hierman file. If a mismatch is found during the correction run, a warning message is issued, and then this correction template is sent for re-correction on a different server.

`SERVER`

Executes validation of template main and context graphics between servers. This method has two different servers compute the hash number of the same correction template. If a mismatch is found, a warning message is issued, and then this correction template is sent for recorrection on a different server.

By default, five (5) servers (all if the total is less than five) start with hash jobs and all other servers start with correction jobs. Once hash job servers have finished hash calculation jobs, they work on correction jobs.

`ON`

For backward compatibility. Behaves exactly the same as `TEMPLATE_HASH_VERIFICATION HIERMAN.`

Configuration File Keywords

You can insert the following keywords directly into the configuration file (established in the `DPROTEUS_CFG` environment variable) when using the proteus and dpserver executables.

The proteus application allows you to set or override any configuration file keyword from the command line. To do this, add the keyword to the command line. For example:

```
proteus -START_PORT 4000
```

On any given line, all characters following an apostrophe are ignored, allowing comments to be added to the configuration file.

- `ABORT_ON_PERMANENT_FAIL`
- `ALLOW_HT_SERVERS`
- `BATCH_LICENSE_COUNT`
- `CHECKPOINT_BYTES`
- `CONCURRENT_MKTOP`
- `DPCLIENT_START_TIMEOUT`
- `DPSERVER_HEARTBEAT_TIME`
- `DPSERVER_HEARTBEAT_TIMEOUT`

- EXIT_HT_SERVERS
- FAILED_SERVER_REMOVE_FLAG
- HIERMAN_FILE_INTEGRITY_CHECK
- HIERMAN_FILE_INTEGRITY_CHECK_RETRY_PERIOD
- HIERMAN_VERSION_CHECK
- KEEP_INACTIVE_SERVERS
- LICENSE_SCHEME
- MAX_BURST_COUNT
- MAX_BURST_TIME
- MAX_HASH_JOB
- MAX_HASH_FAILURE
- MAX_SYNC_ERR_RETRIES
- NO_LOCAL_SERVER
- NO_SVR_VERSION_CHECK
- NO_TPL_TIMESTAMP_CHECK
- REPORT_JOB_STATISTICS
- RETRY_FAILED_COUNT
- RETRY_FAILED_IMMEDIATELY
- SERVER_DIE_ON_FAIL
- SERVER_START_DELAY
- SERVER_WAIT
- START_PORT and END_PORT
- USE_APPLICATION_DP
- USE_COMMON_DP
- USE_PIPELINE_DP
- USE_FFLUSH

Note: The Proteus DPT decomposition recipe makes use of an additional DPT_IO DP configuration file keyword and related parameters. The keyword and parameters are described in the *Proteus™ Double Patterning Technology User Guide*.

ABORT_ON_PERMANENT_FAIL

Description

Use this parameter to abort the job when a template has exceeded `RETRY_FAILED_COUNT`.

Syntax

```
ABORT_ON_PERMANENT_FAIL [ON|OFF]
```

Options

`ON`

Job will be aborted when a template has exceeded `RETRY_FAILED_COUNT`.

`OFF`

Job will not be aborted.

ALLOW_HT_SERVERS

Description

Use this keyword to start hyper-threading dpservers automatically. Hyper-threading servers enable you to improve performance by taking advantage of Intel HT (hyper-threading) technology,

When `ALLOW_HT_SERVERS` is `ON`, a hyper-threading dpserver is started on the same host whenever a normal dpserver is started. (You must configure this keyword `ON` for each Proteus job.)

Alternatively, you can start a hyper-threading dpserver manually by including the `-ht` option on the `proteus` command line.

There are several main differences between hyper-threading and normal dpservers:

- A normal dpserver requires distributed processing (DP) licenses for each worker. A hyper-threading dpserver does not consume a license.
- A normal dpserver is always available to perform template processing. A hyper-threading dpserver is available for template processing only when the following conditions are true:
  - All physical cores on the host are busy.
  - Hyper-threading is enabled on the designated host.

- Each hyper-threading dpserver is paired with a normal dpserver.

- Because a hyper-threading host can run multiple dpserver workers, it may consume additional memory.

Syntax

`ALLOW_HT_SERVERS [ON|OFF]`

Options

`ON`

Start hyper-threading dpservers automatically.

`OFF`

Do not start hyper-threading dpservers automatically. This is the default.

See also

EXIT_HT_SERVERS on page 276

proteus on page 387

BATCH_LICENSE_COUNT

Description

The `BATCH_LICENSE_COUNT` configuration file keyword defines the number of licenses that are in a batch of licenses checked out when `LICENSE_SCHEME` is set to `CLIENT`.

Once a DP worker connects, the DP controller checks out licenses in a batch. If the number of DP workers *does not* exceed the value of `BATCH_LICENSE_COUNT`, no more licenses are checked out. If the number of DP workers *does* exceed the value of `BATCH_LICENSE_COUNT`, the DP controller checks out another batch of licenses until all connecting DP workers are either properly licensed or turned down because all licenses on the server are used.

In the case where more licenses are checked out than are actually needed, unused licenses are checked back in after 5 minutes.

Syntax

`BATCH_LICENSE_COUNT` *n*

Options

n

The number of licenses checked out per batch. The default is 10. It is inefficient to set this value too large, due to the excess work needed to check the unused licenses back in.

See also

LICENSE_SCHEME on page 279

CHECKPOINT_BYTES

Description

Use this parameter to adjust the approximate desired size of the fragment files. The checkpoint decision is based on the total size of both fragment and database files. Larger numbers cause less frequent checkpoints and larger fragment files, while smaller numbers cause more frequent checkpoints and smaller fragment files.

Using `CHECKPOINT_BYTES` allows the Proteus tool to produce intermediate fragment files before all templates are finished. This enables final file output to begin during processing, thus conserving time. Concurrent buildgds can begin assembling the final output as soon as it receives the first checkpoint.

During operation, the tool determines when the fragment file exceeds the requested value of `CHECKPOINT_BYTES`, and finishes the template or burst templates that are currently in progress before checkpointing. For this reason, the fragment file sizes might not match the value specified in `CHECKPOINT_BYTES`.

The value of `CHECKPOINT_BYTES` found in the DP configuration file is used for the last `TEMPLATE_BLOCK`. If the last `TEMPLATE_BLOCK` is GDSII, intermediate `TEMPLATE_BLOCK`s use the user-defined value divided by 8; otherwise, all `TEMPLATE_BLOCK`s use the `CHECKPOINT_BYTES` value set by the user.

Regardless of whether the value of `CHECKPOINT_BYTES` is reached, and as long as the combined size of the fragment files and database files is larger than 10 megabytes, the DP worker sends a fragment file to concurrent buildgds randomly within the 12.5% to 37.5% of area range. The following additional checkpoints occur between 37.5%-62.5%, 62.5%-87.5%, and 87.5%-97.5% of area ranges.

Use caution when setting `CHECKPOINT_BYTES` because it can dramatically increase the number of temporary files stored in the directory containing the DP workers' output. (This directory name is a concatenation of the name of the output file and the .dir extension; output.oas.dir, for example.) A maximum of 1,000 files per directory in a UNIX file structure is recommended. Consider setting the value so that this maximum is not exceeded. This value is dependent on the number of DP workers operating in a job, because each DP worker maintains its own fragment files.

Syntax

`CHECKPOINT_BYTES` *n*

Options

n

The desired size of the fragment files, any value greater than or equal to 1. The default is 100 megabytes for GDSII and 12.5 megabytes for OASIS.

Example

The following line in the configuration file causes the tool to set checkpointing at 50 megabytes:

```
CHECKPOINT_BYTES 50000000
```

CONCURRENT_MKTOP

Description

The mktop utility can be run as a thread on the machine where proteus runs. Such machines typically have larger memories than the cluster nodes, and thus can reduce the risk of mktop performance problems.

The Proteus tool can start mktop in a thread when it is starting dpservers. This ensures that the mktop process does not become a serial addition. If the hardware on which proteus is running has enough cycles to process the proteus, concurrent, and mktop threads, TAT (turnaround time) should not be affected.

To turn this ability on, either put `CONCURRENT_MKTOP` in your dproteus.cfg file or use the proteus command-line switch, -m. The default is not to run mktop on the proteus machine.

DPCLIENT_START_TIMEOUT

Description

This defines the time the DP controller (proteus) should wait between starting the last DP worker (dpserver) and waiting for the first DP worker to connect.

If the environment variable `PROTEUS_LICENSE_QUEUE` is set to a value greater than the value of `DPCLIENT_START_TIMEOUT`, proteus uses the value of `PROTEUS_LICENSE_QUEUE`.

Syntax

`DPCLIENT_START_TIMEOUT` *n*

Options

n

Wait time in seconds. The default is 600 (10 minutes).

See also

PROTEUS_LICENSE_QUEUE on page 258

DPSERVER_HEARTBEAT_TIME

Description

Use `DPSERVER_HEARTBEAT_TIME` to force the DP worker (dpserver) to send messages to the DP controller at specified intervals. This validates the status of the socket connection. This should be set to a period of less than two hours, because TCP/IP might drop the socket if no communication is received within that amount of time.

The DP controller should not receive more than 1 or 2 messages per second (heartbeats). For example, for a correction job distributed to 1,000 dpservers, a heartbeat time of 1,000 seconds would cause proteus to receive one heartbeat message per second.

Syntax

`DPSERVER_HEARTBEAT_TIME` *n*

Options

n

The heartbeat time in seconds. The default is 3600.

See also

DPSERVER_HEARTBEAT_TIMEOUT on page 276

DPSERVER_HEARTBEAT_TIMEOUT

Description

Use this parameter to define the heartbeat timeout time. This command is used in conjunction with the heartbeat time. This timeout can be set such that the DP controller can remove connections to DP workers that are not communicating.

You should use a value at least 3 or 4 times the heartbeat time.

Proteus licensing separates the heartbeat and licensing functionality. If licensing locks up, the DP worker (dpserver) continues to execute normally, and heartbeats continue to be sent to the DP controller. The dpserver session only exits when the license is actually denied.

Syntax

`DPSERVER_HEARTBEAT_TIMEOUT` *n*

Options

n

The heartbeat timeout time in seconds. The default is 11160.

See also

DPSERVER_HEARTBEAT_TIME on page 275

EXIT_HT_SERVERS

Description

Set this keyword `ON` to retire the paired hyper-threading dpserver whenever a normal dpserver exits.

Syntax

`EXIT_HT_SERVERS [ON|OFF]`

Options

`ON`

Retire the paired hyper-threading dpserver when a normal dpserver exits. This is the default.

`OFF`

When a normal dpserver exits, convert the paired hyper-threading dpserver to a normal dpserver.

See also

ALLOW_HT_SERVERS on page 271

FAILED_SERVER_REMOVE_FLAG

Description

In some unusual situations when failing computer hardware is running dpservers, abnormal Proteus tool results can be created by dpservers before they fail, which is then included in the tool output. To help prevent this data from reaching the final output during hardware failure, use `FAILED_SERVER_REMOVE_FRAG` to discard results from the last fragment file created from a server that unexpectedly disconnected. The discarded results are scheduled for recorrection. In situations when servers are manually exited using `kill -9`, this might cause some recorrections.

Syntax

```
FAILED_SERVER_REMOVE_FLAG [ON|OFF]
```

Options

`ON`

Results from the last fragment file created from a server that unexpectedly disconnected are discarded.

`OFF`

Results are not discarded. This is the default.

HIERMAN_FILE_INTEGRITY_CHECK

Description

Use this keyword to perform data integrity checking whenever the Proteus hierarchy manager or dpserver is accessed.

Syntax

```
HIERMAN_FILE_INTEGRITY_CHECK ON|OFF
```

Options

`ON`

Enables data integrity checking. This is the default when `PIPELINE_STRATEGY` is `FRONT_LOAD` or `BACK_LOAD`.

`OFF`

Disables data integrity checking. This is the default when `PIPELINE_STRATEGY` is `OFF` or `SINGLETONS`.

See also

PIPELINE_STRATEGY on page 261

HIERMAN_FILE_INTEGRITY_CHECK_RETRY_PERIOD

Description

Use this keyword to specify the total time period (in seconds) that the hierarchy manager or dpserver will wait while performing data integrity checking (continuing to re-read data in hierman files until the data is no longer stale).

When running a large number of dpservers, consider increasing the wait time if the NFS server is busy and data in hierman files remains stale.

Syntax

`HIERMAN_FILE_INTEGRITY_CHECK_RETRY_PERIOD` *`wait_time`*

Options

`wait_time`

The total time (in seconds) to wait while performing the file integrity check. The default is 10 (seconds).

HIERMAN_VERSION_CHECK

Description

Use this keyword to verify compatibility between hierman and proteus. This keyword takes a single argument.

Syntax

`HIERMAN_VERSION_CHECK [STRICT|EASY|NONE]`

Options

`STRICT`

Requires the entire proteus_tag to match (version + build).

`EASY`

Requires the version (such as J-2014.06-7) to match. This is the default.

`NONE`

Does not compare versions.

KEEP_INACTIVE_SERVERS

Description

Use this parameter to define the number of inactive DP workers (dpservers) that are kept at the end of the job. If you set *n*=`0`, no inactive DP workers are kept. If you set *n* to –1, all inactive DP workers are kept. You should only set *n* to 0 or -1 in rare circumstances, because you want to keep at least one DP worker available for any failed templates. If a DP worker is killed by either the inactive or the time-out method, an explanatory message is printed to the log.

Syntax

`KEEP_INACTIVE_SERVERS` *n*

Options

n

The number of DP workers kept until the end of the job. The default value is 20.

LICENSE_SCHEME

Description

Use this parameter to control the way the licenses are checked out: either each DP worker checks out its own license, or the DP controller checks out licenses for the DP workers.

Syntax

`LICENSE_SCHEME SERVER|CLIENT`

Options

`SERVER`

Each DP worker checks out a license one at a time. This is the default.

`CLIENT`

The DP controller checks out licenses in batch mode. The number of licenses per batch is controlled by the configuration file keyword `BATCH_LICENSE_COUNT`. This setting is useful for jobs with large numbers of DP workers.

Note: Do not use the `CLIENT` setting with multiple license servers.

See also

BATCH_LICENSE_COUNT on page 272

MAX_BURST_COUNT

Description

Use this parameter to specify the number of sibling templates from the same parent cell that are sent in a burst to the same DP worker. This improves the performance by not requiring the DP worker to request each template.

Reasonable `MAX_BURST_COUNT` values are between 20 and 100. However, this value is secondary in importance to `MAX_BURST_TIME`.

Syntax

`MAX_BURST_COUNT` *n*

Options

n

The number of templates to burst. The default is 50.

See also

MAX_BURST_TIME on page 280

MAX_BURST_TIME

Description

Use this parameter to specify the maximum correction time a burst job is permitted to take, in seconds. This is an approximation, based on the time

taken from one representative template from the parent cell, multiplied by `MAX_BURST_COUNT`.

Default `MAX_BURST_COUNT` and `MAX_BURST_TIME` values should be optimal for most situations.

Syntax

`MAX_BURST_TIME` *n*

Options

n

The maximum time permitted for a burst job, in seconds. The default value is 300.

See also

MAX_BURST_COUNT on page 280

MAX_HASH_JOB

Description

When `TEMPLATE_HASH_VERIFICATION` is `ON`, this defines the maximum number of validation jobs sent in one command.

Syntax

`MAX_HASH_JOB` *n*

Options

n

The maximum number of hash jobs. The default is 50.

See also

MAX_HASH_FAILURE

TEMPLATE_HASH_VERIFICATION on page 267

MAX_HASH_FAILURE

Description

When `TEMPLATE_HASH_VERIFICATION` is `ON`, this defines the maximum number of allowed mismatches before a template fails.

Syntax

`MAX_HASH_FAILURE` *n*

Options

n

The maximum number of mismatches allowed. This value defaults to the value of `RETRY_FAILED_COUNT` if the keyword is not present in the configuration file when `TEMPLATE_HASH_VERIFICATION` is `ON`, or to 5 if the keyword is present but the value is not defined.

See also

MAX_HASH_JOB

RETRY_FAILED_COUNT on page 285

TEMPLATE_HASH_VERIFICATION on page 267

MAX_SYNC_ERR_RETRIES

Description

Use this parameter to control how many times the DP worker (dpserver) retries to synchronize (using `fdatasync`) the data to disk after a failure to do so. Such a failure to synchronize the data could be caused by a large number of DP workers synchronizing at the same time or a lack of disk space.

This algorithm prevents large numbers of repeated synchronized file write attempts, enabling more DP workers to operate together correctly. The algorithm pauses between each attempt for a maximum of (15 + (2* iteration)) seconds. If the synchronization is still not successful after the specified number of retries the server exits with an error.

Syntax

`MAX_SYNC_ERR_RETRIES` *n*

Options

n

The number of times for the DP worker to retry to synchronize the data to disk. Must be an integer greater than or equal to zero.

0 causes the DP worker not to retry and to exit immediately. The default value is 5.

NEW_FAST_RECOVERY

Description

Allows Proteus to run recovery more efficiently by skipping fragment file validation for intermediate template calls. When `NEW_FAST_RECOVERY` is `ON`, Proteus only validates fragment files for the final template call that was run in the prior run.

Syntax

```
NEW_FAST_RECOVERY [ON|OFF]
```

Options

`ON`

Validate fragment files for the final template call that was run in the prior run, but do not validate fragment files for intermediate template calls.

`OFF`

Also validate fragment files for intermediate template calls. This is the default.

NO_LOCAL_SERVER

Description

This overrides proteus's default behavior of starting a DP worker on a local host.

Syntax

```
NO_LOCAL_SERVER [ON|OFF]
```

Options

`ON`

Does not start a DP worker on the local host.

`OFF`

Starts a local DP worker on the local host. This is the default.

NO_SVR_VERSION_CHECK

Description

This overrides the default behavior of checking the Proteus tool version tag to ensure the DP worker and DP controller are using the same version.

The version tag check prevents you from unintentionally running different versions of dpserver and proteus at the same time. If they are different versions, the DP worker exits. Add the line `NO_SVR_VERSION_CHECK` to your configuration file to run different versions of proteus and dpserver together.

Syntax

```
NO_SVR_VERSION_CHECK [ON|OFF]
```

Options

`ON`

Turns on override.

`OFF`

Turns off override.

NO_TPL_TIMESTAMP_CHECK

Description

`NO_TPL_TIMESTAMP_CHECK` overrides the default behavior of version checking the log files in a distributed run by disabling the check.

The check prevents proteus and dpserver from getting different hierman files and proceeding to correction under unusual circumstances. When this check is performed, proteus compares a unique number in the hierman .STAT file with the number that dpserver sees in the .STAT file. If these do not match, an error message is printed and dpserver exits. The Proteus tool continues normally if it still has connected dpservers. The error message is as follows:

```
Error: Server is using a different version of the hierman
files
```

```
Server 1 is being forced to exit.
```

Syntax

```
NO_TPL_TIMESTAMP_CHECK [ON|OFF]
```

Options

`ON`

Turns on override.

`OFF`

Turns off override.

REPORT_JOB_STATISTICS

Description

Use this parameter to report job statistics when a distributed processing job is done. The job statistics include such information as: the number of dpservers, the total CPU time consumed by all the servers added together, the average utilization of the servers (where *utilization* is (user time + system time) / elapsed time, and the count_graphics results for every input and output file in the job.

Syntax

`REPORT_JOB_STATISTICS` *`filename`*

Options

`filename`

The name of the text file containing job statistics. If you do not specify a filename, the file `proteus_job_report.txt` will be created.

RETRY_FAILED_COUNT

Description

Use this parameter to specify the number of times a template that has failed on one DP worker is retried on other DP workers.

Syntax

`RETRY_FAILED_COUNT` *n*

Options

n

The number of times a template is retried. The default is 5.

RETRY_FAILED_IMMEDIATELY

Description

Use this parameter to force a failed template to retry immediately, rather than waiting to the end of the job.

Syntax

```
RETRY_FAILED_IMMEDIATELY [ON|OFF]
```

Options

`ON`

Failed template is immediately retried.

`OFF`

Failed template is retried at the end of the job.

REUSE_ADDRESS

Description

Use this parameter to allow the reuse of local addresses when opening sockets for connection. Note that this may cause a port collision issue if multiple Proteus jobs try to open sockets at the same time.

Syntax

```
REUSE_ADDRESS [ON|OFF]
```

Options

`ON`

Proteus allows local addresses to be reused for sockets.

`OFF`

Proteus does not allow local addresses to be reused for sockets. This is the default.

SERVER_DIE_ON_FAIL

Description

Use this parameter to force the DP worker working on the template to exit when a template suffers a failure (such as a runtime error).

Syntax

`SERVER_DIE_ON_FAIL [ON|OFF]`

Options

`ON`

DP worker exits when template fails.

`OFF`

DP worker does not exit when template fails. This is the default.

SERVER_START_DELAY

Description

Use this parameter to cause a delay between the starting of each remote_server script as indicated by each `-s` argument on the proteus command line. Unless you are using a large number of servers, you should leave this number set to the default (0.5). The delay is given in seconds.

Syntax

`SERVER_START_DELAY` *n*

Options

n

A positive floating point number. The default value is 0.5.

SERVER_WAIT

Description

Use this parameter to define the amount of time a DP worker waits for the DP controller to become available.

Syntax

`SERVER_WAIT` *n*

Options

n

The number of seconds the DP worker waits for the DP controller to become available. The default is 16.

START_PORT and END_PORT

Description

START_PORT and END_PORT indicate the range within which to search for an available port. If START_PORT and END_PORT are present and a -p port was not provided on the command line, proteus searches this range for an available port. These numbers are typically around 2300-2500. This is useful if more than one DP controller is to be run on a host, because each DP controller searches for its own port. This must be used with the -s command-line option to work well with DP workers (dpservers). The opened port is reported to the command line for manually starting DP workers.

Syntax

START_PORT *n1*
END_PORT *n2*

Options

n1

The beginning of the range within which to search for an available port.

n2

The end of the range within which to search for an available port.

USE_APPLICATION_DP

Description

Use this parameter to use the older (pre-D-2010.06-3) method of checking out server licenses (application-specific licensing).

USE_APPLICATION_DP can also be used as a job control keyword or a command-line option.

For more details, consult your Synopsys marketing and sales representative.

Syntax

USE_APPLICATION_DP [ON|OFF]

Options

ON

Uses application-specific license scheme. This is the default.

`OFF`

Uses common-DP license scheme.

See also

CHECKOUT_LICENSE on page 259

LICENSE_SCHEME on page 279

USE_COMMON_DP

Description

Use this parameter to run a multi-application recipe using non-application-specific licenses. This mode is called the common-DP mode, which checks out DS licenses instead of application-specific licenses.

The common-DP licensing mode applies only to `PROTEUS_JOB_FLOW` recipes.

`USE_COMMON_DP` can also be used as a job control keyword or a command-line option.

Note: When using common-DP mode, configure your license servers with an even number of DS licenses. In common-DP mode, each DP server checks out two DS licenses – both of them from the same license server. When a license server has an odd number of licenses, one DS license remains unused.

For more details, consult your Synopsys marketing and sales representative.

Syntax

```
USE_COMMON_DP [ON|OFF]
```

Options

`ON`

Uses common-DP license scheme.

`OFF`

Uses application-specific license scheme. This is the default.

See also

CHECKOUT_LICENSE on page 259

LICENSE_SCHEME on page 279

USE_PIPELINE_DP

Description

Use this parameter to run a multi-application recipe with just one PIPELINE_DP license.

The PIPELINE_DP licensing mode applies only to `PROTEUS_JOB_FLOW` recipes.

`USE_PIPELINE_DP` can also be used as a job control keyword or a command-line option.

For more details, consult your Synopsys marketing and sales representative.

Syntax

```
USE_PIPELINE_DP [ON|OFF]
```

Options

`ON`

Uses PIPELINE_DP license scheme.

`OFF`

Uses application-specific license scheme. This is the default.

See also

CHECKOUT_LICENSE on page 259

LICENSE_SCHEME on page 279

USE_FFLUSH

Description

When this keyword is present, `fflush` is called instead of the `fdatasync` algorithm as described in MAX_SYNC_ERR_RETRIES on page 282. The `fdatasync` algorithm forces I/O operations to write to the disk, whereas `fflush` delivers the data to the device driver which may have buffering.

This option is intended for debugging purposes, and is not recommended for production.

Syntax

```
USE_FFLUSH [ON|OFF]
```

Options

`ON`

Calls `fflush` instead of `fdatasync`.

`OFF`

Does not call `fflush`.

VALIDATE_TEMPLATE_DATA

Description

Important: This configuration file option is obsolete. You should use `TEMPLATE_HASH_VERIFICATION` instead.

If `VALIDATE_TEMPLATE_DATA` is `ON` in the configuration file and `TEMPLATE_HASH_VERIFICATION` is `ON` in the job control file, the following warning is issued:

```
WARNING: The configuration keyword VALIDATE_TEMPLATE_DATA is
obsolete. The recipe keyword TEMPLATE_HASH_VERIFICATION is used.
```

If `VALIDATE_TEMPLATE_DATA` is `ON` in the configuration file and `TEMPLATE_HASH_VERIFICATION` is either not in the job control file or is set to `OFF`, the following warning is issued:

```
WARNING: The configuration keyword VALIDATE_TEMPLATE_DATA is
obsolete. Please use the recipe keyword
TEMPLATE_HASH_VERIFICATION instead.
```

See also

TEMPLATE_HASH_VERIFICATION on page 267

Program Usage

This section describes the following distributed processing functions:

- concurrent buildgds
- closegds
- dpserver
- initgds

- mktop
- proteus
- remote_server

Note: The initgds, mktop, and closegds utilities are provided to assemble manually a distributed correction that was interrupted during its final phase. These allow you to combine partial results from the various dpservers into a single GDSII file. By applying initgds, mktop, and closegds, in that order, the final GDSII file is created and checked for completeness.

Currently there are no equivalent tools for manually assembling OASIS files.

`PROTEUS_JOB_FLOW` recipes are supported with initgds or mktop with the limitation that they can have only one `TEMPLATE_CALL` in them.

concurrent buildgds

Note: Stand-alone concurrent buildgds is obsolete.

Description

Concurrent buildgds is handled by proteus (the DP controller). Concurrent buildgds creates the final output file in parallel with the correction of templates, distributing some of the output assembly time over the correction time. This is done through the use of multiple fragment files by each DP worker (dpserver). When a DP worker's output fragment grows larger than the "checkpoint" value, or when a DP worker has completed approximately 25%, 50%, 75%, 90%, or 100% of the total area being corrected, the DP worker closes that output and submits it to concurrent buildgds to be assembled into the final output file. The DP worker creates a new fragment file and continues correcting. The checkpoint value is controlled through the `CHECKPOINT_BYTES` configuration file keyword, which defaults to 100 megabytes for a GDSII file and 12.5 megabytes for an OASIS file.

While copying, concurrent buildgds also monitors all templates going to the output and checks these against the list of all templates from the Hierarchy Manager. At the end of all of the fragments, if any cells are missing, the list is reported and concurrent buildgds exits with an error. Each fragment is deleted

after it is successfully copied, unless the `NO_FRAG_CLEAN` keyword is found in the job control file. `NO_FRAG_CLEAN` retains fragments after they are copied.

The creation of an output OASIS file requires the use of concurrent buildgds.

See also

CHECKPOINT_BYTES on page 273

closegds

Description

This utility marks the end of a .gds file, closing the .gds file and library. It tags an `ENDLIB` structure to the end of the current .gds file under construction, padding it such that the .gds file is constructed in increments of 2-K blocks.

Closegds is not called directly, but is available in the distribution for both manual use and debugging. It only works when the output format is GDSII.

Syntax

`closegds` *`gdsfile`*

Options

`gdsfile`

The .gds file.

dpserver

Description

This is the local correction manager, or DP worker. It communicates with the DP controller to request templates for correction, report status of templates, and for general housekeeping. Many dpservers can be attached to one DP controller to assist in the correction process. Dpservers can be added and removed from the DP controller session at any time. (Work in progress on a dpserver may be aborted and recorrected by another dpserver.) A dpserver is removed by killing the process using the standard UNIX `kill -9` command.

Note: Take care to notice which error messages in the log are communication errors, as opposed to other more significant errors. For example, if a dpserver loses communication and

causes an error message to print, the correction output for the run will still be validated (and resent for correction, if needed) and the job should not be killed.

When using `PRINT`, `PRINTLINE`, and `LOGDATA` in distributed mode, output logs are created for each dpserver used in correction, named logname.*server_number*.

Syntax

`dpserver [-p port] [-c host] [-V level]`

Options

`-p port`

The port used by the DP controller. The default port is 2346.

`-c host`

The host on which the DP controller (proteus) is running. The default is the current host. You must have a .rhosts file to allow the DP controller and remote_server to use a remote shell command without login prompts, which stops execution.

`-V level`

A number from 0 to 5 indicating the verbosity of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. The default is 3.

initgds

Description

The initgds utility writes the header information into the .gds file.

This utility only works when the output format is GDSII. `PROTEUS_JOB_FLOW` recipes are supported with the limitation that they can have only one `TEMPLATE_CALL` in them.

Syntax

`initgds [-h][-s] job_control [gdsfilename [libraryname]]`

Options

`-h`

Help. Displays the text help on initgds.

`-s`

Suppresses printing of syntax warnings.

`job_control`

The file generated by xmscript.

`gdsfilename`

The name of the .gds file to initialize. The default `OUTPUT_FILE` parameter is taken from the *`job_control`* file.

`libraryname`

The name written to the library field of the .gds file. The default `CORLIB` parameter is taken from the *`job_control`* file. Note that if *`libraryname`* is specified, *`gdsfilename`* must also be specified.

---

mktop

Description

The mktop utility is used to build the top hierarchy of a corrected GDS output.

You can run this as a thread on the machine where the DP controller runs. See CONCURRENT_MKTOP on page 274 for details.

The mktop utility works only when `OUTPUT_FORMAT` is set to GDSII. `PROTEUS_JOB_FLOW` recipes are supported with the limitation that they can have only one `TEMPLATE_CALL` in them.

Syntax

`mktop [-h][-s]` *`job_control`*

Options

`-h`

Help. Displays the text help on mktop.

`-s`

Suppresses printing of syntax warnings.

`job_control`

The job control file.

proteus

Description

The proteus binary is a single executable supporting hierarchical processing and distributed correction. This executable launches the DP controller. It accepts `PROTEUS_JOB_FLOW` recipes.

Using the proteus binary is the preferred technique for processing new recipes.

Note: `PROTEUS_JOB_FLOW` recipes require the use of the proteus binary application.

Running proteus automatically runs hierarchical processing steps before beginning distributed processing unless the `NO_RERUN_HIERMAN` configuration file option (`OFF` by default) is present (see Chapter 4 on page 121 for more information about hierarchical processing).

Note: If you source a `dproteus.cfg` file, be aware of whether or not `NO_LOCAL_SERVER` is not present there. If you run proteus on your own machine, proteus tries to launch another process, and `NO_LOCAL_SERVER` prevents this. In such a case, consider deleting it or commenting out `NO_LOCAL_SERVER`.

Typically, intermediate files are deleted at the end of a `PROTEUS_JOB_FLOW` run. Place the keyword `RETAIN_JOB_FLOW_FILES` (`OFF` by default) in your job control file to retain intermediate TINF, and graphics output files, and to create intermediate recipe files.

The `DBU_PROC_OUT` keyword allows you to specify the intermediate graphics file resolution in `PROTEUS_JOB_FLOW` recipes. By default, this value is equal to the value of `DBU_PROC` of the current template block. The `DBU_PROC_OUT` value can be overridden in each template call. The specified value should have an integral ratio with the value of `DBU_PROC` of the current `TEMPLATE_BLOCK`.

Syntax

```
proteus [-options] job_control_file
   [start_template|list_file]
```

Options

`job_control_file`

The output of xmscript.

`-h`

Help. This option displays the text help on proteus.

`-s` *`host`*

DP worker host. Runs the remote_server script with host as an argument. This can occur multiple times with the same or different host names. For example:

```
proteus -s host1 -s host2
-s host2 go.pjx
```

The number of allowed DP workers is limited by the number of file descriptors, which can be raised by the system administrator.

`-p` *`port`*

Default port is 2346. Ports are chosen around this starting point. if running multiple DP controllers on the same host, a different *`port`* must be specified for each.

`-V` *`level`*

A number from 0 to 5 indicating the verbosity level of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. Default is 3 unless -f is present, in which case the default is 4.

`-CONFIG_DIRECTIVE`

Overrides any line in the dproteus.cfg file by giving a new definition on the command line. For example:

```
-START_PORT 4000
```

`-f`

Forces proteus into recovery mode to finish a previous failed correction. This cannot be run simultaneously with the *`start_template|recovery_file`* method.

Note: `proteus -f` recovery for `PROTEUS_JOB_FLOW` with `PIPELINE_STRATEGY FRONT_LOAD` or `PIPELINE_STRATEGY BACK_LOAD` might recorrect some templates that were already processed. During `proteus -f` recovery, some DP workers might intermittently produce errors due to file synchronization, but they will not produce bad results.

`-r`

Corrects templates in descending (reverse) order by template number.

`-m`

Runs mktop as a thread on the machine where proteus runs.

`-repair [`*`start_template end_template|list_file`*`]`

Repairs the output file according to the provided list.

`-restart` *`TC_n`*

Forces proteus to restart at the specified `TEMPLATE_CALL`, discarding previous results from `TEMPLATE_CALL`s following *`TC_n`*. The parameter *`TC_n`* is the nth `TEMPLATE_CALL` in the `PROTEUS_JOB_FLOW` section, and cannot be 0, negative, or exceed the maximum number of `TEMPLATE_CALL`s or an error message is printed.

At restart, proteus removes all hierman files, fragment files, output files, and TINF files that are generated from *`TC_n`* and forward. It quickly skips `TEMPLATE_CALL` 1 through n-1 and starts hierman back-end processing for *`TC_n`*. An error message is printed if any needed files are missing. Once templates are available for *`TC_n`*, proteus starts correction directly from *`TC_n`*.

Important: There are limitations on the types of recipe changes that are allowed when performing a restart. Anything that changes the `TEMPLATE_BLOCK`s before the restart is not allowed, including hierman parameters and other global changes such as `NEW_*` or REVERT keywords, `DBU_*` keywords, and so forth. There is no checking mechanism on what is allowed or not with the `-restart` option, so use caution when modifying a recipe with `-restart`. The Proteus tool will still execute if non-allowable recipe content is changed, but incorrect output could result.

Restart is not allowed for the `TEMPLATE_CALL` after a DPT coloring step. `NO_FRAG_CLEAN` is recommended anytime recovery/restart is used for best recovery performance.

remote_server

Description

This is a UNIX shell script that can be customized. When using proteus, it executes this shell script once for each `-s` option on its command line. Note that you can execute remote_server manually to add DP workers to a DP controller session. See Starting DP Workers Individually Using dpserver on page 250.

Syntax

```
remote_server client_host port verbosity call_number
   call_type
```

Options

`client_host`

Remote host name as specified on the proteus command line.

`port`

The TCP port number used by the client.

`verbosity`

A number from 0 to 5 indicating the verbosity of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. The default value is 3. This can be set on the command line from proteus. See proteus on page 296.

`call_number`

Count of the number of calls to remote_server. For example, if this is the 10th call to remote_server, this argument is 9. (Numbering starts at 0.)

This optional parameter can be used within the remote_server shell script to introduce a startup delay between servers. See remote_server Shell Script Examples on page 301 for an example.

`call_type`

Flag specifying whether remote_server is called by distributed hierman or proteus. The *`call_type`* is either 0 (hierman) or 1 (proteus). For all OPC runs, the value is 1. See Chapter 9, Proteus Applications for more information on distributed hierman.

remote_server Execution

After calling remote_server, the following occurs:

1. remote_server first executes a test script called system_tests.sh that can be found in $PRECIM_HOME/bin. If this script fails, a message is written to the standard output of the proteus process. The system test script tests for a certain operating system compatibility and whether rsh or remsh work properly for the remote host.

   For all remote execution scripts, rsh is used for Linux, and remsh is used for all other operating systems (currently SUN and HP).

2. Next, remote_server uses rsh or remsh to run dpserver on the specified host. The full command line passed to the host includes three components:

   - `rsetup="cd `pwd`;DISPLAY=$display;export DISPLAY"`

     This section sets up the environment so that each dpserver is run from the current directory (that is, where the OPC job is run) and has the DISPLAY variable properly set, if necessary.

   - `dpcommand="`which dpserver` -p$2 -c$HOST -V$3"`

     This section sets the dpserver command to specify the port number, proteus host name, and verbose level for logging.

   - `shcommand="{ { $dpcommand >>logfile.txt.$1.$$; } 3>&1 1>&2 2>&3 |tee -a logfile.txt.$1.$$; } &>/dev/null &"`

     ```
     if [ $OS != "Linux" ]; then
        remsh $1 -n "nohup sh -c \"$rsetup; $shcommand\""&
     else
        rsh $1 -n "nohup sh -c \"$rsetup; $shcommand\""&
     fi
     ```

     This statement combines the `rsetup` and `dpcommands` into one command line. It then uses standard UNIX redirection and `tee` command to output all of the dpserver text output to a logfile that contains the hostname and process number of the remote_server call. (Note that to get the best output in the log files, use `tee -a`, which appends the output to the end of the file rather than overwriting it.)

     The end `&>/dev/null` ignores outputs of the overall `shcommand` so that the later `nohup` could close rsh/remsh connection as soon as dpserver is launched.

Using ssh

Some users might require using ssh (secure shell) instead of rsh or remsh. For this case, UNIX administrators and users need to set up ssh and related configuration files so that no password is required to log into each dpserver machine from the proteus machine.

The remote_server script should be modified to use ssh instead of rsh or remsh. You can update the file $PRECIM_HOME/bin/remote_server, or this file can be copied to another location and modified. Be sure to add the new remote_server to your path so that it is called before the original file in the PRECIM_HOME directory.

After modifying your `PATH` variable, test the location of remote_server using the command `which remote_server`. Ensure that all other requirements are met as listed in System Requirements on page 244 and User Account Requirements on page 246.

remote_server Shell Script Examples

The remote_server script can be customized to suit the needs of a particular environment. Options for customizing the script might include: changing message logging; setting up environment variables; setting the environment (use korn shell or bourne shell instead of c shell); modifying the execution parameters, such as the `-V` option; and modifying for execution in a batch environment such as submitting a job to a batch queue. Consult your system administrator for assistance in this customization. See also Using ssh on page 301.

The remote_server script can be as simple as the ones shown in the following examples.

Example 1 Example A

```
#!/usr/bin/csh -f

remsh $1 -n "cd ‘pwd‘;dpserver -p$2 -c$HOST -V$3 >>& \
     logfile.txt.$1.$$"&
```

Example 2 Example B

```
#!/usr/bin/csh -f

remsh $1 -n "cd ‘pwd‘;xterm -iconic -title dproteus_$1 -1 -1f \
     logfile.txt.$1.$$ -display $display -e dpserver -p$2 \
     -c$HOST -V$3"&
```

Example 3 Example C

```
#!/usr/bin/csh -f

@temp = $4 * 30
sleep $temp

remsh $1 -n "cd ‘pwd‘;dpserver -p$2 -c$HOST -V$3 >>& \
     logfile.txt.$1.$$"&
```

In the previous examples, $1 indicates the first argument, $2 indicates the second argument, and so forth. Here, $1 is the remote host name, $2 is proteus's port, $3 is proteus's verbosity level, and $4 is the sequence of each server (where the first is 0, the second is 1, and so forth).

In Example a, remote_server performs a remote shell on host that changes to the current directory starts a dpserver with the proper arguments, and appends dpserver's messages to a log file.

In Example b, remote_server performs a remote shell on <host> that changes to the current directory. It starts an xterm that executes a dpserver with the proper arguments, and sends dpserver's messages to a log file.

In Example c, remote_server calculates a sleep time based on the sequence of this remote_server (argument $4 from proteus), so the first server has 0 sleep time, the second has 30 seconds, the third has 60 seconds, and so forth. After the sleep period, the script continues as in Example B.

Note: You must have a .rhosts file to allow proteus and remote_server to use a remote shell command without login prompts, which stop execution.

Error Recovery

Note the following items in reference to error recovery:

- If any templates fail, they are marked for retry.
- Templates are retried up to five times after all templates have been attempted at least once.
- If a DP worker loses communication with the DP controller, the condition is recognized and the DP worker is removed from the list of active DP workers. If the DP worker was working on a template, the template is marked for retry and passed to another DP worker for correction.

- If communication is lost, but the DP worker completes the template successfully, the template can be corrected again by another DP worker. This can result in duplicate templates in the fragment files. This duplication is handled by concurrent buildgds and no duplicate templates will exist in the output file.
- In the case of a failed correction, proteus can invoke a full automatic recovery by using the `-f` command-line option.
- The DP controller (proteus) puts a list of failed templates in a file named after the job control file with .failed appended. This failed file is in the correct format for the recovery list described in the previous bullet item.

Currently Known Limitations

There are a few limitations to the DP controllers and DP workers:

- Currently, all DP controllers and DP workers must be able to see the same disk space. Typically this is accomplished using NFS mounting. See your system administrator for details.
- If the correction consists mostly of small templates (fewer than 50 vertices, correction time of less than 0.1 seconds), and many DP workers are connected, the overhead of the DP controller/DP worker communication and file I/O can introduce some inefficiency. This is typically less than a 10-15% of overall CPU time. As a result, DP controller/DP worker does not have the 1/n performance in some cases.
- The maximum number of connections to a DP controller is limited by the shell limit on the number of open file handles. Although this varies, it is often around 1,024. See your system administrator for changing this limit on your system.
- If a job executes quickly, or if more DP workers are started than are needed, DP workers can remain after a client finishes. Leftover DP workers can cause errors on subsequent runs.

Distributed Processing Example

This section provides an example of how you can initiate jobs using distributed processing.

Assume that remote_server has been customized for the needs of the installation, and that the `DPROTEUS_CFG` environment variable has been set to indicate the configuration file. Change to the correction directory, and use the following syntax:

```
% proteus -sServer1 -sServer2 -sServer3 go.example
```

or, to run templates in reverse, you would enter:

```
% proteus -r -sServer1 -sServer2 -sServer3 go.example
```

When proteus starts, it begins a dpserver on the current host, then starts the dpservers on the host. The message from the DP controller is not saved in any log file. Messages from the DP workers are saved in a log file whose name is defined in remote_server.

Output Logs

Example 1

The following is an output log from a proteus run on a multiblock recipe.

```
proteus Release A-2007.12-SP2-4 Revision Proteus_A-2007.12-SP2-
4_Development-1620626 (64f/64m LINUX_X86_64).
host: hydrox07

 Proteus (TM) / PROTEUS (TM)
 Version A-2007.12-SP2-4

*** Copyright (C) 1995 - 2009 Synopsys, Inc. ***
*** This software and the associated documentation are ***
*** confidential and proprietary to Synopsys, Inc. ***
*** Your use or disclosure of this software is subject to ***
*** the terms and conditions of a written license agreement ***
*** between you, or your company, and Synopsys, Inc. ***
*** ***


Testing for license PROTEUS_OPC...

Checking out PROTEUS_OPC...
License PROTEUS_OPC checked out.
hierman -fe first_example.pjx

 Hierarchy management started for job PROTEUS; Wed Jul 22 15:36:42
2009

Warning: Previous run of job ./temp/PROTEUS
```

```
 did not complete.
Job was interrupted sometime after Wed Jul 22 15:35:58 2009.

Reading gdsfile sample.gds
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................

Scanning for Topcell(s)
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................
 Topcell: TOP
 native hierarchy: 267 cells; 500 refs
     Times: User: 0.01 Sys: 0.01 Memory: 2.73M
Separating graphics from SREFs & AREFs.
     Times: User: 0.00 Sys: 0.00 Memory: 2.19M
Calculating clustering statistics.
     Times: User: 0.00 Sys: 0.00 Memory: 2.45M
Cleaning 1D AREF transforms.
Partitioning large graphic cells.
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................
     Times: User: 0.00 Sys: 0.00 Memory: 2.19M
Subdividing large AREFs.
     Times: User: 0.00 Sys: 0.00 Memory: 2.19M
Constructing framework for revised hierarchy.
 revised hierarchy: 267 cells; 500 refs
     Times: User: 0.00 Sys: 0.00 Memory: 2.19M
Generating spatial bins for Topcell.
 Bin Count : 2 x 3 (horizontal x vertical)
 Bin Dimensions : (35554,46074) (x,y)
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................
 revised hierarchy: 274 cells; 872 refs
 7 instances identified (6 cluster instances).
     Times: User: 0.00 Sys: 0.00 Memory: 2.20M

Writing partial hierarchy results.
     Times: User: 0.00 Sys: 0.00 Memory: 2.20M
     Total Times: User: 0.01 Sys: 0.01 Elapsed: 0 Memory: 2.73M

 Hierarchy management complete for job PROTEUS; Wed Jul 22 15:36:42
2009
 Hierman FrontEnd Times:User: 0.03 Sys: 0.03 Elapsed: 1 Memory:
1.80M
 Cleaning up double patterning files from a previous run
 rm -rf ./temp/PROTEUS_TB1.DPT 2&gt; /dev/null

 Cleaning up TB 1 fragment files and outputs from a previous run
```

```
 rm -rf first_example_out.gds.dir/TB1/ 2&gt; /dev/null
 rm -f FIRST_BLOCK_out_1.oas 2&gt; /dev/null

 Cleaning up TB 2 fragment files and outputs from a previous run
 rm -rf first_example_out.gds.dir/ 2&gt; /dev/null
 rm -f first_example_out.gds 2&gt; /dev/null

 mkdir -p first_example_out.gds.dir/TB1/ 2&gt; /dev/null
 mkdir first_example_out.gds.dir/TB1/TINF 2&gt; /dev/null
 initgds FIRST_BLOCK_out_1.oas
 Cleaning up double patterning files from a previous run
 rm -rf ./temp/PROTEUS_TB2.DPT 2&gt; /dev/null
 mkdir -p first_example_out.gds.dir/ 2&gt; /dev/null
 mkdir first_example_out.gds.dir/TINF 2&gt; /dev/null
 initgds first_example_out.gds
Generating templates.
 generate templates for TB 1: 10%
 generate templates for TB 1: 20%
 generate templates for TB 1: Done Wed Jul 22 15:36:43 2009

     Times: User: 0.01 Sys: 0.06 Memory: 1.82M
 7 output templates (6 cluster templates) generated for TB1:
 flat: 6 templates (6 instances)
 holder: 1 templates (1 instances)
 Total topcell bounding area: 9827.8487 microns.
 Sum of all template bounding areas: 9860.4983 microns.
 Sum of all ambit-biased template bounding areas: 11001.9324
microns.

 Correction job started for TB 1, Wed Jul 22 15:36:43 2009

trying to open port:2346
success opening port:2346
/remote/avanti/proteus/mask_synthesis/bin/grid_server hydrox07
2346 3 0 1
/remote/avanti/proteus/mask_synthesis/bin/grid_server (1.19)
dpserver "hydrox07.0" log file at: dpserver.log.0
Client: 1:1:INIT :N_TMPL=7 SOCK_CNT=1 JCF=first_example.pjx TB=1
LASTTB=2
 # mktop done, ~ 14.29% Completed ~ 0.00% area (of TB 1)
 # 6 done, ~ 28.57% Completed ~ 16.67% area (of TB 1)
 # 5 done, ~ 42.86% Completed ~ 33.36% area (of TB 1)
 # 4 done, ~ 57.14% Completed ~ 50.03% area (of TB 1)
 # 3 done, ~ 71.43% Completed ~ 66.70% area (of TB 1)
 # 2 done, ~ 85.71% Completed ~ 83.37% area (of TB 1)
 # 1 done, ~ 100.00% Completed ~ 100.00% area (of TB 1)

Concurrent: All data is present in the output for TB 1.
```

```
 Prepare the holder cells for bound box data calculation for TB1.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................

 Collecting cells bound box data for TB1.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................

 Finalizing OASIS output file for TB1.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................

 Correction job complete for TB 1, Wed Jul 22 15:36:51 2009
 TB1 Times:User: 0.02 Sys: 0.07 Elapsed: 9 Memory: 6.24M
 Scanning cell graphics from previous template block.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.01 Sys: 0.02 Memory: 6.24M
Generating templates.
     Times: User: 0.00 Sys: 0.01 Memory: 2.55M
 generate templates for TB 2: 10%
 generate templates for TB 2: 20%
 generate templates for TB 2: Done Wed Jul 22 15:36:51 2009

     Times: User: 0.02 Sys: 0.03 Memory: 2.55M
 7 output templates (6 cluster templates) generated for TB2:
 flat: 6 templates (6 instances)
 holder: 1 templates (1 instances)
 Total topcell bounding area: 9844.6008 microns.
 Sum of all template bounding areas: 10486.2144 microns.
 Sum of all ambit-biased template bounding areas: 11662.0726
microns.

 Correction job started for TB 2, Wed Jul 22 15:36:51 2009

 # mktop done, ~ 14.29% Completed ~ 0.00% area (of TB 2)
 # 6 done, ~ 28.57% Completed ~ 16.60% area (of TB 2)
 # 5 done, ~ 42.86% Completed ~ 33.57% area (of TB 2)
 # 4 done, ~ 57.14% Completed ~ 50.06% area (of TB 2)
 # 3 done, ~ 71.43% Completed ~ 66.45% area (of TB 2)
 # 2 done, ~ 85.71% Completed ~ 83.43% area (of TB 2)
 # 1 done, ~ 100.00% Completed ~ 100.00% area (of TB 2)

Concurrent: All data is present in the output for TB 2.
 server 1: hydrox07 exited: MSG_NUM 38 Total Times:User: 0.35
Sys: 0.04 Elapsed: 3 Memory: 6.46M

 Correction job complete for TB 2, Wed Jul 22 15:36:53 2009
```

```
 Correction job complete for all template blocks, Wed Jul 22
15:36:53 2009

 TB2 Times:User: 0.04 Sys: 0.14 Elapsed: 11 Memory: 6.74M

Checking in PROTEUS_OPC...

 All servers exited normally.

 Total Times:User: 0.07 Sys: 0.17 Elapsed: 13 Memory: 6.74M
```

Example 2

One of the log files from a proteus is shown below, displaying the area log for the output. Similar messages can be seen, in addition to setup and cleanup messages and intermediate steps for each correction template. When using `PRINT`, `PRINTLINE`, and `LOGDATA` in distributed mode, output logs are created for each server used in correction.

```
trying to open port:2349
success opening port:2349

Reusing hierarchy data created Wed Dec 15 14:42:33 2004.
   Warning: output file ./demo.oas will be replaced.
    rm -rf ./demo.oas.dir 2> /dev/null
    rm -f ./demo.oas 2> /dev/null

 1 copy of DC locked (expires 27-jan-2005)....
    mkdir ./demo.oas.dir 2> /dev/null
    mkdir ./demo.oas.dir/TINF 2> /dev/null
initgds ./demo.oas
remote_server mite 2349 3 0 1
Executing system check script on remote host
sh -c "{ { dpserver -p2349 -cgerbil -V3 >>logfile.txt.mite.19048;
} 3>&1 1>&2 2>&3 |tee logfile.txt.mite.19048; } 3>&1 1>&2 2>&3"
Client: 1:1:INIT           :N_TMPL=43 SOCK_CNT=1 JCF=go.pjx
    # mktop done, ~ 39.53% Completed ~  0.00% area
    #    42 done, ~ 41.86% Completed ~  4.66% area
    #    41 done, ~ 44.19% Completed ~ 12.36% area
    #    40 done, ~ 46.51% Completed ~ 18.36% area
    #    39 done, ~ 48.84% Completed ~ 18.50% area
    #    38 done, ~ 51.16% Completed ~ 22.22% area
    #    37 done, ~ 53.49% Completed ~ 29.03% area
    #    36 done, ~ 55.81% Completed ~ 35.85% area
    #    35 done, ~ 58.14% Completed ~ 42.66% area
    #    34 done, ~ 60.47% Completed ~ 49.48% area
    #    33 done, ~ 62.79% Completed ~ 56.29% area
```

```
#    32 done, ~ 65.12% Completed ~ 63.10% area
#    31 done, ~ 67.44% Completed ~ 65.13% area
#    30 done, ~ 69.77% Completed ~ 65.44% area
#    29 done, ~ 72.09% Completed ~ 65.76% area
#    28 done, ~ 74.42% Completed ~ 66.47% area
#    27 done, ~ 76.74% Completed ~ 67.18% area
#    26 done, ~ 79.07% Completed ~ 83.54% area
#    25 done, ~ 81.40% Completed ~ 99.89% area
#    24 done, ~ 83.72% Completed ~ 99.90% area
#    23 done, ~ 86.05% Completed ~ 99.92% area
#    22 done, ~ 88.37% Completed ~ 99.93% area
#    21 done, ~ 90.70% Completed ~ 99.94% area
#    20 done, ~ 93.02% Completed ~ 99.96% area
#    19 done, ~ 95.35% Completed ~ 99.97% area
#    18 done, ~ 97.67% Completed ~ 99.99% area
#    17 done, ~ 100.00% Completed ~ 100.00% area

Concurrent: 26 templates and mktop data has been appended
           to the output from the fragment file "./demo.oas.dir/
demo.oas.1.0".

Concurrent:    All data is present in the output.
  server  1:     mite exited: Total Times:User:    7.15 Sys:
0.10 Elapsed:      13

Correction job complete, Wed Dec 15 14:43:16 2004

  Total Times:User:    0.10 Sys:    0.09 Elapsed:      23
```

7

Celltool

Provides syntax and examples for Celltool.

Celltool Syntax

Celltool is used to:

- Extract and process data for a specific cell for recipe development or troubleshooting in the Proteus WorkBench application.
- Execute phases of the correction process on a template.

Celltool takes the form:

`celltool [-options] job_control_file`

where *`job_control_file`* is the output of xmscript.

When debugging the first `TEMPLATE_CALL`, the data that must exist is

- the input GDS/OASIS
- output from hierman FE and context analysis for the first template block

When debugging a `TEMPLATE_CALL` other than the first, the data that must exist is

- the output fragment files from the previous `TEMPLATE_CALL` (be sure you have `NO_FRAG_CLEAN` in your recipe during the prior run of the proteus binary)
- the temp files output by the context analysis for that `TEMPLATE_CALL`

The following options are valid in Celltool:

`-tb` *`n`*	Specifies a specific template block to be executed. Only the specified template block will be executed. Running celltool on a `PROTEUS_JOB_FLOW` recipe without the `-tb` argument causes the execution to run on `TEMPLATE_BLOCK 1` by default.
`-i` *`template`*	Here, *`template`* is the index of the correction template upon which to act. This must be present for the `-e` option and the `-m` option, if `-n` is not provided.
`-n` *`name`*	This overrides the output cell name.
`-I`	For GDSII jobs, this calls `initgds` before making the GDS structure. This is valid only if `-m` is used. For OASIS jobs, this is automatically invoked with the `-m` option, creating a new output file and overwriting any existing file.
`-C`	For GDSII jobs, this calls `closegds` after making the GDS structure. This is valid only if `-m` is used. For OASIS jobs, this is automatically invoked with the `-m` option to close the file at the end of the `-m` operation.
`-E` *`extension`*	This overrides the default extension of the input and output .vrt files.
`-b` *`x y x y`*	This reports the instances (with transform) of graphics templates that overlap the bounding box defined by *`(x, y)`*s. The *`(x, y)`*s are given at 1xW scale (scaling from input to 1xW: `DBU_IN*SCALE_IN`). This is a standalone option, and no other options are permitted. Reported transform information is in the form: *`template_number rotation mag mirror x_offset y_offset`* as calculated from the topcell. When used with this option, celltool accepts encrypted content. See #ENCRYPT #END_ENCRYPT on page 15 for details.

`-bi` *`x y x y`*	This reports the instances (with transform) of graphics templates that overlap the bounding box defined by (x, y)s. The (x, y)s are given at 1xW scale (scale factor of `DBU_IN`, scaling from input data to 1xW: `DBU_IN*SCALE_IN`). This is a standalone option, and no other options are permitted. Reported transform information is in the form: *`template_number rotation mag mirror x_offset y_offset`* as calculated from the topcell. When used with this option, celltool accepts encrypted content. See #ENCRYPT #END_ENCRYPT on page 15 for details.
`-bo` *`x y x y`*	This reports the instances (with transform) of graphics templates that overlap the bounding box defined by (x, y)s. The (x, y)s are given at 1xW scale (scale factor of `DBU_OUT`, scaling from output data to 1xW: `DBU_IN*SCALE_IN*SCALE_OUT`). This is a standalone option, and no other options are permitted. Reported transform information is in the form: *`template_number rotation mag mirror x_offset y_offset`* as calculated from the topcell. When used with this option, celltool accepts encrypted content. See #ENCRYPT #END_ENCRYPT on page 15 for details.

`-T` *`template number`*

Template report. This reports the details of the supplied template number, including the listing of all placements and the cells that comprise the context. The following items are reported for the original cell:

- Templ: Template number
- Inst: Instance number of placement
- Rot: Whether a rotation has been applied to this instance (Yes=1, No=0)
- Mag: Whether a magnification transform has been applied to this instance (Yes=1, No=0)
- Mir: Whether a mirroring transform has been applied to this instance (Yes=1, No=0)
- X Off/Y Off: The cell's offset within the template. X and Y Offsets are the coordinates of the center of the template. w,s,e,n (West, South, East, North) refer to the respective locations of the four extremities of the template.
- BIS1234: B=Blind, I=Invisible, S=Suppress
  B,I,S options allow you to specify special handling by the hierarchy manager of templates containing certain cells.
  1, 2, 3, 4 refer to MARK types defined in the order in the recipe file.
  The letter "N" appearing under any of the above (BIS1234) indicates that the corresponding option was not enabled for that particular template.

When used with this option, celltool accepts encrypted content. See #ENCRYPT #END_ENCRYPT on page 15 for details.

`-P` *`cellname`*

Placement report. This reports all templates to which this named cell contributes in either .main or .context. This is limited to where the named cell is not under a cluster (meaning, clusters are not searched for this cell). The following items are reported for the original cell:

- Templ: Template number within which the named cell will be reported as .main
- Inst: Instance number of any placement of this cell
- Rot: Whether a rotation has been applied to this instance (Yes=1, No=0)
- Mag: Whether a magnification transform has been applied to this instance (Yes=1, No=0)
- Mir: Whether a mirroring transform has been applied to this instance (Yes=1, No=0)
- X Off/Y Off: The cell's offset within the template. X and Y Offsets are the coordinates of the center of the template. w,s,e,n (West, South, East, North) refer to the respective locations of the four extremities of the template.
- BIS1234: B=Blind, I=Invisible, S=Suppress
  B,I,S options allow you to specify special handling by the hierarchy manager of templates containing certain cells.
  1, 2, 3, 4 refer to MARK types defined in the order in the recipe file.
  The letter "N" appearing under any of the above (BIS1234) indicates that the corresponding option was not enabled for that particular template.

`-M`

Mark report. This reports any instance that is marked and has special treatment set (`blind`, `suppress`, and so forth).

`-N` *`name`*

This reports template information for original cells. The following items are reported for the original cell:

- H/G - Holder or Graphics cell
- type - a letter indicating the cell type:
  A - AREF scaffold
  F - Flat
  G - Graphics scaffold
  L - Leaf scaffold
  N - Native cell
  S - SREF scaffold
- n - the number of templates from this cell
- range - the range of template numbers for this cell
- name - the name of this original cell

If the `ALL` keyword is given for the name, all cells are reported. This command cannot be used with any other options nor in `COMPRESSION FLAT` mode.

`-G` *`inst`*

Genetics report. This reports the lineage of the supplied instance up to the topcell.

`-h`

Help. This displays information on CellTool.

`-e`

This extracts graphics. This option creates a .vrt file for each input layer of a template block in the form *layer_name*.vrt.

`-p`

This executes Boolean operations and creates a .vrt file for each Boolean result layer of the template block. This uses the previous template block files as input. In addition, for each `corbasic()` function call, the following files are generated:

Inputs to corBASIC include:

- COR_IN.n.vrt - all `COR_IN:x` items
- REC_IN.n.vrt - all `REC_IN:x` items
- REF_IN.n.vrt - all `REF_IN:x` items

Outputs from corBASIC include:

- DISTAR_OUT.n.vrt - dissected target
- COR_OUT.n.vrt - corrected target
- DISREC_OUT.n.vrt - dissected recursion
- REC_OUT.n.vrt - corrected recursion
- REF_OUT.n.vrt - cleaned reference
- AUX_OUT.n.vrt - aux features, present if generated

In the previous names, n refers to the `corbasic()` call number, which is incremented for each `corbasic()` call in the recipe.

`-m`

If a GDSII output format has been chosen, this performs `GRAPHICS_OUTPUT`, allowing the specified cell to be appended to the output file. For a graphics cell, this uses the previous output_group files as input. This option appends the data to the `OUTPUT_FILE` defined in the job control file. The `-I` and `-C` (initialize and close file, respectively) options can also be used independently.
If an OASIS output format has been chosen, the `celltool -m` option also invokes the `-I` and `-C` options, which will create a new output file, overwriting any existing file, appending the cell information, and closing the file. This results in a legal OASIS file that contains a single cell. Celltool generates a warning message that reads:
`Warning: When OASIS output format has been set, the -m option works as -m -I -C, so the output OASIS file will contain only one template.`

`-xy` *`x_coord y_coord`* `[-tb` *`TB#`*`] [-cb` *`corbasic_call#`*`]`	Retrieves information such as instance and polygon number by coordinates. ▪ `-xy` *`x_coord y_coord`* specifies an (x,y) location in output GDSII/OASIS scale in nanometers (this corresponds to the `-bo` Celltool option for bounding boxes). ▪ `-tb` *`TB#`* specifies the `TEMPLATE_BLOCK` (default is 1). ▪ `-cb` *`corbasic_call#`* specifies the `corbasic()` call within that `TEMPLATE_BLOCK` (default is 1).

Specifying Celltool Options

At least one option must be specified. The order in which you write the options on the command line is not important, but Celltool maintains an order of execution regardless. For example:

```
-e -d <jcl> -i5
```

performs the extraction through the dissection.

The source group files opened and saved by another application do not contain any mark, main information, or context information. As a result, Boolean groups might not behave as expected.

All operations require a valid Hierarchy Manager run prior to execution.

Comma-Delimited Arguments in Celltool

The Celltool report options `-T`, `-P`, `-N`, and `-G` accept comma-separated lists for names (in the cases of `-P` and `-N`) or for numbers (in the cases of `-T` and `-G`). For example,

```
celltool -T 5,9,2 test.pjx
```

runs a template report on templates 5, 9, and 2.

```
celltool -P nor3,nor2,nor1  test.pjx
```

runs a placement report on cell names `nor3`, `nor2`, and `nor1`.

The `-f` suboption takes one argument that specifies a filename containing a list similar to the previous comma-separated lists. This suboption can only be used following a `-T`, `-P`, `-N`, or `-G`, and it can only occur once on the command line. It must be the only argument to the -T, `-P`, `-N`, or `-G` option.

For example,

```
celltool -T -f list.txt test.pjx
```

reads the list contained in the file `list.txt`. This list file is meant for use as an alternative to the command-line, comma-separated lists described previously.

Items within the list file can be separated by a space, tab, new line, or comma. Apostrophes begin comments that extend to the end of the line. Such comments are ignored by the list file reader.

Celltool Examples

The following examples demonstrate the Celltool functions.

Note: Examples 2 through 5 use GDSII, but the results would be the same for OASIS in all but Example 2.

Example 1

The following example uses the `-tb`, `-e`, `-p`, `-m`, and `-i` options to create the .vrt files for template 98.

Note: If this example were to use OASIS, a warning message would be generated because the `-m` option is used. See `-m` in Celltool Syntax on page 311.

```
% celltool -tb 2 -e -p -m -i 98 go_demo.pjx
celltool Release A-2007.12-SP2-2 Revision Proteus_A-2007.12-SP2-
2_17Jun09-1597635 (64f/64m LINUX_X86_64).
host: hydrox07
                    Proteus (TM) / PROTEUS (TM)
                     Version A-2007.12-SP2-2

***   Copyright (C) 1995 - 2009 Synopsys, Inc.                 ***
***   This software and the associated documentation are       ***
***   confidential and proprietary to Synopsys, Inc.           ***
***   Your use or disclosure of this software is subject to    ***
***   the terms and conditions of a written license agreement  ***
***   between you, or your company, and Synopsys, Inc.         ***
```

```
***                                                                    ***

Testing for license PROTEUS_OPC...

Checking out PROTEUS_OPC...
License PROTEUS_OPC checked out.

Testing for license PA...

Checking out PA...
License PA checked out.

Testing for license DS...

Checking out DS...
License DS checked out.

 Warning: When OASIS output format has been set, the -m option works as
-m -I -C, so the output OASIS file will contain only one template.

 Checking current state for running on TB #:2/3...

  Correction job started for TB 2, Thu Jul 16 14:18:24 2009

init outputfile OPC_BLK_out_2.oas
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Source Group
   : cor_layer.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Source Group  :
outrig_layer_1.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Source Group  :
outrig_layer_2.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Source Group  :
base_chrome_org_in.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : outrig_layer.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : outrig_out.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : cor_all.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : cor_main.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group  :
cor_main_bound.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : target_bound.vrt
```

```
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : target_bound.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : empty_layer.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : avoid_ring.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group :
cor_all_healed.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group :
cor_all_target.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : contacts.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : ref_target.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : ref_inrig.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group :
cor_all_target.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group :
do_flash_layer.vr
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Correction
   : COR_IN.0.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Correction
   : REC_IN.0.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Correction
   : REF_IN.0.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Correction
   : DISTAR_OUT.0.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Correction
   : DISREC_OUT.0.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Correction
   : COR_OUT.0.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Correction
   : REC_OUT.0.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Correction
   : REF_OUT.0.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Correction
   : AUX_OUT.0.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : blott.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : blott.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : cor_all_out.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : cor_all_out.vrt
```

```
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : mrcErrors0.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : mrcErrors1.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : mrcErrors2.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group
   : mrcErrors3.vrt
   Saving data for template 98 of 798 "98N_SDFCNQD2HVT_3R" Program Group :
base_chrome_org_in_PPP_DEBUG_MAIN.vrt
     graphics:98N_SDFCNQD2HVT_3R
       Times:User:    8.48 Sys:    0.21 Memory:  152.29M
       Times:User:    0.00 Sys:    0.00 Memory:  124.92M
close output file OPC_BLK_out_2.oas

  Correction job complete for TB 2, Thu Jul 16 14:18:30 2009

       Post Times: User:    0.00 Sys:    0.00 Memory:  124.92M
       Total Times: User:   8.48 Sys:   0.21 Elapsed:    12 Memory: 152.29M

Checking in PROTEUS_OPC...

Checking in PA...

Checking in DS...
```

Example 2

The following example uses the `-e`, `-m`, and `-i` options to create the .vrt files for template 40, and perform the `GRAPHICS_OUTPUT` function.

Note: If this example were to use OASIS, a warning message would be generated because the `-m` option is used. See `-m` in Celltool Syntax on page 311.

```
% celltool -e -m -i 40 test.jcl
celltool Release 2004.09 Rev W-2004_09_development-493831 (64f/64m)
Copyright 1995 - 2004
Synopsys, Incorporated

_Legal_Notice_
1 copy of DC locked (expires 20-dec-2003)....
```

```
1 copy of DS locked (expires 20-dec-2003)....
 Saving data for template 40 of 46 "40N_testCell" Source Group  :
to_correct.vrt
 Saving data for template 40 of 46 "40N_testCell" Source Group  :
seg_ref.vrt
 Saving data for template 40 of 46 "40N_testCell" Source Group  :
seg_ref_2.vrt
 Saving data for template 40 of 46 "40N_testCell" Program Group :
relevant.vrt
 Saving data for template 40 of 46 "40N_testCell" Program Group : merged.vrt
 Saving data for template 40 of 46 "40N_testCell" Correction : COR_IN.0.vrt
 Saving data for template 40 of 46 "40N_testCell" Correction : REC_IN.0.vrt
 Saving data for template 40 of 46 "40N_testCell" Correction : REF_IN.0.vrt
 Saving data for template 40 of 46 "40N_testCell" Correction :
DISTAR_OUT.0.vrt
 Saving data for template 40 of 46 "40N_testCell" Correction :
DISREC_OUT.0.vrt
 Saving data for template 40 of 46 "40N_testCell" Correction :
COR_OUT.0.vrt
 Saving data for template 40 of 46 "40N_testCell" Correction :
REC_OUT.0.vrt
 Saving data for template 40 of 46 "40N_testCell" Correction :
REF_OUT.0.vrt
 Saving data for template 40 of 46 "40N_testCell" Correction :
AUX_OUT.0.vrt
 Saving data for template 40 of 46 "40N_testCell" Program Group :
expanded.vrt
 Saving data for template 40 of 46 "40N_testCell" Program Group : blott.vrt
 Saving data for template 40 of 46 "40N_testCell" Program Group :
intrude.vrt
 Saving data for template 40 of 46 "40N_testCell" Program Group :
finished.vrt
graphics:40N_testCell
Times:User:   11.77 Sys:    0.07
Correction job complete, Tue Feb  4 23:01:32 2003

Post Times: User:   0.00 Sys:   0.01
Total Times: User:  12.51 Sys:  0.21 Elapsed:  17
```

Example 3

This example reports the instances of graphics templates that overlap the bounding box defined by vertices at (0, 0) and (10000, 10000). The results are scaled to the input.

```
% celltool -bi 0 0 10000 10000 startline.jcl
celltool Release 2004.09 Rev W-2004_09_development-493831 (64f/
64m)
Copyright 1995 - 2004
Synopsys, Incorporated

_Legal_Notice_

1 copy of DC locked (expires 20-dec-2004)....

1 copy of DS locked (expires 20-dec-2004)....
Bounding box:  X_left=0   Y_bottom=0   X_right=10000  Y_top=10000
Scaled to 1xW: X_left=0   Y_bottom=0   X_right=16000  Y_top=16000
Results will be scaled to input
-----------------------------------------------------------------
Templ #   Xform:  Rot  Mag  Mir  X Off  Y Off  Name
-----------------------------------------------------------------
   251           0    1   N    0     0     251G_bitcellvert_cell
   252           0    1   N    0     0     252G_bitcellvert_cell
   256           0    1   N    0     0     256G_bitcellvert_cell
-----------------------------------------------------------------
```

Example 4

This example reports the instances of graphics templates that overlap the bounding box defined by vertices at (0, 0) and (10000, 10000). The results are scaled to the output GDS.

```
% celltool -b0 0 0 10000 10000 startline.jcl
celltool Release 2004.09 Rev W-2004_09_development-493831 (64f/
64m)
Copyright 1995 - 2004
Synopsys, Incorporated

_Legal_Notice_

1 copy of DC locked (expires 20-dec-2004)....

1 copy of DS locked (expires 20-dec-2004)....
Bounding box:  X_left=0   Y_bottom=0   X_right=10000  Y_top=10000
Scaled to 1xW: X_left=0   Y_bottom=0   X_right=10000  Y_top=10000
Results will be scaled to output GDS
-----------------------------------------------------------------
Templ #   Xform:  Rot  Mag  Mir  X Off  Y Off  Name
-----------------------------------------------------------------
   251           0    1   N    0     0     251G_bitcellvert_cell
-----------------------------------------------------------------
```

Example 5

This example reports the instances of graphics templates that overlap the bounding box defined by vertices at (0, 0) and (10000, 10000). The results are scaled to 1xW.

```
% celltool -b 0 0 10000 10000 startline.jcl
celltool Release 2004.09 Rev W-2004_09_development-493831 (64f/
64m)
Copyright 1995 - 2004
Synopsys, Incorporated

_Legal_Notice_

1 copy of DC locked (expires 20-dec-2004)....

1 copy of DS locked (expires 20-dec-2004)....
Bounding box:  X_left=0  Y_bottom=0   X_right=10000   Y_top=10000
Scaled to 1xW: X_left=0  Y_bottom=0   X_right=10000   Y_top=10000

Results will be scaled to 1xW
-----------------------------------------------------------------
Templ #   Xform:  Rot  Mag  Mir  X Off  Y Off  Name
-----------------------------------------------------------------
   251           0    1   N    0      0      251G_bitcellvert_cell
-----------------------------------------------------------------
```

Example 6

The command `celltool -xy 30500 131300 -tb 2 -cb 1` returns:

```
TB# 2, corBASIC call# 1
XY coordinates:  X=30500       Y=131300
Scaled to 1xW:   X=15250       Y=65650
Results will be scaled to output file.
-----------------------------------
Templ #   Name       Inst #
-----------------------------------
-----------------------------------
 227 227G_YHHROWBL     1358

From REF_IN Polygon#  11 61
-----------------------------------------------------------------
```

8

Recipe Debugger

Provides an introduction to the Recipe Debugger interface.

Getting Started with Recipe Debugger

The Recipe Debugger (rdebug) is used to debug Proteus recipes. It is invoked from the command line by:

`rdebug [-options] job_control_file [pwb]`

where:

- *`job_control_file`* is the name of an existing job control file that has been processed by xmscript and proteus.
- `pwb` starts a copy of the Proteus WorkBench application.

Note: When using the Proteus WorkBench application as a viewer for rdebug, be aware that PWB has a switchable Proteus library. Consult the *Proteus WorkBench Command Manual* for details, as well as the *Proteus WorkBench User Guide*.

The following command-line options are available:

`-p` *`port`*	The network port number to connect to viewer. If the port is unavailable, up to 20 sequential ports are tried. The default port is 5000.
`-V` *`level`*	A number from 0 to 5 indicating the verbosity level of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. The default is 0.

`-stippleMode` *n*	Controls the stipple pattern used to display polygons on the viewer. 0 = uses PWB defaults 1 = rotates between 4 different stipple patterns 2 = rotates between 8 different stipple patterns
`-colorMode` *n*	Controls the fill color used to display polygons on the viewer. 0 = uses PWB defaults 1 = uses old-style colors
`-v`	Prints the executable version.
`-h`	Help. This option displays the text help on rdebug.
`-segmentTooltip`	Controls the display of segment information when using the Proteus WorkBench application. When set to on (the default), the ftype, polygon number, and segment number for polygons are displayed in the PWB window.
`-python_fast_trace`	Causes rdebug to run in an enhanced Python trace mode, which could reduce rdebug runtime. On by default.

The Recipe Debugger invokes the same correction engine as the corexec and dpserver programs, but it also includes an integrated interface that allows you to step through the correction recipe incrementally. The rdebug program starts one copy of the Proteus WorkBench application and one copy of corexec (nondistributed correction program).

The Recipe Debugger does not output results to a .gds file. It does create a .gds.TINF file and an empty .gds file if none exists, but does not overwrite existing files.

Upon invocation of rdebug, two windows appear on your workstation: the Proteus WorkBench window and the debugger interface (the debugger interface is shown in Figure 61 on page 332). By default, rdebug starts at the first valid template (the first template with graphics).

The debugger uses the Proteus WorkBench application to display graphical information. Using its interface, you can interact with the data in the Proteus WorkBench application to roam, zoom, model, save, or perform any other standard Proteus WorkBench function. The graphics sent to the Proteus

WorkBench application vary, depending on the debugger's location in the recipe.

Updating a Recipe During Debugging

You can modify a recipe during debugging as follows:

1. Update the compiled job control file with the modification.
2. Save the job control file.
3. Click the Restart button in the debugger.
4. Continue with the debugging process.

Recipe Debugger Window

When you invoke rdebug, the Recipe Debugger window appears. The following sections describe its components in the order they appear in the rdebug window.

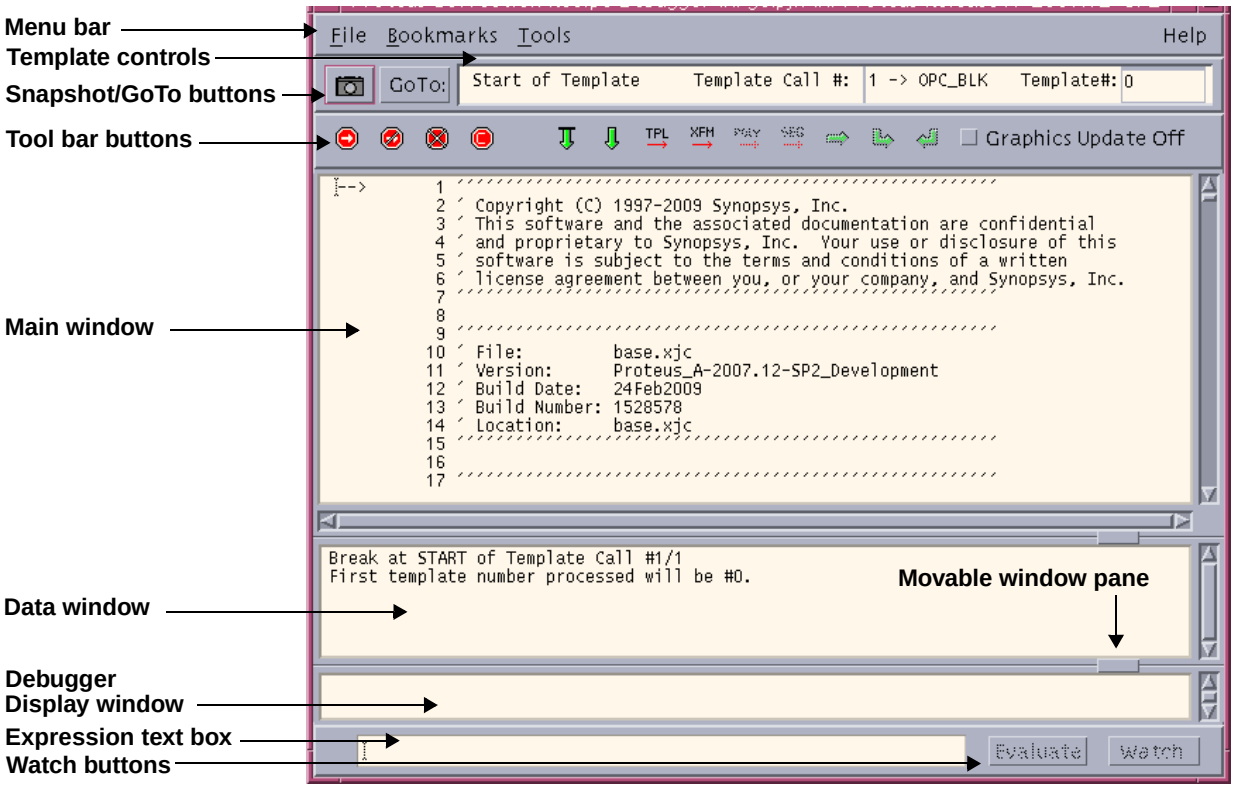


Figure 61 The Recipe Debugger Window

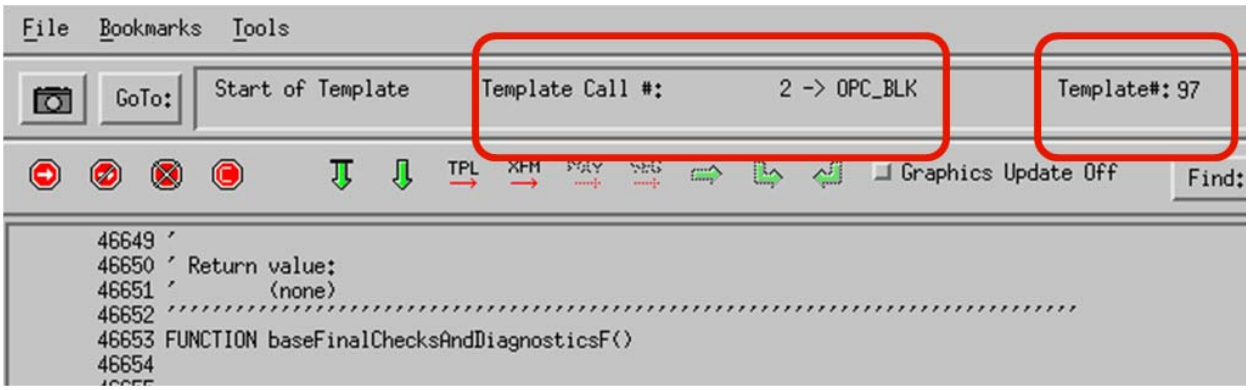
Menu Bar

The menu bar contains four pull-down menus.

File	Used to exit the software.
Bookmarks	Accesses the bookmark menu commands Add Bookmark and Delete All Bookmarks. See Bookmarks Menu on page 346.
Tools	Accesses the Graphics Window, Breakpoints, Layer Mappings, and Watch Window commands. See Tools Menu on page 347.
Help	Displays release and copyright information on the active version of the software. Provides access to the Proteus documentation (see the *Proteus Release Notes* for usage instructions).

Template Controls

The interface for controlling which `TEMPLATE_CALL` to debug is available in a pull-down menu at the top of the rdebug window (next to **Template Call #**), as shown in Figure 62.



```
'
' Return value:
'         (none)
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
FUNCTION baseFinalChecksAndDiagnosticsF()

```

Figure 62 TEMPLATE_CALL pull-down menu

By default, rdebug starts at the first valid template (the first template with graphics), not template 0. To run a different `TEMPLATE_CALL`, select it from the pull-down menu (you can also select a template number) and choose the **GoTo** button. The next time you choose XFM, the line pointer jumps down to the

beginning of the appropriate `TEMPLATE_BLOCK` and walks through it with consecutive button pushes.

When debugging the first `TEMPLATE_CALL`, the data that must exist is

- the input GDS/OASIS
- output from hierman FE and context analysis for the first template block

When debugging a `TEMPLATE_CALL` other than the first, the data that must exist is

- the output fragment files from the previous `TEMPLATE_CALL` (be sure you have `NO_FRAG_CLEAN` in your recipe during the prior run of the proteus binary)
- the temp files output by the context analysis for that `TEMPLATE_CALL`

In other words, context analysis and template generation must complete in order for the required temp files to exist for template block execution. An error message will be generated if the required files have not yet been generated by proteus.

Tool Bar Buttons

The following tool bar buttons are available. Hovering over any one of the tool bar buttons with the mouse displays the ToolTip for that button.

Snapshot and GoTo Buttons

The **Snapshot** and **GoTo** buttons allow you to specify your location in the recipe. Select your intended destination in the recipe from the **GoTo** list; you can choose **Start of Template**, **Template Block**, **CorBASIC**, or **CorBASIC Dissect**. (**CorBASIC Dissect** sets a system-defined breakpoint target. See System-Defined Breakpoint Targets on page 352.) After selecting a destination, click **GoTo** to jump to the defined recipe location. Click **Snapshot** (the camera icon) to fill in the Template, Ftype, Poly, and Face fields automatically.

Breakpoint Buttons

The breakpoint tool bar buttons are shown in Figure 63.

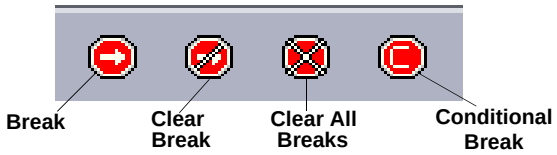


Figure 63 Breakpoint Tool Bar Buttons

Break

This button defines a breakpoint. To set the breakpoint, place the cursor on the line you want to mark, and click this button. (See Tools > Show Breakpoints on page 349.)

Clear Break

This button clears the breakpoint at the current line.

Clear All Breaks

This button clears all breakpoints you have defined in the recipe.

Conditional Break

This button sets a conditional breakpoint. For information on setting a conditional breakpoint, see Conditional Breakpoints on page 352.

Navigation Buttons

The navigation tool bar buttons are shown in Figure 64.

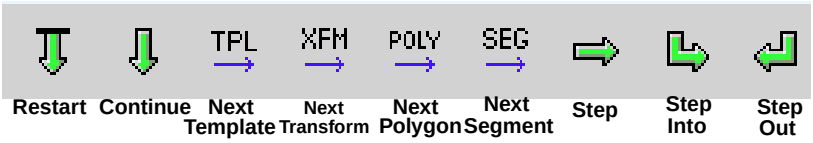


Figure 64 Recipe Navigation Tool Bar Buttons

Hot Keys

The following hot keys are available for use instead of the Navigation buttons:

- c = Continue
- t = Next Template
- x = Next Transform
- p = Next Polygon
- g = Next Segment
- n = Step (Next)
- s = Step Into
- f = Step Out

Restart

This button is equivalent to exiting and then restarting rdebug. After **Restart** is chosen, the current job status and data files are destroyed, the job control file is reread, and correction execution starts from the beginning. The job control file can be edited between restarts. The Hierarchy Manager can also be run between restarts in cases where you modify parameters (such as template size).

When you choose **Restart**, rdebug remembers the last `TEMPLATE_CALL` and the last template number that was being processed. It updates the `TEMPLATE_CALL` pull-down menu with the last `TEMPLATE_CALL` and the Template # entry box with the last template number that was being processed when you chose **Restart**.

During a restart, the locations of breakpoints are saved, then reset in the reloaded job file. If the job control file has not changed, breakpoints map to the original lines.

Continue

This proceeds to the next visible breakpoint, or until the end of the recipe is encountered. The Proteus WorkBench window is updated as specific commands in the recipe are read (such as `SHOW`, `DRAWBAR`, or `UPDATE_GRAPHICS`), stopping only when it reaches either a breakpoint or the end of the recipe.

Continue automatically continues to the next template.

Next Template

This proceeds to the beginning of the next template. The Proteus WorkBench window is updated as each line in the recipe is read, stopping only when either the next template or the end of the recipe has been reached.

Next Transform

This proceeds to the next major pattern transform. Breaks for major pattern transforms are defined as:

- Source group Booleans.
- Pre-crimped dissection patterns.
- The corBASIC program.
- Python flow statements.

Next Polygon

This proceeds to the next `SET_POLY_NO` function call and performs that operation, so the pointer is on the line following `SET_POLY_NO`. This is active only during corBASIC debugging.

Next Segment

This proceeds to the next `SET_SEG_NO` function call and performs that operation, so the pointer is on the line following `SET_SEG_NO`. This is active only during corBASIC debugging.

Step

This executes the current line. This is active only during corBASIC or Python flow debugging.

Step Into

This steps into a corBASIC function, or into a `PYTHON_MODULE` or a user-defined function when debugging a Python recipe.

For example:

```
--> temp = protosolveF ( 4 )
CORWT = CORWT + temp
```

Step executes the current line (assign a value to temp), then moves the current line indicator to: `CORWT = CORWT + temp`.

Step Into enters the `protosolveF` function. The current line indicator jumps to the first line of the `FUNCTION protosolveF()` declaration.

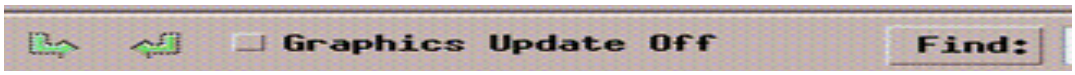
The Step Into button is active only during corBASIC or Python flow debugging.

Step Out

This proceeds until the current corBASIC function or `PYTHON_MODULE` returns. This is active only during corBASIC or Python flow debugging.

Graphics Update

If you invoke rdebug with the Proteus WorkBench application as your viewer, the **Graphics Update Off** check box appears on your toolbar next to the **Step Out** button.



Check this box if you do not want the recipe viewer to update automatically as you step through the recipe. The function is on by default.

Note: If a line has a breakpoint set on it, graphics *will* be updated in the Proteus WorkBench application, even if the **Graphics Update** box is unchecked (set to Off).

If you have a large template, turning this functionality off might save you time as you move to the desired location in your recipe. You can then turn the graphics update back on when you reach the desired location. The graphics that were displayed when the box was checked will remain in the view until an `UPDATE_GRAPHICS` command is reached in the recipe or typed into the command line in the corBASIC part of recipe execution.

Find

Click the **Find** button to locate the text string specified in the field to the right of the button. The search begins at the start of the recipe and highlights the first located instance. Repeatedly clicking the **Find** button sequences through all instances of the specified text string.

The search box accepts up to 1,024 characters. The default text search is case sensitive. However, if the **Case Insensitive** check box is selected, the search

performed through the **Find** operation is case-insensitive. The value of this check box is saved when you exit rdebug, so that when you re-invoke rdebug it remembers the value of this check box from the previous session.

You can search the file forward or backward by clicking one of the radio buttons shown in Figure 65.

Figure 65 Find Button

Main Window

The Main Window contains the job file script. Lines are numbered from the first line in the job file (line #1). The arrow symbols (-->) mark the current line of the debugger.

Data Window

The Data Window contains current state information for the recipe. The type of information displayed varies depending on whether the current debug line is within the corBASIC recipe or the Boolean process groups. In Figure 61 on page 332, the data window displays register values such as the current template number, current `CORWT`, current `TYPE`, and so forth.

Once the Recipe Debugger is stepping through the script, you can double-click on a variable in the Data Window to select and automatically paste it into the Watch tool bar expression text box (located in the lower-left corner of the Debugger window). You must identify a specific element of an array when using the debugger for array definitions.

Debugger Display Window

The Debugger Display window can display values for any variable in the program.

Expression Text Box

Use this field to enter variables, expressions, or functions to evaluate. Variable names can be written as either *`functName.VarName`* or simply *`VarName`*.

For example, enter the variable name `slope`. The search for this variable is done in two steps:

1. Search for `slope` as a `GLOBAL` variable name.
2. If `slope` is not found, and no function name was specified, prepend the current function name. For this example, search `protosolve.slope` as local variable name.

Some typical uses of the expression text box include:

- entering a variable name to determine its current value
- entering an expression to evaluate
- entering a function to execute
- entering the `SHOW` function to observe graphical data in the Proteus WorkBench window
- modifying the variable name to its effect on correction

Debugger Expression Evaluations

The expression is always evaluated within the scope of the current function. If the debugger is currently stopped in the `myfunc` function and you evaluate the expression `c=a+b`, the local variables `myFunc.a` and `myFunc.b` are used. The evaluation window opens and displays the current function name, the evaluated expression, and the result. When an expression is placed within the Debugger Watch window, it is evaluated within the scope of the original function and labeled.

Valid expressions include the = assignment operator. Therefore, you can use the evaluation window to specify the register values directly by defining an assignment expression.

Function calls can also be valid expressions. For example, you can write a function to write the `SEG_NO` and `CORWT` values for every segment to an output file using the `PRINTLINE` function. Then you can execute this function at will from the debugger evaluation window.

Note: Do not enter an assignment expression in the Debugger Watch window. Because the Debugger Watch window updates whenever the program breaks and control is turned over to the Recipe Debugger, assignment expressions in the Debugger Watch window are likely to be overwritten. (See Watch on page 345.)

SHOW_* corBASIC Functions

When using corBASIC, you can use the `SHOW`, `SHOW_POINT`, `SHOW_POLY`, and `SHOW_SEG` functions to display graphics during recipe debugging. See the *corBASIC Reference Manual* for more information on these functions.

The `SHOW_*` functions can be used directly in the evaluation window (see Debugger Expression Evaluations on page 340). They can also be embedded at permanent fixed locations in the corBASIC recipe.

When you embed a `SHOW_*` function in the recipe, it is executed only when you are in debugging mode. Because of this, `SHOW_*` functions do not impact execution speed when performing real corrections.

Note: The screen is redrawn each time the `SHOW` function is called. This can cause a significant lagtime in the debugging processing if the function is called multiple times.

The `SHOW_*` commands can be used in conjunction with each other to display the segments in various ways. For example, if you enter the following in the Debugger Watch Window (see Watch on page 345):

```
SHOW (clr_anno)
SHOW (this_seg)
```

the old segment is erased and the current segment highlighted.

The `SHOW` command can also take a *`this_poly`* argument, which will display the current polygon (without `CORWT` adjustments).

`SHOW_POINT(`*`x, y [, DBU`*`])` displays a large X at the designated coordinates. The optional *`DBU`* argument controls scaling of the coordinates.

`SHOW_POLY(`*`ftype, poly_no)`* highlights any polygon.

`SHOW_SEG(`*`ftype, poly_no, seg_no`*`)`highlights any segment.

SHOW_* Commands in the Watch Window

You can use the `SHOW_*` commands in the Debugger Watch Window to view particular segments and protosegments.

For example:

1. Enter **SHOW (clr_anno)** in the Watch Tool Bar field to clear the annotation figures.
2. Enter **SHOW (this_seg)** in the Watch Tool Bar field to highlight the current segment. The segment is highlighted in the Proteus WorkBench window.
3. Enter **SHOW (this_ps)** in the Watch Tool Bar field to highlight the current protosegment. The protosegment is highlighted in the Proteus WorkBench window.
4. Enter **SHOW (this_poly)** in the Watch Tool Bar field to highlight the current polygon. The current polygon is determined based on the current values of `FTYPE` and `POLY_NO`. It is highlighted in the Proteus WorkBench window through a change in color.
5. Enter **SHOW_POINT (*x*, *y* [, *DBU*])** in the Watch Tool Bar field to display a large X at the chosen coordinates. The optional *DBU* argument controls scaling of the coordinates.
6. Enter **SHOW_POLY (*ftype*, *poly_no*)** in the Watch Tool Bar field to highlight any polygon. The polygon chosen is highlighted in the Proteus WorkBench window through a change in color.
7. Enter **SHOW_SEG (*ftype*, *poly_no, seg_no*)** in the Watch Tool Bar field to highlight any segment. The segment chosen is highlighted in the Proteus WorkBench window by drawing lines on either side of it.

show* Python Functions

When using Python, you can use the `showLayer` function to display the requested Layer or Edge object in the IC WorkBench Plus application. Use the `showPoint` function to display a large X at the designated coordinates.

You can include a `show*` function directly in the Python recipe or type it into the expression text box of rdebug, then click Evaluate. See the *Python in Proteus User Guide* for more information on these functions.

Examples

```
showLayer(layer1, 12, True)
```

```
showPoint(150, 150, dbu = 2)
```

Show and Remove Protoboxes

You can use the following SHOW_* or CLEAR_* commands in the Debugger Watch Window to display or remove the protoboxes on the viewer:

SHOW_ALL_PROTOBOXES

Description

Shows all protoboxes in rdebug. For additional information, see also Protobars and Protoboxes and Protobar and Protobox Registers in the *corBASIC Reference Manual*.

Syntax

```
SHOW_ALL_PROTOBOXES()
```

SHOW_PROTOBOX

Description

Shows an individual protobox in rdebug. For additional information, see also Protobars and Protoboxes and Protobar and Protobox Registers in the *corBASIC Reference Manual*.

Syntax

```
SHOW_PROTOBOX(pid)
```

Arguments

pid

The return value from `CREATE_PROTOBOX`.

CLEAR_ALL_PROTOBOXES

Description

Clears all protoboxes in rdebug. For additional information, see also Protobars and Protoboxes and Protobar and Protobox Registers in the *corBASIC Reference Manual*.

Syntax

```
CLEAR_ALL_PROTOBOXES()
```

CLEAR_PROTOBOX

Description

Clears an individual protobox in rdebug. For additional information, see also Protobars and Protoboxes and Protobar and Protobox Registers in the *corBASIC Reference Manual*.

Syntax

```
CLEAR_PROTOBOX(pid)
```

Arguments

pid

The return value from `CREATE_PROTOBOX`.

Watch Buttons

The Watch buttons appear at the bottom of the main window.

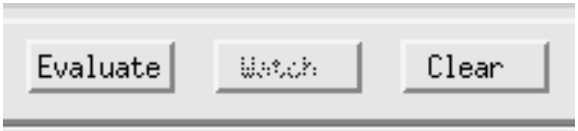


Figure 66 Watch Buttons

Evaluate

Click **Evaluate** to calculate the variable, function, or expression value (corBASIC or Python) and print the result to the main window. The value is printed only once, and it is not continuously updated.

This function is the equivalent of pressing the Return key.

Watch

Click **Watch** to access the Debugger Watch Window from the main window. The variable name is added to a table of variables to monitor. This table of names and values reflects the current state of the program. Values are updated each time the program breaks.

From the Tools menu, **Show Watch Window** also invokes the Debugger Watch Window, evaluating all the variables being watched and displaying current values. When rdebug is first invoked, the values are 0.

Note: Expressions placed in the watch window are also continuously updated. For example, in the following sequence, if `i=3` is placed in the watch window, the loop never exits because the value of `i` is always 3.

```
FOR i = 0 to 7

>>> .....

NEXT i
```

Clear

Click Clear to clear the window showing the results of an evaluation.

Menu Commands

The following sections describe the menus accessible from the Recipe Debugger menu bar.

Bookmarks Menu

When you choose **Bookmarks** from the menu bar, the Bookmarks menu appears. Use the Bookmarks menu commands to add, delete, and access additional bookmarks in the recipe.

Bookmarks > Add Bookmark

Use bookmarks to advance quickly to a specific template number, iteration number, polygon number, and segment number.

To define bookmarks:

1. Step to a point of interest, such as the corBASIC `protosolveF` function, template 3, iteration 2, polygon 13, segment 7.
2. Set a breakpoint in the `protosolveF` function with the red **Set Break** button.
3. Choose **Bookmarks > Add Bookmark**. A new menu option is added to the Bookmarks menu, entitled: Template 7; corBASIC Recipe Iter 2; Polygon 13; Segment 7.
4. Perform your debugging.

Note: The rdebug application fails when a bookmark is added to a `PROTEUS_JOB_FLOW` recipe and you try to access the bookmark in a subsequent run.

Bookmarks > Delete All Bookmarks

To delete the bookmarks in a file choose **Bookmarks > Delete All Bookmarks**.

Bookmarks > Template <n>: <bookmark>: [GoTo|Delete]

To go to a specific bookmark, choose **Bookmarks > Template <n>:<the desired bookmark> > GoTo**. The **GoTo** command is equivalent to choosing **Restart**, then stepping manually until you reach the specified template, iteration, polygon, and segment.

Each time you go to a bookmark, you are restarting the correction process, even if the bookmark location is later in the correction than your current debug line. If the identical bookmark cannot be matched (perhaps because the job file

has been modified) the bookmark proceeds until it reaches the end of correction.

To delete a specific bookmark, choose **Bookmarks > Template <n>:<the desired bookmark> > Delete**.

Tools Menu

When you choose **Tools** from the menu bar, the Tools menu appears. The following sections describe the commands accessible from the Tools menu.

Tools > Open Graphics Table

When you choose **Tools > Open Graphics Table** from the menu bar, the following window appears.

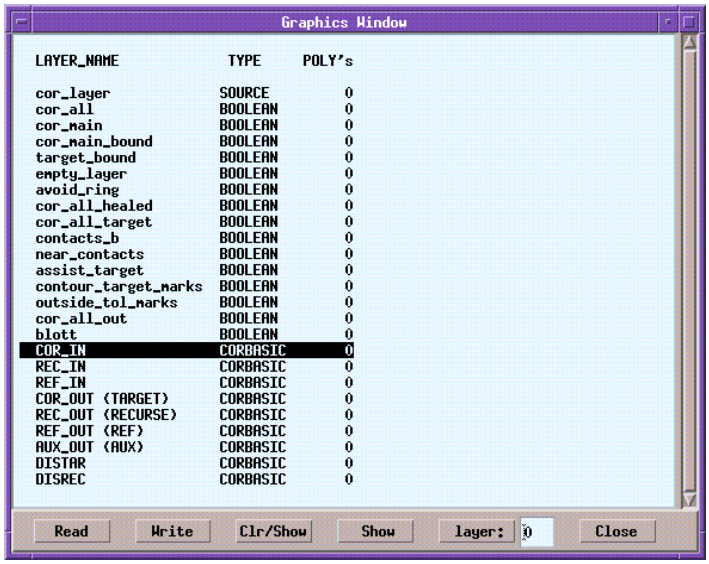


Figure 67 The Graphics Window

This window displays all named graphical layers, protoboxes, and protobars in the recipe. The last column in the table indicates the number of polygons in that named layer.

This window always represents the current state of the recipe. At the initial break at the start of the recipe, all layers contain zero polygons. As you step through the recipe, you see the polygon count change accordingly.

The graphical shapes contained within these layers also change in real-time as the recipe is executed. At any point where the recipe breaks, you can use this window to select a graphical layer and observe the current graphics by clicking Show.

Read

Click this button to read a .vrt format file and overwrite the data in a selected layer with the data from that file.

Sometimes it is useful during debugging to insert simplified graphics into the correction recipe. This is analogous to using the evaluation window to assign a value of your choice to a corBASIC variable. However, the **Read** button assigns graphics of your choice to a named layer.

To use this function:

1. Select a named layer by clicking on it with the mouse.
2. Click **Read** and select a graphics file (in .vrt format) to read. This new data overwrites the existing layer graphics.

The **Read** button is disabled during corBASIC recipe execution. Graphic elements cannot be reassigned during this phase of the correction.

Write

Click this button to write the selected layer to a .vrt file.

Clr/Show

Click this button to clear the Proteus WorkBench window, then display the selected layer graphics.

Show

Click this button to send the selected layer graphics to Proteus WorkBench for display. This button does not clear the Proteus WorkBench graphics, so the new data overlays the existing layers.

You might wish to change to Window B or Window C before clicking **Show** to avoid overlaying the pattern.

Layer

Specify the layer on which to put the data from the graphics table in the Proteus WorkBench window. To use this function:

1. Specify the layer by selecting the desired one from the Layer Name list.
2. In the **Layer:** field, enter the layer on which you want the data placed.
3. Click the **Clr/Show** or **Show** button to execute the command and place the desired data from the graphics table on the specified layer in the Proteus WorkBench window.

Close

Click this button to close the Graphics Watch window.

Tools > Show Breakpoints

Breakpoints are specific locations in the recipe where the program interrupts its execution. The debugger uses two basic types of breakpoint: user-identified breakpoints, and system-defined breakpoint targets. Use the debugger interface to set breakpoints for the recipe. (See also Breakpoint Buttons on page 335.)

You can scroll through all the breakpoints, user-identified or system-defined, from the Break Point Watch Window. Open this window from the menu bar **Tools > Show Breakpoints**.

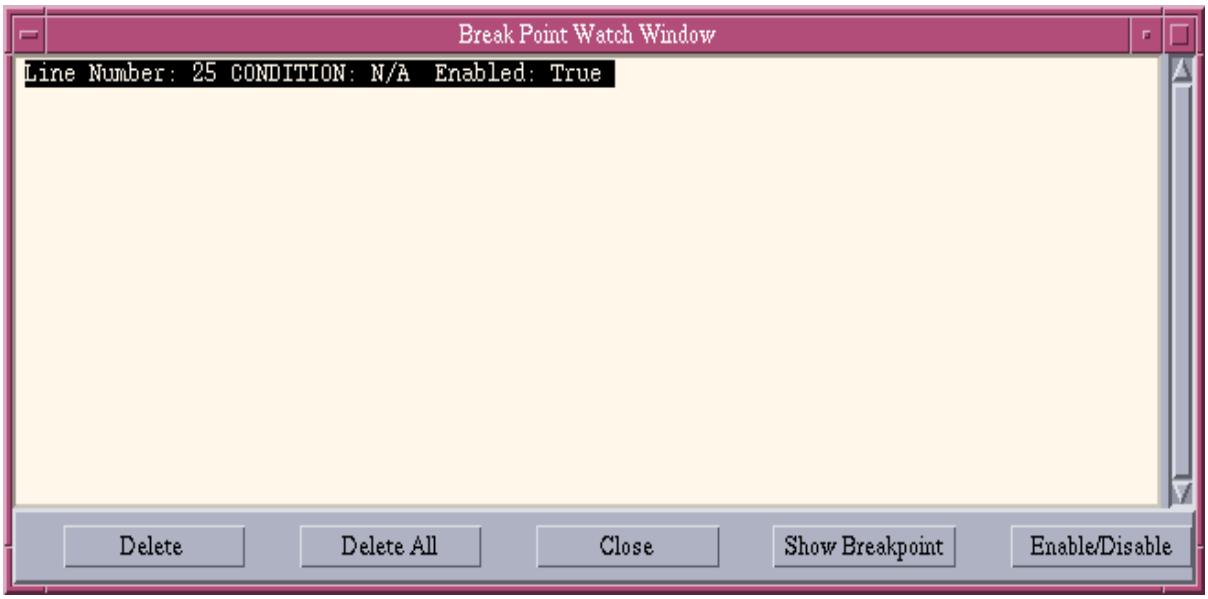


Figure 68 Break Point Watch Window

From this window, you can see a list of all breakpoints. By selecting a specific breakpoint and clicking **Show Breakpoint**, the main rdebug text window shows the line of the breakpoint. You can delete a specific breakpoint by selecting it from the list and clicking **Delete**. Alternately, you can click the **Delete All** button to delete all breakpoints. You can also enable or disable a breakpoint using the **Enable/Disable** button.

User-Identified Breakpoints

You can set user-identified breakpoints at any location in the recipe. Exercise care when doing so, because if you place a breakpoint on a line that is commented out, or is part of a flow control block (for example, an “if” block) that is not executed, when you start the debugger these breakpoints are bypassed during execution.

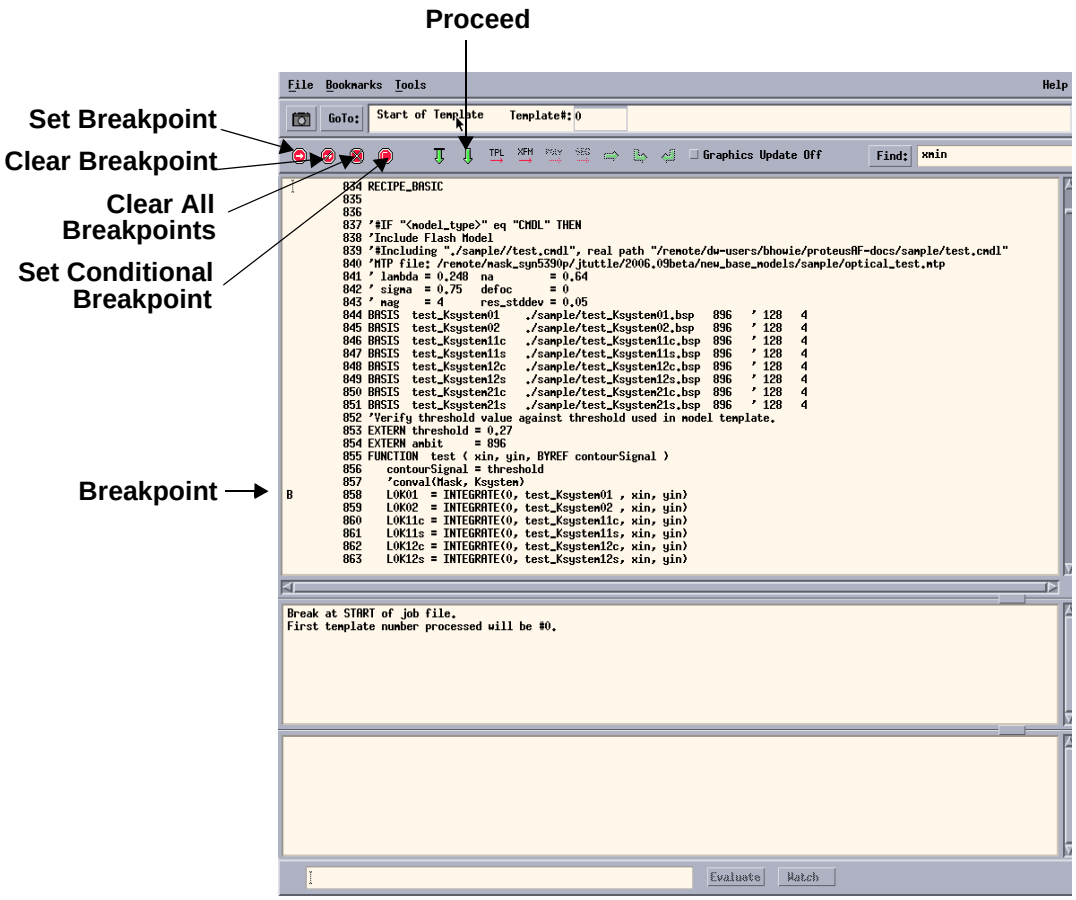


Figure 69 Breakpoint Tool Bar Functions

To set a user-identified breakpoint:

1. Select a line by clicking on it once in the main text window.
2. Click the first tool bar button, **Set Breakpoint**. As the program is executing, every executed line checks for the presence of a breakpoint. The break in the program occurs before the line is executed. If you put a breakpoint on a line that is never executed (for example, a comment or the `END_RECIPE` keyword), the breakpoint is never reached and the program does not break.
3. Repeat this process to insert as many breakpoints as are needed in the recipe.
4. Once you have inserted the breakpoints, click the **Proceed** button (the downward pointing green arrow) to begin program execution at the current line and continue execution until a breakpoint is encountered, or until the

end of the recipe is reached. The debugger does not exit at the end of a recipe, but waits for the debug and correction recipe to be restarted. To exit the debugger, use the **File > Exit** menu command.

Conditional Breakpoints

Conditional breakpoints identify lines at which the debugger will stop if specific conditions are met. When these points meet the specified criteria, the debugger stops and sets a breakpoint.

To set a conditional breakpoint:

1. Enter the condition the point needs to satisfy in order to stop debugging execution in the command-entry field at the bottom of the interface.
2. Select the line on which to set the breakpoint.
3. Click the conditional breakpoint button in the toolbar. A “C” appears at the beginning of the selected line. A message is output in the watch window indicating that you set a conditional breakpoint.
4. Start the execution of the debugger. Execution will stop if the expression is true (non-zero) on the line where the breakpoint is set.

To delete a conditional breakpoint, use the delete breakpoint button available on the toolbar. You can modify the breakpoint (update the condition) by selecting the appropriate line, adding a new condition in the text window, and clicking the conditional breakpoint button again. A new message is generated indicating the conditional breakpoint is updated.

Note: Conditional breakpoints are line-specific; a line has to be chosen before you can set a conditional breakpoint.

System-Defined Breakpoint Targets

There is one “system-defined” breakpoint target available from the **GoTo** field: **corBASIC Dissect**. This target appears as shown in Figure 70.

Figure 70 GoTo System-Defined Breakpoint Target

After selecting the **corBASIC Dissect** breakpoint target, clicking **Proceed** executes the recipe until it reaches either the next user-identified breakpoint, the next occurrence of a `DISSECT_TEMPLATE()` function call, or the end of the recipe. The debugger then resets the **GoTo** list to **corBASIC** so that repeatedly clicking **Proceed** advances to the next occurrence of the `baseMainF()` function in the recipe.

You must manually reset the **GoTo** list to **corBASIC Dissect** before being able to advance to the next breakpoint `DISSECT_TEMPLATE()` function call.

As with user-identified breakpoints, only system-defined breakpoint targets that are neither commented out, nor part of a conditionally excluded logic flow, execute.

You can specify each of the following control parameters:

Template

Enter an integer to identify which template (if the recipe contains more than one) to target for the breakpoint.

Ftype#

The Ftype# parameter lets you specify a figure type that can be dissected. Enter 0 for a `COR` figure; enter 1 for a `REC` figure.

Poly#

Enter an integer to identify `DISSECT_TEMPLATE()` function calls that contain a `POLY_NO` value equal to the specified integer.

Face#

Enter an integer to identify the number of segments into which a face or edge will be dissected.

Tools > Show Layer Mappings

The Layer Mapping window allows you to view which layers or named user types are associated with numbered user types. To open the Layer Mapping Window from the menu bar, choose **Tools > Show Layer Mappings**.

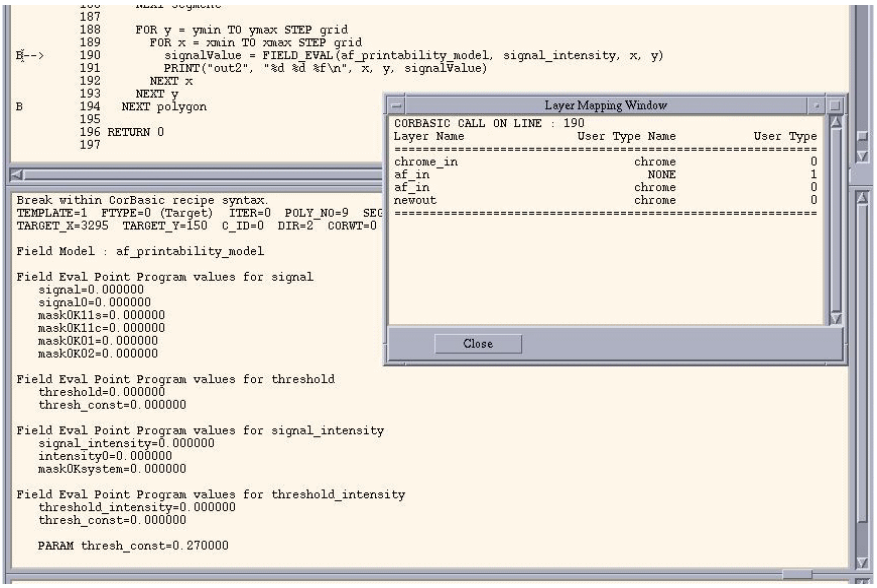


Figure 71 The Layer Mapping Window

The Layer Mapping Window provides the line on which the corBASIC call was made along with the variable information (layer name, user type name, and user type for the specific XMDL model).

Tools > Show Watch Window

The Debugger Watch window is available only during the corBASIC debugging process, and allows approximately 20 expressions. It appears upon the first Watch execution (see Watch Buttons on page 344).

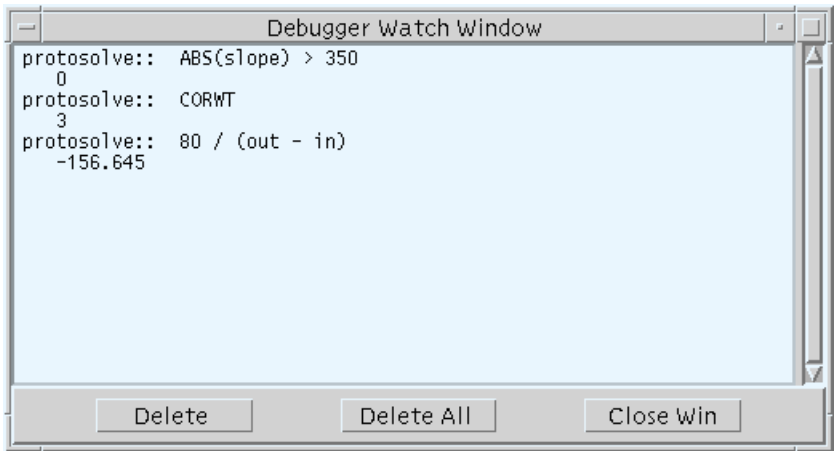


Figure 72 Debugger Watch Window

Delete

Click **Delete** to remove a variable or expression from the Debugger Watch window.

To remove a variable:

1. Select the variable to remove with a single mouse click.
2. Click **Delete**. The line is removed from the window.

Delete All

Click **Delete All** to clear the table of all watched variables and expressions.

Close Win

Click **Close Win** to close the Debugger Watch window. The table of watched variables and expressions remains intact and will be visible the next time the window appears.

Tools > Show Call Stack

Choose Tools > Show Call Stack to display a window showing the current call. This displays the stack in both Python and corBASIC code. Select an entry in the stack trace and choose Show, which switches scope to that level in the call stack. You can then view variables in this scope by entering them in the Evaluate window. Choosing Return in the stack trace window will bring the scope back to the current function being executed.

Note that the Show Call Stack feature for corBASIC code is slightly different from that displayed for Python code. For corBASIC code, when you select an entry in the stack trace and choose Show, you are directed to the location where the function is being called, not the function definition.

XMDL Model Point Program Variables

To examine XMDL model point program variable values when debugging XMDL models, step into the `FIELD_POINT_EVAL` function to display the values of these variables prior to point program execution in the main rdebug window. Stepping past the `FIELD_POINT_EVAL` function shows the resulting point program variable values after the point program execution.

Usage Examples

The following examples demonstrate how to use the Recipe Debugger in various situations.

Examining the Postcorrection Results

Suppose you wish to examine the final .gds file and debug an uncovered error. Use the following general procedure to examine the results and make necessary corrections:

1. Use Proteus WorkBench to locate the (x, y) position of the defect.
2. Use the `-b` option in Celltool to determine the templates that interact at the position. Or use the Proteus WorkBench application to determine which templates interact at the position.
3. Use the Recipe Debugger to single-step through templates to identify either the exact template or multiple templates (such as template notches) where the problem resides.

4. Identify the problem as a Boolean or corBASIC recipe problem. If it is a Boolean problem, single-step to the Boolean causing the error. If it is a recipe problem, identify the position within the template, click the **Next Polygon** and **Next Segment** buttons to step through the recipe, locate the segment with the error, then debug the recipe logic.
5. Click the **Next Template** button to move to the next template if multiple interactions occur, and repeat the debugging process.

Changing Variable Values

You can modify the values of variables in the Watch Tool Bar field.

To modify the value of a variable in the recipe:

1. Locate and double-click on the variable you wish to modify. It appears in the Watch Tool Bar field.
2. Enter **= *value*** to the right of the variable you have selected
3. Click **Evaluate**.

The new value is assigned to the variable.

9

Proteus Applications

Covers utilities and applications used with the Proteus tool, including UNIX binary applications and Perl script applications.

Utilities

While some of these applications are Perl scripts, most are UNIX binaries.

Most utilities have small help descriptions that can be accessed from the UNIX command line. These descriptions typically contain a brief list of options, as well as a simple overview of what the specific utility can do. This help information is accessed by adding the -h option at the end of the command line. For example, if you enter **initgds -h** at the command line, the following information appears in the xterm:

```
Version D-2010.06-16

Copyright <C> 1995 - 2012 Synopsys, Inc.
This software and the associated documentation are
confidential and proprietary to Synopsys, Inc.
Your use or disclosure of this software is subject to the
terms and conditions of a written license agreement
between you, or your company, and Synopsys, Inc.

Usage: initgds [-options] <job control file>
[<gdsfilename> [<libraryname>]]

   Used to initialize a GDS file.

   <job control file> - the output of xmscript.
   <gdsfilename> - the name of the GDS file to initialize
                   default OUTPUT_FILE parameter in
                   <job control file>
```

```
   <libraryname> - the name of library given in the gdsfile
                   default CORLIB parameter in
                   <job control file>
NOTE: if <libraryname> is specified, <gdsfilename> must be
specified.
options:
      -s         - suppress printing of syntax warnings.
      -v         - prints executable version and exits.
      -h         - help, print this message.
```

UNIX Binary Applications

The following applications are available directly from the UNIX prompt:

bin2asc	Converts Proteus binary files to ASCII.
celltool	Executes phases of the correction process on a template. See Chapter 7, Celltool, for more information on Celltool.
closegds	Specifies the end of a .gds file under construction.
corexec	Runs a linear correction job. See Chapter 5, Correction, for more information on corrections.
count_graphics	Counts polygons in a GDSII or OASIS file and outputs the flat and hierarchical counts for those polygons for each layer/datatype.
dpconsole	Changes the number of DP workers (dpservers) associated with a DP controller such as proteus.
dpserver	Performs correction work in a distributed correction process. One or more dpservers connects to a DP controller (proteus) to perform a correction. See Chapter 6, Distributed Processing, for more information on dpserver.
gds2vrt	Converts GDS or OASIS test patterns to .vrt format for input to the Proteus WorkBench application or the ProGen tool. This is intended for small test patterns only, as unpredictable behavior may result if large file conversions are attempted.

gdsmerge	Combines two GDS libraries from separate stream files.
gdssplit	Extracts selected portions of a .gds file based on layer and datatype, and/or cell name.
hierman	Invokes the Hierarchy Manager. See Chapter 4, Hierarchy Management, for more information on hierman.
initgds	Writes the header information into the .gds file.
mktop	Builds the cell hierarchy and places it in the output .gds file.
oasmap	Maps specified layer and datatype information from a source OASIS file to a destination OASIS file.
oasmerge	Combines two OASIS hierarchies from separate OASIS files.
oassplit	Extracts selected portions of an OASIS file and places them in a user-specified destination file.
printlog	Reads binary log file data and produces a formatted ASCII text file.
proteus	Supports hierarchical processing and distributed pipeline processing.
puf2vrt	Converts .puf format graphics files to .vrt format graphics files.
pwb	Invokes the Proteus WorkBench application. See the *Proteus WorkBench User Guide* for more information.
rdebug	Invokes the Recipe Debugger. See Chapter 8, Recipe Debugger, for more information on rdebug.
remote_server	Starts a dpserver on the specified node. See Chapter 6, Distributed Processing, for more information on remote_server.
vert2puf	Converts .vrt format graphics files to .puf format graphics files.

vrt2gds	Converts .vrt format graphics files to GDSII format files.
xmscript	Builds output job files under control of the master job script. See Chapter 2, xmscript Directives, for more information on xmscript.

Note: The following applications are available for the OASIS file format: celltool, corexec, dpserver, gds2vrt, hierman, oasmap, oasmerge, oassplit, proteus, rdebug, and xmscript.
Note that, despite the name referring to GDS, gds2vrt automatically detects the type of input file and converts it.
No other executables support OASIS.

bin2asc

This utility converts Proteus binary files to ASCII for troubleshooting and analysis.

`bin2asc [-options] binary_filename [text_file].`

where:

- *`binary_filename`* is the name of the binary file to convert to ASCII.
- *`text_file`* is the target ASCII file name.

The available command-line options are:

`-e` *`end`*	The ending record number. Default end is last record.
`-f`	Flattens records to one line each.
`-h`	Help. This option displays the text help on bin2asc.
`-s` *`start`*	The starting record number. The default start is 0.
`-S`	Prints out the contents of the .STAT file.

The output format is not static, and can change at any time.

celltool

This utility executes phases of the correction process on a template.

`celltool [-options] job_control_file`

where:

- *job_control_file* is the output of xmscript.

Refer to Chapter 7, Celltool, for more information on Celltool, including command-line options.

All Celltool operations require valid hierarchy management files run prior to execution. Valid hierarchy management files are generated and saved when the proteus binary has completed at least one run on the recipe that contains the `NO_FRAG_CLEAN` keyword.

closegds

This utility marks the end of a .gds file. When invoked, closegds tags an `ENDLIB` structure to the end of the current .gds file under construction, padding it such that the .gds file is constructed in increments of 2-K blocks.

`closegds gdsfile`

where:

- *gdsfile* is the current .gds file under construction.

corexec

This utility runs a linear correction job on a single CPU, where each template is corrected in turn. Maintenance utilities such as initgds, closegds, and mktop are all executed by corexec.

The corexec application can be run only when valid hierman files exist. Valid hierman files are generated and saved when the proteus binary has completed at least one run on the recipe that contains the `NO_FRAG_CLEAN` keyword.

Note: Running corexec on one template could result in corrupt OASIS output that cannot be opened by the Proteus WorkBench application.

```
corexec job_control_file [start_template] [end_template]
   [-tb n] [-no_output] [-pdb]
```

where:

- *job_control_file* is the output of xmscript.
- *start_template* is the template number at which to resume correction.
- *end_template* is the template number at which to end correction.

The available command-line options are:

`-tb n`	Specifies a specific template block to be executed. Only the specified template block will be executed. Running corexec on a `PROTEUS_JOB_FLOW` recipe without the `-tb` argument causes the execution to run on `TEMPLATE_BLOCK 1` by default.
`-no_output`	Prevents corexec from generating output files, such as GDS/OASIS and TINF files. Useful when you do not want to overwrite the existing output files during development.
`-pdb`	Invokes the Python debugger (pdb). For details, refer to the *Python in Proteus User Guide*.

count_graphics

This utility counts polygons in a GDSII or OASIS file and outputs the flat and hierarchical counts for those polygons for each layer/datatype. Results are printed to standard output (stdout).

The “hierarchical” count is a count of every instance of graphics that occurs in the original file. If any polygon has any repetitions via cell references or cell arrays, it will be counted only once.

The “flat” count is derived by resolving each cell reference multiplying the total number of polygons in the cell by its total number of references.

Note: Only polygons that are native to the input file are represented. No processing is done to merge any abutting polygons, and any path with a zero width is ignored. This applies to both the hierarchical and flat counting methods.

```
count_graphics input_file
```

where:

- *input_file* is a GDSII or OASIS file.

The available command-line options are:

-h	Help. This option displays the text help on count_graphics.

dpconsole

This utility changes the number of DP workers (dpservers) associated with a DP controller, such as proteus. Using dpconsole, you can tell a DP controller to release some of its dpservers or to spawn new ones without having to rsh to a specific host and kill or start the dpserver yourself. Dpconsole should simplify the process of changing the number of dpservers from previous scripts and techniques. In addition, you can have the DP controller checkpoint and shut down a dpserver when it finishes a template, preserving all the work it performed since its last checkpoint.

Note: Make sure to invoke the same version of dpconsole as proteus; if the version of proteus does not match the version of dpconsole you are connecting to it, dpconsole will be rejected and exit.

The dpconsole connects to the DP controller in the same way as dpserver. It can only connect to proteus during the same time frame that dpservers can; for example, after proteus prints that it has opened a port such as `success opening port:2346`, and before the job is done and proteus no longer accepts dpserver connections.

Also note that keeping a dpconsole connection open to proteus prevents proteus from completely finishing; exiting dpconsole allows proteus to complete its job.

When using dpconsole with a pipeline recipe, proteus does not respond to dpconsole commands immediately during context analysis, but stores and processes them as soon as the next template block begins correcting templates.

If you want to kill an entire job, the supported method is to use Ctrl-C on the proteus client. When using dpconsole, you should keep at least one dpserver active on the proteus job to which you are connected.

`dpconsole [-options]`

The available command-line options are:

`-c` *`hostname`*	The DP controller's hostname to which dpconsole connects. The default is the same host where dpconsole is running, so you do not need to use this parameter if you are running dpconsole on the same host as the DP controller (proteus).
`-p` *`port_number`*	The port number to which to connect. If you do not specify this value, dpconsole starts at the first default proteus port (2346) and tries all port numbers up to the maximum (2445) until it is able to connect. You need to use this parameter only if you are running more than one DP controller on the same host and want dpconsole to connect to the one using the higher port number, or if you do not want dpconsole trying to connect to those ports on the DP controller host until it finds a valid one.
`-h`	Help. This option displays the text help on dpconsole.
`-V` *`level`*	A number from 0 to 5 indicating the verbosity of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. The default is 3, where all normal output is printed.

dpconsole Commands

After launching dpconsole, you can use the following available commands:

`?`	Prints a list of available commands and their usage.
`exit`	Tells dpconsole to disconnect from the proteus client and exit. It will not affect proteus or any of the dpservers. dpconsole waits for proteus to process all commands it has sent before exiting. If proteus has not processed all dpconsole commands (for example, if proteus is doing context analysis), then dpconsole waits until proteus processes them to exit. You can cause dpconsole to exit immediately using Ctrl-C, but any commands that have not yet been processed by proteus might not be executed once dpconsole is shut down.
`spawn [`*`host_name`*`| `*`remote_server_request`*`]`	The spawn command causes the proteus to immediately send a dpserver request to the remote_server command. It takes the same values that you would pass to proteus with the `-s` parameter, and the dpserver is started the same way as if you had given that value to the proteus client using `-s` originally. The proteus client sends an update to dpconsole when the dpserver is actually running and ready for work with the connection number and host name of the dpserver; for example: `Dpserver #4 spawned on host_thirteen.` Keep in mind that if the remote_server request is a hostname, it means that one dpserver will be spawned on that host. If you would like to spawn multiple dpservers on a specific host at the same time (for example, three on host_one) use a command such as: `spawn host_one host_one host_one` See Examples on page 370 for an example of the use of the `spawn` command with *`remote_server_request`*.

`info[`*`hostname`*`|`*`dpserver_number`*`|`*`~amount_of_dpservers`*`|`*`amount_of_dpservers~`*`|`*`first_dpserver~last_dpserver`*`]`

Requests a list of the active servers that are connected to proteus. The list includes dpservers and dpconsoles. It is a snapshot of the current state of the servers at the time proteus processes the `info` command. Each entry lists the following information for a specific dpserver/dpconsole:

- *Con*: Connection number (also known as dpserver number) of that dpserver/dpconsole.
- *Act*: Active state. This is “Active” most of the time, representing an actively working dpserver/dpconsole. It might also be “Inactive” or “Pending Inactive” for a dpserver that is between template blocks or that does not have any work. It is “Exiting” for a dpserver that is in the process of shutting down but has not yet completed shut down.
- *Type*: Type is either “Correct Svr,” which is a dpserver, or “Resource Manager,” which is a dpconsole.
- *Host*: Name of the host on which dpserver/dpconsole is running.
- *Port*: Port number being used for communication. This is the same for all connections on the list.

At the end of the dpserver list, the total number of live dpservers is printed, along with the number of dpservers that are currently working. This is normally all of them, except at the end of a template block when some dpservers do not have any templates to correct.

In order to manage the amount of information provided by the `info` command, you can use optional parameters to customize which dpservers are reported. Using a dpserver number provides only a report about the dpserver with that connection number, and using a hostname provides a list of only the dpservers running on that host. Passing an amount N of dpservers prefixed with a tilde (~N) reports only the first N dpservers, while passing an amount suffixed with a tilde (N~) reports only the last N dpservers. Passing in a range of dpservers (X~Y) reports the Xth through the Yth dpserver, including both the Xth and Yth dpserver.

`kill [hostname|dpserver_number|~amount_of_dpservers|amount_of_dpservers~]`

Gets rid of certain dpservers as quickly as possible by immediately sending an EXIT message to them, without preserving any of the work they did since their last checkpoint. For example, you might use the `kill` command to clear a host for use with another proteus client. The killed dpservers will exit (cleanly) as soon as they process the EXIT message, although processing time may vary.
You must provide at least one *`hostname`*, *`dpserver_number`*, or *`amount_of_dpservers`* (prefixed or suffixed with the tilde (~)).

- Using a *`dpserver_number`* (the connection number listed in the `info` command) causes proteus to kill that dpserver. If you specify the connection number of the dpconsole you are using, the dpconsole closes the connection and exits.
- Using a *`hostname`* causes proteus immediately to kill every dpserver on that host.
- Using an *`amount_of_dpservers`* prefixed with a tilde (~N) causes proteus to kill the first N dpservers (the ones that connected to the proteus client first) that are still live and useful.
- Using an *`amount_of_dpservers`* suffixed with a tilde (N~) causes proteus to kill the last N dpservers (the ones that connected to the proteus client most recently) that are still live and useful.

When killing an *`amount_of_dpservers`*, proteus sends an update to dpconsole with a list of the hosts that had dpservers killed, so you know which hosts no longer have dpservers running on them.

See Examples on page 370 for examples of the use of the `kill` command.

`retire [hostname| dpserver_number| ~amount_of_ dpservers]`	Similar to `kill`, except that proteus does not send the EXIT message immediately. Instead, it waits until the dpserver completes its current template and then tells the dpserver to checkpoint. Once the dpserver has checkpointed, proteus sends it an EXIT message. Use this command instead of `kill` if you do not want to lose any work that has been done. Note, however, that the dpserver will not be shut down until it has finished its current work. You can use one or more host names, dpserver numbers, or amount of dpservers prefixed with the ~ (tilde) character, just like `kill`, with two differences in behavior: ▪ If you pass the connection number of the dpconsole you are using, it will not be retired. It will have no effect. ▪ If you pass an amount of dpservers, for example with `retire ~6`, proteus retires the first six dpservers to finish their work (instead of the first six live dpservers on its list, as the `kill` command does). The Proteus tool alerts the dpconsole as each dpserver is retired with the dpserver's connection number and host name. For example, `Dpserver #4 retired on host_thirteen.`
`sleep [duration]`	Pauses the dpconsole application until the end of the specified *duration*, at which time subsequent commands are issued. Note that dpconsole (and the DP controller job to which it is connected) will wait for the sleep command to finish before exiting.

Examples

Using dpconsole in a Script

The dpconsole application is designed to work both interactively and in a script. If using a script to pass commands to dpconsole, send it a list of commands using the < Unix operator, as shown. (All examples use the tcsh shell.)

```
#!/bin/sh
dpconsole -c client_host < dpconsole.commands
```

Where `dpconsole.commands` is a file containing commands for dpconsole, one to a line:

```
info
spawn  host_one  host_two
retire ~2
exit
```

To list the commands in the same script where you invoke dpconsole, use the following:

```
#!/bin/sh
dpconsole -c client_host <<-EndOfList
    info
    spawn  host_one  host_two
    retire ~2
    exit
EndOfList
```

The << operator tells Unix to feed the next lines of the same script as input to dpconsole until it sees the characters designated after the << (in this case, those characters are `EndOfList`). The dash (`-`) in between the << and `EndOfList` tells Unix to ignore leading whitespace on the lines fed to dpconsole, so you can safely indent them in order to see them more easily.

Alternatively, to issue all dpconsole commands and invoke dpconsole on one line, you can pass commands in a string separated by newline characters. For example:

```
echo "kill host_one \n spawn host_two" | dpconsole -c proteus_host
```

Logging dpconsole Output

If using dpconsole in a script, you can capture the output the standard way in Unix; that is, using > to direct the output to a file.

To use dpconsole interactively and still get a log of dpconsole's output, you can pipe dpconsole's output to both a log file and the screen by using the program tee. For example:

```
dpconsole -c  client_host |& tee dpconsole.log
```

The `|&` command tells Unix to send both the standard output and the error output to the tee program, which takes a file name and puts the data out to both the file and the screen. This is useful in some cases, such as to examine a long list of dpservers provided by the `info` command.

Using the spawn Command

```
spawn sge_cluster_name:3 host_one host_one host_one
```

or

```
spawn lsf_cluster_name:3 host_one host_one host_one
```

Using the kill Command

Some examples of kill commands and their meanings are:

To kill dpserver number 3:

```
kill 3
```

To kill all dpservers on host_three:

```
kill host_three
```

To kill the first seven live dpservers:

```
kill ~7
```

You can also list multiple host names, dpserver numbers, or amounts of dpservers:

```
kill 3 host_three ~7
```

This kills dpserver number 3, all dpservers on host_three, and seven other live dpservers.

```
kill ~4  2  3 bad_host
```

This kills dpservers number 2 and 3, all the dpservers on bad_host, and four more live dpservers.

Note that if you pass multiple entries to kill that include amounts of dpservers, the amounts of dpservers are processed last. The entries that correspond to more specific dpservers, the dpserver numbers and the hostnames, are sent to proteus first in the order they appear in the kill command. In the last example, even if dpservers 2 and 3 are two of the first four live dpservers, they are killed first. Then all the dpservers on the host bad_host are killed, and after that four remaining live dpservers are killed.

dpserver

This is the local correction manager. It communicates with proteus to request templates for correction, report status of templates, and for general housekeeping. It also manages the correction process. Many dpservers can be attached to a proteus session to assist in the correction process.

The dpservers can be added and removed from the proteus session at any time. (Work in progress on a dpserver can be terminated and recorrected by another dpserver.) A dpserver is removed by terminating the process using the standard UNIX `kill` command. The dpservers can be added as described in the following, or with the remote_server script.

Every dpserver gets its own logfile, named logname.*server_number*.

`dpserver [-p port] [-c host] [-V level]`

where:

`-c host`	The host the proteus is running on. The default is the current host.
`-p port`	The port used by the proteus. The default port is 2346.
`-V level`	A number from 0 to 5 indicating the verbosity of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. The default is 3.

Refer to Chapter 6, Distributed Processing, for more information on distributed processing.

gds2vrt

This is used to convert test patterns to .vrt format for input to the Proteus WorkBench application or the ProGen tool. This is intended for small test patterns only; unpredictable behavior can result if large file conversions are attempted.

`gds2vrt [-options] input_file`

where *`input_file`* is the GDS or OASIS file to convert to .vrt format.

The available command-line options are:

`-b`	This reads the GDS BOX elements as BOUNDARY elements.
`-c cell`	This is the name of cell to extract. The default name is the topcell.
`-d dbu`	This is the database unit of input file. The default value is read from the input file.
`-h`	Help. This option displays the text help on gds2vrt.

`-j` *`jobname`*	This allows you to override the specified job name for temporary files.
`-k`	This disables the clean-up of temp files (for debugging).
`-m`	Merge. This removes polygon overlaps.
`-o` *`filename`*	The filename is the name of output file. The default filename is *inputfile*.vrt, where *inputfile* is the base name (everything preceding the .gds or .oas extension) of the input file.
`-s` *`scale`*	This is the scale factor from input to 1x. The default value is 1.
`-t` *`type`*`:`*`layer`*`:`*`datatype`*	This is the type mapping. This option puts data from input file layer on type. If input file datatype is not given, all datatypes are taken. This parameter may occur multiple times for the same or different types. For example: `gds2vrt -t 0:31 -t 0:35 -t 1:33 -t 2:34 example.gds`
`-u` *`output_dbu`*	This is the database unit of the output .vrt file. The default value is `input_dbu`.

gdsmerge

This is used to combine two GDS libraries from separate stream files. The default behavior (no options) is to copy all cells in both libraries into a new single library (resulting in two or more topcells). The default name for the new library is taken from source1. Duplicate structure names are detected and generate an error. This utility and gdssplit support 8,196 user types (layers) and 8,196 datatypes.

The syntax for this command is as follows:

```
gdsmerge -s1 source1 -s2 source2 -d destination
   [-tops1 cellname] [-tops2 cellname] [-topd cellname]
   [-libd libname] [-nl_127|-nl_31|-nl_32760]
   [-prefix1 prefix1] [-prefix2 prefix2]
```

where the required parameters are:

`-s1` *`source1`*	Source GDSII stream input file
`-s2` *`source2`*	Source GDSII stream input file

`-d` *`destination`*	Destination GDSII stream output file.

The available command-line options are:

`-h`	Help. This option displays the text help on gdsmerge.
`-libd` *`libname`*	This sets the destination library name. The default is the source library name.

`-nl_127|`
`-nl_31|`
`-nl_32760`
	Specifies the maximum length for GDSII structure names. The default is -nl_127 (127 characters).
	-nl_31 sets the maximum length to 31 characters,
	-nl_32760 to 32,760 characters.

`-offset1` *`x y`* `-offset2` *`x y`*	The `-offset1` term operates on the input file specified by the `-s1` parameter, and the `-offset2` term operates on the input files specified by the `-s2` parameter. The effect of these parameters is to translate data in the respective files by *x* nanometers along the x-axis and *y* nanometers along the y-axis, where x and *y* are floating point numbers. Both x and y are required to be present on the command line. Both options must be used in conjunction with the `-topd` option, which specifies an overall topcell for the merge output file.
`-prefix1` *`prefix`*	Here, *`prefix`* prepends all structures copied from source 1.
`-prefix2` *`prefix`*	Here, *`prefix`* prepends all structures copied from source 2.
`-tops1` *`cellname`*	This uses *`cellname`* as the topcell in source 1. This option may be specified multiple times.
`-tops2` *`cellname`*	This uses *`cellname`* as the topcell in source 2. This option may be specified multiple times.
`-topd` *`cellname`*	This is optional. If used, *`cellname`* is used as a holding topcell in the destination file.
`-v`	This prints version information to the parent xterm.

Error Conditions

If the database units in the two input files are different, gdsmerge terminates with an error message.

If two or more cells have the same name after prefixes have been applied, gdsmerge terminates with an error message and a list of all collisions. Any reference to undefined cells causes a termination with an error message. At least one unreferenced topcell must exist in each input file.

---

gdssplit

This is used to extract selected portions of a .gds file based on user type (layers) and datatype, and/or cell name. The result is placed in a user-specified destination file. The destination file contains the minimum number of hierarchical cells (GDS structures) necessary to represent the graphical data occurring on the layers and data structures specified on the command line. This utility and gdsmerge support 8,196 user types (layers) and 8,196 datatypes.

Cells that do not contain `PATH`s or `BOUNDARY` types on the specified layers and datatypes, and that have no descendants containing these types, do not appear in the destination file. This means that empty cells (cells that contain no `PATH`s, `BOUNDARY` types, or references to other cells) do not appear in the destination file.

Optionally, gdssplit can be used to create a complement file containing graphics not copied to the destination file. The graphics included in the complement file must, however, lie beneath the collection of topcells specified in the call to gdssplit. Topcells are specified with one or more `-tops` options. If no `-tops` options are specified, the list of topcells defaults to the collection of topcells that occur within the GDSII source file. (The GDSII standard allows a file to contain any number of topcells.) When complete, the complement file contains enough of the original hierarchy beneath the topcell list to contain all `BOUNDARY`, `PATH`, and `BOX` elements not written to the destination, and any `TEXT` or `NODE` elements occurring beneath the topcell list.

Note that square brackets [ ] enclose optional parameters, and curly braces { } enclose parameters that can be specified multiple times:

```
gdssplit  -s input_file -d output_file [{-tops cellname}]
   [-topd cellname] [-libd libname] [-prefix prefix]
   [-copy all] [-nl_127|-nl_31|-nl_32760]
   [{-copy Ls:[Ds] [to Ld:[Dd]]}] [-dbu [m] dbunit]
```

where the required parameters are:

`-s` *`input_file`*	The source file name
`-d` *`output_file`*	The destination file name

The available command-line options are:

`-copy all`	This copies all cells that contain PATHs and BOUNDARY types on any layer or datatype, or that have descendants that do. Empty cells that contain no PATHs or BOUNDARY types, or reference cells that constrain PATHs or BOUNDARY types, are not copied.
`-copy` *`Ls`*`:[`*`Ds`*`]` `[to` *`Ld`*`:[`*`Dd`*`]]`	This copies element data on layer Ls and datatype Ds to layer Ld, datatype Dd. If Ds is absent, this copies all datatypes. If Dd is absent, it uses the source datatype. This option may be specified multiple times.
`-dbu [m]` *`dbunit`*	This sets the database unit in the destination file. If [m] is present, dbu is in microns. If it is absent, dbu is in meters.
`-dcomp` *`filename`*	This is the filename for complement data.

`-h`	This displays the text help on gdssplit. The following information is displayed:
	`gerbil ~...proteusprog % gdssplit -h`
	`gdssplit Release W-2004.09`
	`Revision Proteus_2004.09-6_25Jul05-764531`
	`(64f/64m SUN7).`
	`host: gerbil`
	`Copyright 1995 - 2005`
	`Synopsys, Incorporated`

`-libd` *`libname`*	This sets the destination library name. The default is the source library name.

`-nl_127|-nl_31|-nl_32760`	Specifies the maximum length for GDSII structure names. The default is -nl_127 (127 characters). -nl_31 sets the maximum length to 31 characters, -nl_32760 to 32,760 characters.
`-opc_scale` *`scale_cor scale_out corgrid snap_45`*	This option treats scales and snaps as the Proteus tool does. The first three arguments are doubles, *`corgrid`* is given in microns, and *`snap_45`* is either `SINGLE` or `DOUBLE`. All arguments are required. If any non-45-degree lines are found, crimping is performed. The crimping grid is defined such that for each polygon, the number of vertices is less than the maximum allowed in the `OUTPUT_VERT_MAX` parameter.
`-prefix` *`prefix_value`*	Prepends all copied structures with *`prefix_value`*.
`-tops` *`cellname`*	This option uses *`cellname`* as the topcell, and may be specified multiple times.
`-topd` *`cellname`*	This option uses *`cellname`* as a holding topcell in the destination file.
`-treatBoxAsBoundary`	This option treats GDS `BOX` types as `BOUNDARY` types and `BOXTYPE`s as datatypes.
`-v`	This prints the version information to the command line.

Error Conditions

If two or more cells have the same name after prefixing and truncation, gdssplit terminates with an error message and a list of all collisions. Any reference to undefined cells causes a termination with an error message. At least one unreferenced topcell must exist in the input file.

hierman

`hierman [-`*`options`*`] `*`job_control_file`*

where:

- *job_control_file* is the output of xmscript.

This is used to execute standalone hierarchy management. Normally, proteus invokes Hierarchy Management automatically as needed. To separate the input hierarchy analysis and setup steps from a proteus run, `hierman -fe` can be used to execute only front-end hierarchy processing.

Note: Performing hierarchy management on a `PROTEUS_JOB_FLOW` recipe requires the use of the proteus binary application. See proteus on page 387 for details. On a `PROTEUS_JOB_FLOW` recipe, hierman can only be run with the -fe option.

The Hierarchy Manager finds an optimum number of unique correction templates to maximize correction processing efficiency and minimizes the size of the output file.

When invoked with a `PROTEUS_JOB_FLOW` recipe, hierman performs all processing steps of hierarchy management. The hierman application can also use the following option with a `PROTEUS_JOB_FLOW` recipe:

`-fe`

performs only the front-end (FE) processing steps of hierarchy management (scan, scaffolding, and instance generation). The back-end steps (context analysis and template generation) are not executed.

By default, the proteus binary reruns the hierman front end (FE), even if hierman files already exist.

Use the dpconfig option `NO_RERUN_HIERMAN` in order not to rerun hierman FE if it was run before (this is not recommended for production). However, if the .STAT file does not exist or the previous run did not successfully complete the FE, proteus runs hierman FE even if this keyword is used.

To prevent accidentally leaving `NO_RERUN_HIERMAN` in the dpconfig file for production, a permanent warning is issued when it is present (and applicable) and the .STAT file indicates a previous successful hierman run.

Note: At this time, hierman does not support the `-tb` *n* option (as do corexec, rdebug, and celltool). Instead, proteus is used to accomplish context analysis and template generation for each template block.

Refer to Chapter 4, Hierarchy Management, for more information on the Hierarchy Manager. For details on distributed processing, see Chapter 6,

Distributed Processing.

initgds

This is used to write the header information into the .gds file.

Note: This application is not supported with `PROTEUS_JOB_FLOW` recipes.

`initgds [-options] job_control_file [gdsfile [libraryname]]`

where:

- *`job_control_file`* is the output of xmscript.
- *`gdsfile`* is the name of the .gds file to initialize. The default `OUTPUT` filename parameter is taken from the *`job_control_file`*.
- *`libraryname`* is the name of library given in the .gds file. The default `CORLIB` parameter is taken from the *`job_control_file`*. If *`libraryname`* is specified, the .gds file must be specified.

The available command-line options are:

`-h` Help. This option displays the text help on initgds.

`-s` Suppresses printing of syntax warnings.

oasmap

This is used to map specified layer and datatype information from a source OASIS file to a destination OASIS file. It is also possible to do this with oassplit, but oasmap is faster because it only changes the layer and datatype. While copying, oasmap cannot skip any records or change anything other than the values for layer and datatype.

Note that square brackets [ ] enclose optional parameters, and curly braces { } enclose parameters that can be specified multiple times:

```
oasmap -s source_file -d destination_file
[{-l [all] layer[:datatype|all] to layer[:datatype]}]
[-r] [-report] [-h]
```

where the required parameters are:

`-s` *`source_file`* The input source file name

`-d` *`destination_file`* The output destination file name

The available command-line options are:

`-h` Help. This option displays the text help on oasmap.

`-l [all]` *`layer`*`[:`*`datatype`*`|all]to` *`layer`*`[:`*`datatype`*`]`

This defines the layer and datatype mapping. It puts data from *`layer`* in *`source_file`* onto *`layer`* in *`destination_file`*. This parameter can occur multiple times for the same or different layer and datatypes. If the optional `[all]` is present, all layers (`-l all`) or datatypes (`-l layer1:all`) are mapped. For example:
`oasmap -s file1 -d file2 -l all to 5:1`
... maps all layer/datatypes to 5:1.
`oasmap -s file1 -d file2 -l 2:all to 5:1`
... maps all datatypes on layer 2 to 5:1.

`-r` Reports diagnostic information during debugging.

`-report` Produces a list of all layer and datatypes that occur in the file. This report is written to stdout. The use of this option overrides all other options by ignoring them.

oasmerge

This is used to combine two OASIS hierarchies from separate OASIS files. The two files must have the same DBU value. The default behavior (no options) is to copy all cells in both files into a new single file (resulting in two or more topcells). The default name for the new file is taken from source1. Duplicate structure names are detected and generate an error.

The oasmerge utility merges just one file if one of the files has no graphics, or has none under the topcell specified, or the topcell specified does not exist.

Note that oasmerge proceeds without error if given the incorrect topcell, or topcell is missing.

CELLNAMEs, PROPNAMEs, PROPSTRINGs, LAYERNAMEs, XNAMEs, and TEXTSTRINGs are prefixed in the output file for OASIS base data processing according to the corresponding source files for eliminating possible collisions.

Note that square brackets [ ] enclose optional parameters, and curly braces { } enclose parameters that can be specified multiple times.

```
oasmerge -s1 source1 -s2 source2 -d destination
[{-tops1 cellname}] [{-tops2 cellname}] [-topd cellname]
[-scale1 scale] [-scale2 scale]
[-prefix1 prefix_value] [-prefix2 prefix2_value]
[-offset1 x y] [-offset2 x y]
[-z level] [-cmp|-no_cmp] [-r]
```

where the required parameters are:

`-s1` *`source1`*	Source OASIS stream input file
`-s2` *`source2`*	Source OASIS stream input file
`-d` *`destination`*	Destination OASIS stream output file.

The available command-line options are:

`-cmp|-no_cmp`	Sets OASIS compaction on or off for the output data. Compaction is off (`-no_cmp`) by default.
`-h`	Help. Displays the text help on oasmerge.
`-offset1` *`x y`* `-offset2` *`x y`*	The `-offset1` term operates on the input file specified by the `-s1` parameter, and the `-offset2` term operates on the input files specified by the `-s2` parameter. The effect of these parameters is to translate data in the respective files by *x* nanometers along the x-axis and *y* nanometers along the y-axis, where x and *y* are floating point numbers. Both x and y are required to be present on the command line. Both options must be used in conjunction with the `-topd` option, which specifies an overall topcell for the merge output file.
`-prefix1` *`prefix_value`*	Prepends all name-based records copied from *`source1`* with *`prefix_value`*. Must be used in conjunction with the `-topd` option.

`-prefix2` *`prefix2_value`*	Prepends all name-based records copied from *`source2`* with *`prefix2_value`*. Must be used in conjunction with the `-topd` option.
`-r`	Reports diagnostic information during debugging.
`-scale1` *`scale`*	Sets the scale factor for the cells from *`source1`*. Must be used in conjunction with the `-topd` option.
`-scale2` *`scale`*	Sets the scale factor for the cells from *`source2`*. Must be used in conjunction with the `-topd` option.
`-tops1` *`cellname`*	Sets *`cellname`* as the topcell from the *`source1`*. This option may be specified multiple times.
`-tops2` *`cellname`*	Sets *`cellname`* as the topcell from the *`source2`*. This option may be specified multiple times.
`-topd` *`cellname`*	Uses *`cellname`* as a holding topcell in the destination file.
`-z` *`level`*	Sets compression level for output data. Can be any integer value from 0 (no compression) to 9 (maximum compression). Default value is 6.

oassplit

This is used to extract selected portions of an OASIS file based on user type (layers) and datatype, and/or cell name. The result is placed in a user-specified destination file. The destination file contains the minimum number of hierarchical cells (OASIS structures) necessary to represent the graphical data occurring on the layers and data structures specified on the command line.

Optionally, oassplit can be used to create a complement file containing graphics not copied to the destination file. The graphics included in the complement file must, however, lie beneath the collection of topcells specified in the call to gdssplit. Topcells are specified with one or more `-tops` options. If no `-tops` options are specified, the list of topcells defaults to the collection of topcells that occur within the OASIS source file. (The OASIS standard allows a file to contain any number of topcells.) When complete, the complement file contains enough of the original hierarchy beneath the topcell list to contain all POLYGONs and PATHs not written to the destination, and any TEXTSTRINGs occurring beneath the topcell list.

The oassplit utility writes an empty file when the split request causes one or both of the output files to have no graphics.

Note that square brackets [ ] enclose optional parameters, and curly braces { } enclose parameters that can be specified multiple times.

```
oassplit  -s input_file -d output_file [{-tops cellname}]
   [-topd cellname] [-prefix prefix] [-copy all]
   [{-copy Ls[:Ds] [to Ld[:Dd]]}] [-dbu [m] dbunit]
   [-z level][-cmp|-no_cmp][-dcomp filename][-r]
```

where the required parameters are:

`-s` *`input_file`*	The source file name
`-d` *`output_file`*	The destination file name

The available command-line options are:

`-cmp|-no_cmp`	Sets OASIS compaction on or off for the output data. Compaction is off (`-no_cmp`) by default.
`-copy all`	Copies all cells that contain graphics records on any layer or datatype, or that have descendants that do.
`-copy` *`Ls`*`:[`*`Ds`*`]` `[to` *`Ld`*`:[`*`Dd`*`]]`	Copies data on layer *`Ls`* and datatype *`Ds`* to layer *`Ld`*, datatype *`Dd`*. If *`Ds`* is absent, this copies all datatypes. If *`Dd`* is absent, it uses the source datatype. This option can be specified multiple times.
`-dbu [m]` *`dbunit`*	Sets the database unit in the destination file. If `[m]` is present, dbu is in microns. If `[m]` is absent, dbu is in meters.
`-dcomp` *`filename`*	The filename for complement data. Note that oassplit handles complement data differently than gdssplit. When a cell is extracted with the `-tops` option and the `-dcomp` option is also specified, oassplit does not create the complement file. Instead the following message is printed: `There is no data to write to the complement file dcomp_out.oas`
`-h`	Displays the text help on oassplit.

`-prefix` *`prefix_value`*	Prepends all copied structures with *`prefix_value`*.
`-r`	Reports some diagnostic information during debugging.
`-tops` *`cellname`*	Uses *`cellname`* as the topcell, and may be specified multiple times.
`-topd` *`cellname`*	Uses *`cellname`* as a holding topcell in the destination file.
`-z` *`level`*	Sets compression level for output data. Can be any integer value from 0 (no compression) to 9 (maximum compression). Default value is 6.

printlog

This utility reads and filters binary log file data and produces a formatted ASCII text file.

The logfile produced by correction processing contains a list of records. Each record is a single integer code and a (double) floating-point value. The printlog utility scans the logfile and maintains a list of the last values encountered for codes 0 to 255. When any codes specified in the log template file are encountered, a block of formatted text is written to the output text file. The text of each block is copied from definitions in the template file, except where specified fields are replaced with numeric values.

The maximum number of printlog format templates is 2000, while the maximum number of lines in the format file is 4000.

Generally, this statement is preferred for direct output to a formatted text file.

`printlog` *`templatefile logfile`* `[`*`outfile`*`]`

Where the syntax of the log template is as follows:

```
[
%%h
<header text line 1>
<header text line 2>
%%
]
{
%%<print code>
<text> <format> <text> <format> ....
<text> <format> <text> <format> ....
%%
}
```

The *format* term takes the form %####.##<*code*> and appears anywhere in the text block. The number of pound (#) characters specifies fixed integer and fractional decimal places for printing the current value of code. The asterisk (*) can be used instead of the # digit markers to specify a variable-length field. A block is printed whenever `print_code` is encountered. The basic structure is:

```
printlog *.ltp_file *.log_file text_file
```

where:

- **.ltp_file* is the log template.
- **.log_file* is the log data file.
- *text_file* is the output text file.

The default output text file is PRINTLOG.TXT.

Sample Template File

```
%%h
This output produced from logfile created by recipe
"chip.rcp" on TESTCHIP pattern.
%%
%%4
   ----- Segment Code %##4 ----- Direction %#3 ------
   Target Location:  X = %*1nm.  Y = %*2nm.
          Protowt (nm):      Evaluation:
%%
%%8
          %####.#7           %#.######8
%%
```

Sample Output

The following example uses the preceding template file for its output.

```
  This output produced from logfile created by recipe chip.rcp
  on TESTCHIP pattern.
      ---- Segment Code    3 ---- Direction    6 ----
      Target Location:  X = 0 nm.  Y = 3500 nm.
             Protowt (nm):      Evaluation:
               150.0              0.381430
               140.0              0.372100
               130.0              0.362645
               120.0              0.353311
               110.0              0.344100
               100.0              0.335010
                90.0              0.326041
                80.0              0.317194
                70.0              0.308247
                60.0              0.299647
                50.0              0.290952
                40.0              0.282598
                30.0              0.274157
                20.0              0.265843
                10.0              0.257861
                 0.0              0.249800
               -10.0              0.241671
               -20.0              0.233869
               -30.0              0.226386
------ Segment Code    3 ----- Direction    6 ----
      Target Location:  X = 0 nm.  Y = 3500 nm.
             Protowt (nm):      Evaluation:
               150.0              0.381430
               140.0              0.372100
               130.0              0.362645
               120.0              0.353311
```

proteus

The proteus binary is a single executable supporting hierarchical processing and distributed correction. It accepts `PROTEUS_JOB_FLOW` recipes.

Using the proteus binary is the preferred technique for processing new recipes.

```
proteus [-options] job_control_file
   [start_template|list_file]
```

where:

- *job_control_file* is the output of xmscript.

The available command-line options are:

`-h`	Help. This option displays the text help on proteus.
`-s` *host*	Server host. Runs the remote_server script with *host* as an argument. This can occur multiple times with the same or different *host* names e.g.: `proteus -s < host1 > -s < host2 > -s < host2 > go.pjx` The number of allowed servers is limited by the number of file descriptors, which can be raised by the system administrator.
`-p` *port*	Default port is 2346. Ports are chosen around this starting point. if running multiple clients on the same host, a different *port* must be specified for each.
`-V` *level*	A number from 0 to 5 indicating the verbosity level of messages. 0 indicates nearly silent messages, 5 indicates the most verbose. Default is 3 unless `-f` is present, in which case the default is 4.
CONFIG_ DIRECTIVE	Overrides any line in the dproteus.cfg file by giving a new definition on the command line. For example, `-START_PORT 4000`
`-f`	Forces proteus into recovery mode to finish a previous failed correction. NOTE: `proteus -f` recovery for `PROTEUS_JOB_FLOW` with `PIPELINE_STRATEGY FRONT_LOAD` or `PIPELINE_STRATEGY BACK_LOAD` might recorrect some templates that were already processed. During `proteus -f` recovery, some dpservers might intermittently produce errors due to file synchronization, but they will not produce bad results.
`-r`	Corrects templates in descending (reverse) order by template number.

`-ht` *`host`*	Hyper-threading server host. Runs the `remote_server` script with *host* as an argument. For example, to run *recipe.pjx* using a normal dpserver and a hyper-threading server named *flamingo*: `proteus -s flamingo -ht flamingo recipe.pjx`
`-m`	Runs mktop as a thread on the machine where proteus runs.
`-restart` *`TC_n`*	Forces proteus to restart at the specified `TEMPLATE_CALL`, discarding previous results from `TEMPLATE_CALL`s following *`TC_n`*. See proteus on page 296 for further details.

Running proteus automatically runs hierarchical processing steps before beginning distributed processing unless the `NO_RERUN_HIERMAN` configuration file option (`OFF` by default) is present (see hierman on page 378).

Note: If you are sourcing a dproteus.cfg file, be aware of whether or not `NO_LOCAL_SERVER` is not present there. If you run proteus on your own machine, proteus tries to launch another process and `NO_LOCAL_SERVER` prevents this. In such a case, you might want to delete it or comment out `NO_LOCAL_SERVER`.

Typically, intermediate files are deleted at the end of a `PROTEUS_JOB_FLOW` run. Place the keyword `RETAIN_JOB_FLOW_FILES` (`OFF` by default) in your job control file to retain intermediate TINF, and graphics output files, and to create intermediate recipe files.

The `DBU_PROC_OUT` keyword allows you to specify the intermediate graphics file resolution in `PROTEUS_JOB_FLOW` recipes. By default, this value is equal to the value of `DBU_PROC` of the current template block. The `DBU_PROC_OUT` value can be overridden in each template call. The specified value should have an integral ratio with the value of `DBU_PROC` of the current `TEMPLATE_BLOCK`.

The proteus binary is also capable of recovery when run with the `-f` option, and restart from a specific `TEMPLATE_CALL` with the `-restart` option.

puf2vert

This is used to convert .puf format graphics files to .vrt format graphics files.

`puf2vert [-options] [job_control_file]`

where:

- *`job_control_file`* is the job control file to convert.

The available command-line options are:

`-c` This option suppresses crimping on non-45s.

`-g` This option suppresses `CORGRID` snaps.

`-G` This option specifies `CORGRID` for grid snap. The default is taken from the *`job_control_file`* if not bypassed. A default of 160 is used if this file is bypassed.

`-h` Help. This option displays the text help on puf2vert.

`-j` This option bypasses the job control read, making the *`job_control_file`* optional.

`-m` This option defines `MIN_N45_L` for non-45 crimping. The default is taken from the *`job_control_file`* if not bypassed. A default of 1.0 is used if this file is bypassed.

`-o` This option defines the output filename. The default is TSOURCE.TMP.

`-v` This option suppresses void (hole) reversal.

remote_server

This script starts a dpserver on the specified node.

`remote_server client_host port verbosity call_number call_type`

where:

- *`client_host`* is the host on which a new dpserver is to be run.
- *`port`* is the port used by the DP controller.
- *`verbosity`* is the number from 0 to 5 indicating the verbosity of messages. 0 indicates nearly silent messages, 5 indicates the most verbose.

- *call_number* is the index of this server in the sequence of servers being started (typically by proteus). This parameter can be used within the remote_server shell script to introduce a start-up delay between servers. See the remote_server shell script for an example.
- *call_type* specifies whether remote_server was called by distributed hierman or proteus. The *call_type* is either 0 (hierman) or 1 (proteus).

All arguments are required for this syntax.

The remote_server shell script is a modifiable script used to initiate dpservers from the client. The client executes a remote_server for each `-s` *host* on the command line, and for the current host (unless overridden in the configuration file).

This script can be customized by knowledgeable users to suit the needs of a particular environment. Options for customizing the script might include: changing message logging; setting up environment variables; setting the environment (use the korn shell or bourne shell instead of c shell); modifying the execution parameters such as the `-V` option; and modifying for execution in a batch environment, such as submitting a job to a batch queue. Consult your system administrator for assistance for this customization.

Consult Chapter 6, Distributed Processing, for more information on remote servers.

vert2puf

This is used to convert .vrt format graphics files to .puf format graphics files.

`vert2puf` *puffile vertfile_1* [*vertfilex* ...]

where:

- *puffile* is the name of the .puf file to convert.
- *vertfile_1* is the destination file.
- *vertfilex* is the list of subsequent .vrt files to which to convert the .puf file.

vrt2gds

This is used to convert .vrt test patterns to GDSII format. This is intended for small test patterns (less than 64,000 polygons by default) only. Unpredictable behavior can result if large file conversions are attempted.

`vrt2gds [-options] vrtfile`

where *`vrtfile`* is the name of the .vrt file to convert to GDS.

The available command-line options are:

`-c cell`	Defines the cell to extract. Here, cell is the name of the output cell (default is `vert_cell`).
`-C`	This overrides the execution of closegds. Use this if multiple cells are being created on the same .gds file. This should be specified on all but the last in the series.
`-d output_dbu`	Defines the database unit of the output. Here, *`output_dbu`* is the database unit of output gds file. The default value is the dbu of the .vrt file, or 1 nm if none exists. Data is snapped to output_dbu if different from the dbu of the .vrt file.
`-h`	This accesses the text help on this command.
`-I`	This overrides the execution of initgds. Use this if multiple cells are being created on the same .gds file. This should be specified on all but the first in the series.
`-L lib`	Defines the output library. Here, *`lib`* is the name of the output library (default is `PROTEUS_LIB`).
`-n`	This prevents inner holes in patterns from being filled.
`-O`	This overwrites the output file if it exists without querying you.
`-o filename`	This defines the name of output file (default is vrtfile.gds)
`-s type`	This initiates type-skipping. Data of type *`type`* is prevented from writing to the .gds file. This parameter can appear more than once for different types. For example:`vrt2gds -s 2 -s 5 example.vrt`
`-sall`	This skips data for all datatypes not mapped by the `-t` option from writing to the .gds. This option should be used together with `-t` option only.

`-t type:layer:datatype`

This defines the type mapping. It puts data from type *type* onto GDS layer *layer*. If *datatype* is not given, it defaults to 0. This parameter can occur multiple times for the same or different types. For example:

```
vrt2gds -t 0:31 -t 0:35 -t 1:33 -t 2:34
example.vrt
```

All types default to a layer of the same number.

xmscript

This utility builds output job files under the control of the master job script. See Chapter 2, xmscript Directives, for more information on xmscript.

```
xmscript [-options] [master_job_script [job_control_file]]
-or-
xmscript [-options]
```

where:

- *master_job_script* is the input of xmscript.
- *job_control_file* is the output of xmscript.

The available command-line options are:

`-ami` This option causes xmscript to allow missing input files. When this option is used, xmscript does not error out even if no input file exists.

`-c` This option suppresses comment generation in the output file. Without this option, all preprocessor commands (`#DEFINE`, and so forth) are converted to comments in the output file. See #GENERATE_COMMENTS on page 22.

`-C` This option instructs the software not to perform variable substitutions within the comments. See #COMMENT_SUBSTITUTE on page 10.

`-h` This option displays the text help on xmscript.

`-r` This option starts xmscript with `#REMOVE_FALSE_BLOCKS` set to `ON`. See #REMOVE_FALSE_BLOCKS on page 28.

`-s` This option suppresses the correction syntax checking of the output file.[1] See #SYNTAX_CHECK_OPC on page 30.

`-v` This option prints the version information to the command line.

`-e` This option is used to encrypt a specified file from the command line. In this mode, when xmscript encounters a `#ENCRYPT` keyword, all text is encrypted up to a matching `#END_ENCRYPT`. No other operations, such as substitution replacement, `#IF`, `#DEFINE`, or `#INCLUDE`, will be performed. The GUI mode is not supported when using the `-e` command-line option; specify the `-x` option to avoid opening the GUI window.
Syntax checking should be turned off (using `-s`) when in encrypt-only mode. Without the `–s` option, multiple corBASIC syntax errors are often reported.

`-x` This option runs the preprocessor without opening a window. This option executes the job, and requires that both *`master_job_script`* and *`job_control_file`* be specified on the command line. This option takes all defaults from the script.

1. If the `-s` option is not specified and a syntax error is encountered, xmscript does not create the output file. This prevents hierman from running on an incorrect or empty output file. If there are errors, xmscript issues an error message and the output file is renamed ERROR_OutputFileName, where OutputFileName is the original output file name you specified, with or without an extension.

Perl Script Applications

The following applications are available as perl scripts:

`pmdl2cmdl.pl`	This script converts files from .pmdl format to the common model file (.cmdl) format.
`ppuf2gds.pl`	This script converts files from Precim .puf format to GDSII.

pmdl2cmdl.pl

This script is used to convert .pmdl files to common model files (.cmdl).

`cmdl2pmdl.pl` *`pmdl_file`*

where *`pmdl_file`* is the .pmdl model file to convert to the new .cmdl file.

ppuf2gds.pl

This script is used to convert from .puf files to GDSII format.

`ppuf2gds.pl [-options] puffile`

The available command-line options are:

`-c "cellname"`	This defines the name of the output cell. The default is `puf_cell`.
`-C`	This option overrides the execution of closegds. This is to be used if multiple cells are being created in the same .gds file. The `-C` option should be specified on all but the last in the series.
`-f "filename"`	This defines the name of output .gds file. The default name is *puffilename*.gds.
`-h`	This displays the text help information on `ppuf2gds.pl`.
`-i`	This overrides the execution of initgds. This is to be used if multiple cells are being created in the same .gds file. The `-i` option should be specified on all but the first in the series.
`-k`	This option overrides the final clean-up of temporary files.
`-L "filename"`	This option defines the output library name. The default library name is `PROTEUS_LIB`.
`-m "type:layer"`	This option maps Proteus tool types to GDS layers. Use the form 0:1, where the first number is the type and the second number is the GDS layer. The `type:layer` argument is a comma-separated list. Only those types specified are output. The argument defaults to: 0:0,1:1,2:2,3:3,4:4,5:5,6:6,7:7,8:8,9:9. Spaces are permitted, but, if used, require the list to be enclosed in quotation marks.
`-o`	This option overwrites ./TARGET.TMP without prompting you. Note that a .vrt file is created named ./TARGET.TMP. If this file exists, it is overwritten when `-o` is used.

A

NEW_* Keywords

Explains NEW_* keyword usage in the Proteus tool.

NEW_* Keywords

Keywords with the prefix `NEW_` introduce new functionality that is not yet intended to be the default behavior in the Proteus tool. `NEW_*` keywords must be added specifically to your recipe to enable their function.

A `NEW_*` keyword is typically made the default and deprecated in the next major release after it is introduced. In the release following its deprecation, the `NEW_*` keyword becomes obsolete and is no longer documented.

NEW_CORRECT_SEGMENTS_WITH_OPPOSING_DIR

Python-based stitching operations can result in an infinite loop when processing a degenerate polygon (a polygon with only two vertices, or having zero area). You can resolve this issue by setting the keyword `NEW_CORRECT_SEGMENTS_WITH_OPPOSING_DIR`. This keyword is `OFF` by default.

NEW_PRINT_TEMPLATE_CALL_NAME

The new recipe keyword `NEW_PRINT_TEMPLATE_CALL_NAME` prints the template call name in the proteus and server log files. If the current template call is encrypted, its name will appear as `ENCRYPTED`.

`NEW_PRINT_TEMPLATE_CALL_NAME` is `OFF` by default.

Examples (Non-Encrypted)

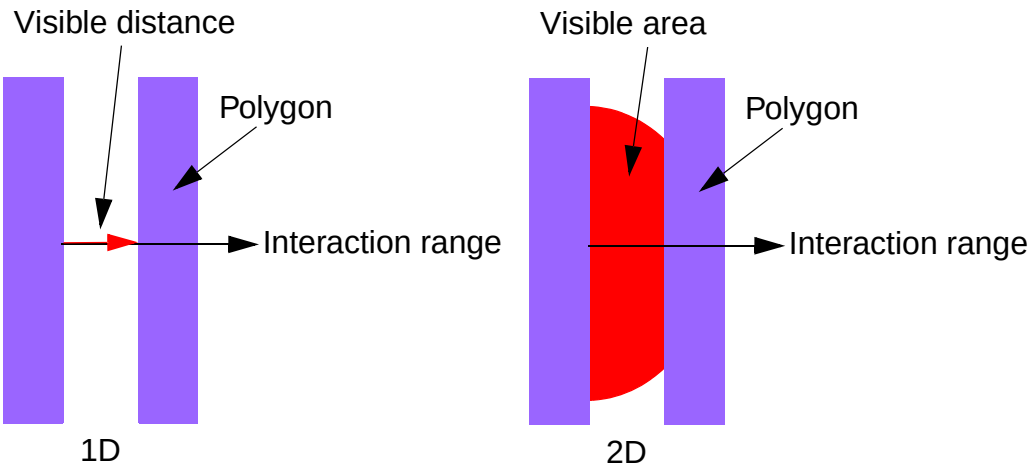
```
# mktop done, ~   0.75% Completed ~   0.00% area (of TC  1 /
OPC_BLOCK)
#   133 done, ~   1.49% Completed ~   0.76% area (of TC  1 /
OPC_BLOCK)
#   132 done, ~   2.24% Completed ~   1.53% area (of TC  1 /
OPC_BLOCK)
```

Examples (Encrypted)

```
# 5851 done, ~  9.82% Completed ~ 95.87% area (of TC 1 / ENCRYPTED)
# 5850 done, ~  9.84% Completed ~ 95.87% area (of TC 1 / ENCRYPTED)
# 5849 done, ~  9.85% Completed ~ 95.87% area (of TC 1 / ENCRYPTED)
```

NEW_VISIBLE_1D_SIGNAL

The `NEW_VISIBLE_1D_SIGNAL` keyword allows you to calculate the visible distance for each evaluation point within the interaction range, or calculate the visible area for each evaluation point within that range. Calculating the visible distance can improve the calculation speed, but results might worsen in some instances, such as encountering unwanted noise, as compared to calculating the visible area.



Setting `NEW_VISIBLE_1D_SIGNAL ON` instructs the tool to calculate the visible signal between two polygons using a method that calculates the

distance (1D). Setting this keyword OFF (the default) instructs the tool to calculate based on the visible area (2D).

NEW_VISIBLE_1D_SIGNAL

B

Deprecated Functionality

Lists deprecated keywords, directives, and environment variables.

Deprecated Keywords

The following keywords, directives, and environment variables are deprecated in this release of Proteus and will become obsolete in a future release. Unless otherwise noted in the keyword description, the functionality enabled by each keyword is locked to the current default behavior.

- #SUPPRESS_COMMENT_GEN
- BOOLEAN_SCALE
- CLUSTER_THRESHOLD
- NEW_DISCRETE_CLIP_LSEGS
- NEW_DYNAMIC_CORBASIC_ARRAYS
- NEW_DYNAMIC_SEGMENT_ARRAYS
- NEW_EDGE_TABLE_BY_LAYER
- NEW_MLO_DIMENSIONAL_INTERSECT_DEFAULT
- NEW_MLO_NONE_EXT
- NEW_MSS_UNIFORM_CORRECTION
- NEW_MULTIPLE_OUTPUT_FILES
- NEW_PYTHON_TRUE_DIVISION
- NEW_SHIFT_LARGE_COORDINATE
- NEW_SUPPRESS_WARNING_COUNT
- NO_RERUN_HIERMAN_FE

- PIPELINE_STRIPES
- PROTEUS_PATH
- USE_REVERSE_TEMPLATE_LIST

#SUPPRESS_COMMENT_GEN

Note: This directive is deprecated and will become obsolete in a future release.

Description

`#SUPPRESS_COMMENT_GEN` is equivalent to `#GENERATE_COMMENTS OFF`. You are encouraged to use `#GENERATE_COMMENTS OFF` rather than `#SUPPRESS_COMMENT_GEN`.

Syntax

```
#SUPPRESS_COMMENT_GEN
```

Options

None.

See also

#GENERATE_COMMENTS on page 22

BOOLEAN_SCALE

Note: This keyword is deprecated and will become obsolete in a future release. Setting this keyword to any value other than 2 results in an error.

Description

This provides an alternate interpretation of Boolean snapping. This keyword is helpful in preventing off-grid intersections during MLO and Boolean operations, which can result in non-45 degree edges in some cases.

The default setting of `BOOLEAN_SCALE 2` turns on the edge-snapping behavior where the Proteus tool works to prevent off-grid interactions by internally scaling graphics up by 2x prior to internal Boolean operations and back down afterward.

Using `BOOLEAN_SCALE` of 2 causes differences when compared with 1x scale due to the change in snapping behavior, even on Manhattan edges. When using `BOOLEAN_SCALE 2` with assist feature (AF) generation, the small changes in Boolean output might cause slightly different AF placements or, in some cases, might change space constraints, causing some AFs to appear or disappear.

Note: Proteus WorkBench Boolean operation does not support `BOOLEAN_SCALE`.

Syntax

`BOOLEAN_SCALE` *`scale_value`*

Options

`scale_value`

The magnitude by which graphics are scaled. Accepted value is:

- 2

  The default. Scales the graphics up 2x prior to Boolean operations, and back down afterward. This is the only accepted value for `BOOLEAN_SCALE`; any other value results in an error.

Examples

Figure 73 illustrates a situation where a non-45 degree edge can be created with 1x Boolean behavior. The dots in the diagram indicate potential grid points. The lines compose different polygons.

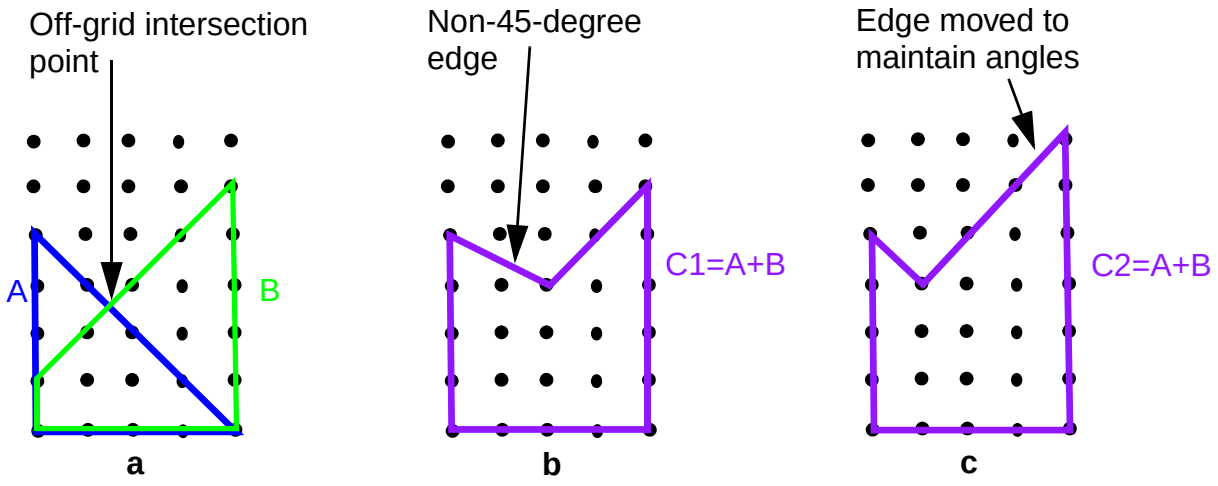


Figure 73 Non-45 degree edge created with 1x Boolean behavior

In Figure 73a, consider a Boolean OR between the polygons A and B (C = A + B). The off-grid intersection indicated cannot be represented and must be snapped to a legal grid location (shown in Figure 73b), causing edges not to be on an even multiple of 45 degrees (non-45 degree edges). The `BOOLEAN_SCALE 2` interpretation moves the snapping point and the edges slightly to avoid creating non-45 degree edges, resulting in a slightly different geometry as shown in Figure 73c. This interpretation can help preserve the original intent, while keeping edges on even multiples of 45 degrees.

See also

ALL_ANGLE_OUTPUT on page 221

CLUSTER_THRESHOLD

Note: This keyword is deprecated and will become obsolete in a future release. You are advised to use `SREF_SCAFFOLD` instead.

Description

This triggers SREF scaffolding when the instance count exceeds the specified positive integer (the count is interpreted as a threshold). When this statement is not present, SREF scaffolding is triggered at 20,000,000. If a count of 0 is specified, SREF scaffolding is always performed. The maximum legal value for this command is 2,147,483,647 (or 2^{31}-1).

Syntax

`CLUSTER_THRESHOLD` *`count`*

Options

`count`

Threshold instance count.

See also

SREF_SCAFFOLD on page 173

NEW_DISCRETE_CLIP_LSEGS

Note: This keyword is deprecated and will become obsolete in a future release.

Description

`NEW_DISCRETE_CLIP_LSEGS` (`ON` by default) modifies the behavior of `CLIP_LSEGS`. When `NEW_DISCRETE_CLIP_LSEGS` is `ON`, `CLIP_LSEGS` checks whether the given line segments are all horizontal, vertical, or +/- 45-degree angles before proceeding. If these criteria are not met, `CLIP_LSEGS` behaves as before. If these criteria are met, `CLIP_LSEGS` then clips the line segments against the given clip box by snapping to the nearest grid points of the value of DBU. The results are discretized to the size of half of DBU.

Syntax

```
NEW_DISCRETE_CLIP_LSEGS ON|OFF
```

Options

`ON`

Turns on modified behavior of `CLIP_LSEGS`. This is the default.

`OFF`

Turns off modified behavior of `CLIP_LSEGS`.

NEW_DYNAMIC_CORBASIC_ARRAYS

Note: This keyword is deprecated and will become obsolete in a future release.

Description

With this keyword the Proteus tool creates non-segment arrays and segment arrays with explicit dimensions when it first accesses them, instead of at parse time.

Syntax

```
NEW_DYNAMIC_CORBASIC_ARRAYS
```

Options

`ON`

Creates non-segment arrays and segment arrays with explicit dimensions only when they are accessed. This is the default.

`OFF`

Creates non-segment arrays and segment arrays with explicit dimensions when the recipe is parsed.

Example

For example, the following arrays are affected when the keyword is on:

```
DIM myArray
GLOBAL DIM SEGMENT_ARRAY mySegArray(10,20)
```

NEW_DYNAMIC_SEGMENT_ARRAYS

Note: This keyword is deprecated and will become obsolete in a future release.

Description

With this keyword the Proteus tool creates segment arrays with no explicit dimensions when it first accesses them, instead of at parse time. When the tool first accesses a segment array register on any segment, it creates a segment array for all segments on all `COR_IN` and `REC_IN` layers.

Syntax

```
NEW_DYNAMIC_SEGMENT_ARRAYS
```

Options

`ON`

Creates segment arrays with no explicit dimensions only when they are first accessed. This is the default.

`OFF`

Creates segment arrays with no explicit dimensions at parse time.

Example

For example, the following array is affected when the keyword is on:

```
DIM SEGMENT_ARRAY mySegArray
```

NEW_EDGE_TABLE_BY_LAYER

Note: This keyword is deprecated and will become obsolete in a future release.

Description

Enables the optimization of internal data structures to reduce the set up overhead for edge inspection functions (for example, `DIST_VECTOR`,

SELECT_SEGS, and FIND_SEGS), which occurs after each call to UPDATE_GRAPHICS.

Syntax

```
NEW_EDGE_TABLE_BY_LAYER ON|OFF
```

Options

ON

Turns on internal data structure optimization. This is the default.

OFF

Turns off internal data structure optimization.

NEW_MLO_DIMENSIONAL_INTERSECT_DEFAULT

Note: This keyword is deprecated and will become obsolete in a future release.

Description

For the MLO dimensional functions external1(), external1Edge(), external2(), external2Edge(), internal1(), and internal1Edge(), NEW_MLO_DIMENSIONAL_INTERSECT_DEFAULT (ON by default) causes the intersect options' default value to revert back to pre-G-2012.09-2 values. See the *Proteus Manufacturing Layer Operations (MLO) Reference Manual* for details.

Syntax

```
NEW_MLO_DIMENSIONAL_INTERSECT_DEFAULT ON|OFF
```

Options

ON

Turns on intersect = 'none' for MLO dimensional functions. This is the default.

OFF

Reverts to pre-G-2012.09-2 default values for dimensional functions' intersect option.

NEW_MLO_NONE_EXT

Note: This keyword is deprecated and will become obsolete in a future release.

Description

`NEW_MLO_NONE_EXT` (`ON` by default) causes the `extension = 'none'` option for MLO dimensional functions to behave slightly differently. The check region is not extended, as is true for the original `extension = 'none'` behavior, but, additionally, the check region is formed with right-angle boundaries at the edge endpoints. The right-angle boundaries of the check region are *exclusive*. The far boundary of the check region is inclusive or exclusive depending on the constraint of the spacing check value. See the *Proteus Manufacturing Layer Operations (MLO) Reference Manual* for details.

Syntax

```
NEW_MLO_NONE_EXT ON|OFF
```

Options

`ON`

Turns on different behavior for the `extension = 'none'` option. This is the default.

`OFF`

Uses original behavior for the `extension = 'none'` option, which is that the check region is not extended.

NEW_MSS_UNIFORM_CORRECTION

Note: This keyword is deprecated and will become obsolete in a future release.

Description

`NEW_MSS_UNIFORM_CORRECTION` (`ON` by default) controls whether or not to use an improved multi-segment solver (MSS) algorithm to correct template boundary issues.

Syntax

```
NEW_MSS_UNIFORM_CORRECTION ON|OFF
```

Options

`ON`

Turns on improved MSS algorithm. This is the default.

`OFF`

Turns off improved MSS algorithm.

NEW_MULTIPLE_OUTPUT_FILES

Note: This keyword is deprecated and will become obsolete in a future release.

Description

The `NEW_MULTIPLE_OUTPUT_FILES` keyword must be `ON` in the job control file when specifying multiple `OUTPUT` sections in the recipe.

The corexec application does not support recipes using `NEW_MULTIPLE_OUTPUT_FILES`. For this reason, corexec prints a message and will not run with a recipe that uses `NEW_MULTIPLE_OUTPUT_FILES`.

Syntax

```
NEW_MULTIPLE_OUTPUT_FILES ON|OFF
```

Options

`ON`

Enables the multiple output file functionality. If a `PROTEUS_JOB_FLOW` recipe specifies more than one output file, this keyword is turned `ON` automatically.

`OFF`

Disables the multiple output file functionality. This is the default.

NEW_PYTHON_TRUE_DIVISION

Note: This keyword is deprecated and will become obsolete in a future release.

Description

`NEW_PYTHON_TRUE_DIVISION` (`ON` by default) causes Python's division operator (/) to work as floating point division even when two integers are used. The default behavior for this operator when integers are used is integer division.

Syntax

```
NEW_PYTHON_TRUE_DIVISION ON|OFF
```

Options

`ON`

Turns on floating point division. This is the default.

`OFF`

Turns off floating point division.

Example

With `NEW_PYTHON_TRUE_DIVISION ON`: 1 / 2 == 0.5

With `NEW_PYTHON_TRUE_DIVISION OFF`: 1 / 2 == 0

NEW_SHIFT_LARGE_COORDINATE

Note: This keyword is deprecated and will become obsolete in a future release.

Description

Use `NEW_SHIFT_LARGE_COORDINATE` to implement coordinate processing that accepts the full range of values permitted under the GDSII standard. Any signed integer representable in 32-bit 2's-complement form is allowed as the value limit of the polygon vertex coordinate. Thus, the range of allowed values is less than 2^{31}-1 and greater than -2^{31}.

Input parameters that would overflow the 32-bit 2's-complement graphics representation internally are detected using hierarchy management, and rechecked prior to correction (in case the job control file is changed between the Hierarchy Manager and corexec).

You can override the limit computed by the Hierarchy Manager using `CELL_COORDINATE_LIMIT`.

Note: Although OASIS provides the capability to represent signed integers greater than 32 bits, the Proteus tool accepts only the range of values permitted under the GDSII standard.

Syntax

```
NEW_SHIFT_LARGE_COORDINATE ON|OFF
```

Options

`ON`

Turns on functionality. This is the default.

`OFF`

Turns off functionality.

See also

CELL_COORDINATE_LIMIT on page 175

NEW_SUPPRESS_WARNING_COUNT

Note: This keyword is deprecated and will become obsolete in a future release.

Description

This provides an improved format for the warning summary in the proteus stdout log, such that the specific count of warnings from parsing the job control file is suppressed, because this count did not account for warnings from other sources.

Syntax

```
NEW_SUPPRESS_WARNING_COUNT ON|OFF
```

Options

`ON`

Turns on improved log format. This is the default.

`OFF`

Turns off improved log format.

NO_RERUN_HIERMAN_FE

Note: This configuration file keyword is deprecated and will become obsolete in a future release. Use `NO_RERUN_HIERMAN` instead.

Description

Running proteus automatically runs hierarchical processing steps before beginning distributed processing unless the `NO_RERUN_HIERMAN_FE` configuration file option (`OFF` by default) is present. “FE” stands for “front end.”

See also

NO_RERUN_HIERMAN on page 257

PIPELINE_STRIPES

Note: This keyword is deprecated and will become obsolete in a future release.

Description

`PIPELINE_STRIPES` is only relevant when `PIPELINE_STRATEGY` is set to either `FRONT_LOAD` or `BACK_LOAD`.

The input chip is divided into stripes (or rows) for concurrent processing by different template blocks. The `PIPELINE_STRIPES` keyword controls the number of stripes, or the size of each data chunk to be processed using PPT. This value is important for tuning the pipeline performance.

Note: Do not include the `PIPELINE_STRIPES` and `STRIPE_HEIGHT` keywords in the same recipe.

Syntax

`PIPELINE_STRIPES` *n*

Options

n

Any non-negative integer. The default is 0.

Setting PIPELINE_STRIPES to 0 invokes auto mode, in which the actual number of stripes used is computed based on input chip size and MAX_CLUSTER value. For auto mode, the number of stripes is restricted to between 3 and 100 (inclusive) and calculated based on the ratio of layout height to stripe height. This is the recommended mode.

The stripe count is set to chip height divided by 10*MAX_CLUSTER. If the ratio is less than 3, PIPELINE_STRIPES is set to 3. If the ratio is greater than 100, PIPELINE_STRIPES is set to 100. Once stripe count is computed, stripe height is set to chip height divided by stripe count.

For performance experiments and improvements, PIPELINE_STRIPES can be set directly, but keep in mind that

- setting PIPELINE_STRIPES to 1 means that the entire input layout is processed in a single chunk, effectively disabling concurrency.
- if you specify a value less than 3 or greater than 100, a warning will be issued.

See also

STRIPE_HEIGHT on page 265

PROTEUS_PATH

Note: This environment variable is deprecated and will become obsolete in a future release.

Description

The PROTEUS_PATH environment variable defines the location of the executable files internally called by Proteus executables. Redefining PROTEUS_PATH can interfere with more common methods of locating executable files, such as PATH. The executable name is appended to PROTEUS_PATH, so directories should be defined with a slash (/) at the end.

Note: If you use PROTEUS_PATH with proteus to point to remote_server, hierman should reside in the same location. The REMOTE_SERVER_PATH configuration file keyword can only be used to point to remote_server.

See also

REMOTE_SERVER_PATH on page 258

USE_REVERSE_TEMPLATE_LIST

Description

Note: This configuration file keyword is deprecated and will become obsolete in a future release. Use `CORRECTION_ORDER` instead.

This option reverses the order in which templates are processed during correction. By reversing the template list, templates with longer processing times may be corrected earlier in the job, so that the correction job does not have to wait for a few templates to complete at the end.

Syntax

```
USE_REVERSE_TEMPLATE_LIST [ON|OFF]
```

Options

`ON`

Corrects templates in order of descending template number. This is the default.

`OFF`

Corrects templates in order of ascending template number. However, if proteus was invoked with the command-line option `-r`, this option is ignored.

See also

CORRECTION_ORDER on page 225

C

Log Files

Explains log files, with examples.

Understanding Log Files

After hierarchy management processing, proteus hierarchy management files are essentially a disk-based list of links back to the original input file. The Proteus tool writes a data structure that you can use to retrieve the data for a particular template for correction. The original GDS or OASIS is unchanged and proteus does not create any GDS or OASIS files.

Proteus log files follow the naming convention *jobname*.HMLOG, where *jobname* is the name you have given your Hierarchy Management job, controlled by the `JOBNAME` keyword, which defaults to `PROTEUS`.

Log File Example 1

The following is a typical log file, with explanatory comments added for each section.

```
hierman Release G-2012.09-8 Revision Proteus_G-2012.09-8_12Feb14-
2919019 (64f/64m LINUX_X86_64).
host: machine1
Proteus (TM) / PROTEUS (TM)
Version G-2012.09-8

*** Copyright (C) 1995 - 2014 Synopsys, Inc. ***
*** This software and the associated documentation are ***
*** confidential and proprietary to Synopsys, Inc. ***
*** Your use or disclosure of this software is subject to ***
*** the terms and conditions of a written license agreement ***
*** between you, or your company, and Synopsys, Inc. ***
*** ***

Testing for license PROTEUS_OPC...

Checking out PROTEUS_OPC...
License PROTEUS_OPC checked out.
```

In the following section, the hierarchy manager scans input file, records cell references and counts native cells and references in all layers in `INPUT`.

```
Reading input file /remote/ltg_pe1_us03/usr/large_2/
everest_0130_00.oas
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................

Scanning for Topcell(s)
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................
 Topcell: everest_gem
 native hierarchy: 1 cells; 0 refs
     Times: User: 0.09 Sys: 0.00 Elapsed: 0.11 Memory: 19.499M
Separating graphics from SREFs & AREFs.
 1 holder cell added
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.466M
Calculating clustering statistics.
     Times: User: 0.01 Sys: 0.00 Elapsed: 0.00 Memory: 19.722M
Cleaning 1D AREF transforms.
```

The following section is for OASIS only; cells needing graphics scaffolding are loaded into buffer files for faster access.

```
Create OASIS decompression buffer file.

    +0%----------+25%--------+50%--------+75%---------+100%
```

Graphics scaffolding (divides large graphics cells into smaller graphics cells):

```
Partitioning large graphic cells (multi-threaded).
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
 1 native cell divided into 217 smaller graphic cells (20000 x
20000)
     Times: User: 1.84 Sys: 0.04 Elapsed: 1.88 Memory: 29.707M
Subdividing large AREFs.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.534M
Constructing framework for revised hierarchy.
 revised hierarchy: 219 cells; 218 refs
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.534M
```

The following section shows instance generation (traversing the cell-reference tree-top down, stopping at cells smaller than `MAX_CLUSTER`. Exception: `MARK`s):

```
Generating instances for processing.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
 219 instances (217 cluster instances) identified.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.01 Memory: 19.977M
```

Leaf context analysis (sort instance by cell, push graphics to neighbors that are within ambit):

```
Calculating context for leaf cells.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.01 Memory: 24.379M
```

Holder context analysis happens here: the Hierarchy Manager traverses the cell reference tree from the bottom up, accumulating hash codes for holder cells. A hash code is the number generated from some combination of data, which is different for every combination of data. This results in a single number that you can use to compare two things to see whether they are the same. In Proteus hierarchy management, four numbers are used to represent the context that each instance would see, and as placement changes, those four numbers change, thus giving a new hash code, representative of the context seen by individual cells.

```
Calculating context for holding cells.
    +0%----------+25%--------+50%--------+75%---------+100%
    ..................................................
           Times: User: 75.26 Sys:    0.51
```

Template Generation for instances with unique hash numbers:

```
Calculating context for holding cells.
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.764M
Generating output files for correction.
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.760M

 220 output templates (218 cluster templates) generated for TB1:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 84966.3180 square microns.
 Sum of all template bounding areas: 79470.0657 square microns.
 Sum of all ambit-biased template bounding areas: 93312.0891
square microns.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.673M
Writing output topcell names.
 Topcell_out: 0N_everest_gem
     Times: User: 0.00 Sys: 0.01 Elapsed: 0.00 Memory: 19.684M
     Total Times: User: 3.58 Sys: 0.10 Elapsed: 4.31 Memory:
29.707M

 Hierarchy management complete for job HIERMAN; Wed Feb 12 12:19:11
2014
```

Log File Example 2

The following is an example of a log file from a proteus run with a `PROTEUS_JOB_FLOW` recipe with `PIPELINE_STRATEGY BACKLOAD`.

```
qrsh -P bhigh -V -cwd -now no -l model=EMT2700|EMT3000 -l
os_version=WS5.0 proteus -s snps:32 MOF_NONE_0210.pjx
/remote/ms_integ3_us03/SCM/DailyBuildReleases/proteus_G-
2012.09-8_12Feb14/amd64/bin/proteus Release G-2012.09-8 Revision
Proteus_G-2012.09-8_12Feb14-2919019 (64f/64m LINUX_X86_64).
host: machine1

 Proteus (TM) / PROTEUS (TM)
Version G-2012.09-8

*** Copyright (C) 1995 - 2014 Synopsys, Inc. ***
*** This software and the associated documentation are ***
*** confidential and proprietary to Synopsys, Inc. ***
*** Your use or disclosure of this software is subject to ***
*** the terms and conditions of a written license agreement ***
*** between you, or your company, and Synopsys, Inc. ***
*** ***

Testing for license PROTEUS_OPC...

Checking out PROTEUS_OPC...
License PROTEUS_OPC checked out.

Testing for license PA...

Checking out PA...
License PA checked out.
hierman -fe MOF_NONE_0210.pjx

 Hierarchy management started for job HIERMAN; Wed Feb 12 12:08:34
2014
Initialization completed.
     Times: User: 1.04 Sys: 0.02 Elapsed: 1.07 Memory: 17.543M

Reading input file /remote/ltg_pe1_us03/usr/large_2/
everest_0130_00.oas
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................

Scanning for Topcell(s)
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................
 Topcell: everest_gem
```

```
 native hierarchy: 1 cells; 0 refs
     Times: User: 0.08 Sys: 0.00 Elapsed: 0.10 Memory: 19.499M
Separating graphics from SREFs & AREFs.
 1 holder cell added
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.466M
Calculating clustering statistics.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.722M
Cleaning 1D AREF transforms.
Choosing Smart Block Compression cells.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.05 Sys: 0.00 Elapsed: 0.04 Memory: 19.501M
Create OASIS decompression buffer file.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.17 Sys: 0.01 Elapsed: 0.30 Memory: 19.732M
Partitioning large graphic cells (multi-threaded).
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
 1 native cell divided into 217 smaller graphic cells (20000 x
20000)
     Times: User: 1.83 Sys: 0.02 Elapsed: 1.89 Memory: 29.707M
Subdividing large AREFs.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.534M
Constructing framework for revised hierarchy.
 revised hierarchy: 219 cells; 218 refs
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.02 Memory: 19.538M
Generating instances for processing.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
 219 instances (217 cluster instances) identified.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.981M
Generating spatial bins for TopCell.
 Bin Count : 16 x 14 (horizontal x vertical)
 Bin Dimensions : (19590,19684) (x,y)
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.818M
Generating spatial bin cells.
 revised hierarchy: 444 cells; 1244 refs
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.864M
Generating flat bin instances.
 226 instances (224 cluster instances) identified.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.865M

Writing partial hierarchy results.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.665M
     Total Times: User: 3.17 Sys: 0.05 Elapsed: 3.47 Memory:
```

```
29.707M

 Hierarchy management complete for job HIERMAN; Wed Feb 12 12:08:37
2014
 Hierman FrontEnd Times:User: 7.80 Sys: 1.03 Elapsed: 9.44 Memory:
17.630M


 Cleaning up double patterning files from a previous run
 rm -rf ./temp/HIERMAN_TB1.DPT 2> /dev/null

 Cleaning up TB 1 fragment files and outputs from a previous run
 rm -rf ./gds/TB1_out.gds.dir/TB1/ 2> /dev/null
 rm -f HIERMAN_TB1_out.oas 2> /dev/null

 mkdir -p ./gds/TB1_out.gds.dir/TB1/ 2> /dev/null
 mkdir ./gds/TB1_out.gds.dir/TB1/TINF 2> /dev/null
 Cleaning up double patterning files from a previous run
 rm -rf ./temp/HIERMAN_TB2.DPT 2> /dev/null

 Cleaning up TB 2 fragment files and outputs from a previous run
 rm -rf ./gds/TB1_out.gds.dir/TB2/ 2> /dev/null
 rm -f HIERMAN_TB2_out.oas 2> /dev/null

 mkdir -p ./gds/TB1_out.gds.dir/TB2/ 2> /dev/null
 mkdir ./gds/TB1_out.gds.dir/TB2/TINF 2> /dev/null
 Cleaning up double patterning files from a previous run
 rm -rf ./temp/HIERMAN_TB3.DPT 2> /dev/null

 Cleaning up TB 3 fragment files and outputs from a previous run
 rm -f ./gds/TB1_out.gds.dir/* 2> /dev/null
 rm -f ./gds/TB1_out.gds 2> /dev/null
 rm -f ./gds/TB2_out.oas 2> /dev/null
 rm -f ./gds/MOF_BL_pcx2_out.gds 2> /dev/null
 rm -f ./gds/TB1_out.gds_saved 2> /dev/null
 rm -f ./gds/TB1_out.gds_tmp 2> /dev/null

 mkdir -p ./gds/TB1_out.gds.dir/ 2> /dev/null
 mkdir ./gds/TB1_out.gds.dir/TINF 2> /dev/null
 ./gds/TB1_out.gds output file created, Wed Feb 12 12:08:37 2014
 ./gds/TB2_out.oas output file created, Wed Feb 12 12:08:37 2014
 ./gds/MOF_BL_pcx2_out.gds output file created, Wed Feb 12
12:08:37 2014

 Pipeline flow started for TB 1, Wed Feb 12 12:08:37 2014

Generating spatial templates for TB 1.
 spatial context: 64 templates generated for stripe 1.1 of 3
trying to open port:2346
```

```
Generating spatial templates for TB 2.
remote_server snps:32 2346 3 0 1
running remote_server for SGE access, in
/remote/us03home4/usr/bin/gridForProteus/remote_server
 spatial context: 80 templates generated for stripe 1.2 of 3
 spatial context: 74 templates generated for stripe 1.3 of 3
remote_server ltgpe-03 2346 3 1 1
     Times: User: 0.84 Sys: 0.26 Elapsed: 1.17 Memory: 279.543M
Calculating context for holding cells.
 holder context: 10%
 holder context: 20%
 holder context: 30%
 holder context: 40%
 holder context: 50%
 holder context: 60%
 holder context: 70%
 holder context: 80%
 holder context: 90%
 holder context: Done Wed Feb 12 12:08:38 2014

     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 253.192M
 220 output templates (218 cluster templates) generated for TB1:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 84966.3180 square microns.
 Sum of all template bounding areas: 79470.0657 square microns.
 Sum of all ambit-biased template bounding areas: 93312.0891
square microns.

 Hierman BackEnd Times: User: 0.81 Sys: 0.11 Elapsed: 0.91 Memory:
279.543M
qsub -P iheavy -V -cwd -N dpsvr.25437 -j y -o ./logfiles.25437 -
t 1-32 .dpserver.25437
Requesting 32 dpservers on snps...
Client: executing mktop in a separate thread...
Grid job number: 472681
dpserver log files are in logfiles.25437
sh -c "{ { dpserver -p2346 -cltgpe-03 -V3 >>logfile.txt.ltgpe-
03.25560; } 3>&1 1>&2 2>&3 |tee logfile.txt.ltgpe-03.25560; }
3>&1 1>&2 2>&3"
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 0.91% Completed ~ 0.00% area (from TB 1)
Client: 1:1:INIT :N_TMPL=220 SOCK_CNT=1 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 63 done, ~ 1.36% Completed ~ 0.46% area (from TB 1.1)
 # 62 done, ~ 1.82% Completed ~ 0.92% area (from TB 1.1)
```

```
 # 61 done, ~ 2.27% Completed ~ 1.37% area (from TB 1.1)
 # 60 done, ~ 2.73% Completed ~ 1.80% area (from TB 1.1)
 # 59 done, ~ 3.18% Completed ~ 2.29% area (from TB 1.1)
 # 58 done, ~ 3.64% Completed ~ 2.78% area (from TB 1.1)
 # 57 done, ~ 4.09% Completed ~ 3.27% area (from TB 1.1)
 # 56 done, ~ 4.55% Completed ~ 3.73% area (from TB 1.1)
 # 55 done, ~ 5.00% Completed ~ 4.22% area (from TB 1.1)
 # 54 done, ~ 5.45% Completed ~ 4.71% area (from TB 1.1)
 # 53 done, ~ 5.91% Completed ~ 5.20% area (from TB 1.1)
 # 52 done, ~ 6.36% Completed ~ 5.66% area (from TB 1.1)
 # 51 done, ~ 6.82% Completed ~ 6.15% area (from TB 1.1)
 # 50 done, ~ 7.27% Completed ~ 6.64% area (from TB 1.1)
 # 49 done, ~ 7.73% Completed ~ 7.13% area (from TB 1.1)
 # 48 done, ~ 8.18% Completed ~ 7.59% area (from TB 1.1)
 # 47 done, ~ 8.64% Completed ~ 8.08% area (from TB 1.1)
 # 46 done, ~ 9.09% Completed ~ 8.57% area (from TB 1.1)
 # 45 done, ~ 9.55% Completed ~ 9.06% area (from TB 1.1)
 # 44 done, ~ 10.00% Completed ~ 9.52% area (from TB 1.1)
 # 43 done, ~ 10.45% Completed ~ 10.01% area (from TB 1.1)
 # 42 done, ~ 10.91% Completed ~ 10.50% area (from TB 1.1)
 # 41 done, ~ 11.36% Completed ~ 10.99% area (from TB 1.1)
 # 40 done, ~ 11.82% Completed ~ 11.44% area (from TB 1.1)
 # 39 done, ~ 12.27% Completed ~ 11.93% area (from TB 1.1)
 # 38 done, ~ 12.73% Completed ~ 12.42% area (from TB 1.1)
 # 37 done, ~ 13.18% Completed ~ 12.91% area (from TB 1.1)
 # 36 done, ~ 13.64% Completed ~ 13.37% area (from TB 1.1)
 # 35 done, ~ 14.09% Completed ~ 13.86% area (from TB 1.1)
 # 34 done, ~ 14.55% Completed ~ 14.35% area (from TB 1.1)
 # 33 done, ~ 15.00% Completed ~ 14.84% area (from TB 1.1)
 # 32 done, ~ 15.45% Completed ~ 15.30% area (from TB 1.1)
 # 31 done, ~ 15.91% Completed ~ 15.79% area (from TB 1.1)
 # 30 done, ~ 16.36% Completed ~ 16.28% area (from TB 1.1)
 # 29 done, ~ 16.82% Completed ~ 16.77% area (from TB 1.1)
 # 28 done, ~ 17.27% Completed ~ 17.23% area (from TB 1.1)
 # 27 done, ~ 17.73% Completed ~ 17.72% area (from TB 1.1)
 # 26 done, ~ 18.18% Completed ~ 18.21% area (from TB 1.1)
 # 25 done, ~ 18.64% Completed ~ 18.70% area (from TB 1.1)
 # 24 done, ~ 19.09% Completed ~ 19.16% area (from TB 1.1)
 # 23 done, ~ 19.55% Completed ~ 19.65% area (from TB 1.1)
 # 22 done, ~ 20.00% Completed ~ 20.13% area (from TB 1.1)
 # 21 done, ~ 20.45% Completed ~ 20.62% area (from TB 1.1)
Client: 2:1:INIT :N_TMPL=220 SOCK_CNT=2 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 3:1:INIT :N_TMPL=220 SOCK_CNT=3 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 4:1:INIT :N_TMPL=220 SOCK_CNT=4 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
```

```
LASTTB=3
Client: 5:1:INIT :N_TMPL=220 SOCK_CNT=5 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 6:1:INIT :N_TMPL=220 SOCK_CNT=6 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 7:1:INIT :N_TMPL=220 SOCK_CNT=7 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 8:1:INIT :N_TMPL=220 SOCK_CNT=8 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 20 done, ~ 20.91% Completed ~ 21.08% area (from TB 1.1)
Client: 9:1:INIT :N_TMPL=220 SOCK_CNT=9 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:10:1:INIT :N_TMPL=220 SOCK_CNT=10 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:11:1:INIT :N_TMPL=220 SOCK_CNT=11 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:12:1:INIT :N_TMPL=220 SOCK_CNT=12 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:13:1:INIT :N_TMPL=220 SOCK_CNT=13 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:14:1:INIT :N_TMPL=220 SOCK_CNT=14 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:15:1:INIT :N_TMPL=220 SOCK_CNT=15 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:16:1:INIT :N_TMPL=220 SOCK_CNT=16 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:17:1:INIT :N_TMPL=220 SOCK_CNT=17 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:18:1:INIT :N_TMPL=220 SOCK_CNT=18 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:19:1:INIT :N_TMPL=220 SOCK_CNT=19 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:20:1:INIT :N_TMPL=220 SOCK_CNT=20 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 19 done, ~ 21.36% Completed ~ 21.57% area (from TB 1.1)
```

```
Client:21:1:INIT :N_TMPL=220 SOCK_CNT=21 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 18 done, ~ 21.82% Completed ~ 22.06% area (from TB 1.1)
 # 14 done, ~ 22.27% Completed ~ 22.55% area (from TB 1.1)
 # 16 done, ~ 22.73% Completed ~ 23.01% area (from TB 1.1)
 # 17 done, ~ 23.18% Completed ~ 23.50% area (from TB 1.1)
 # 13 done, ~ 23.64% Completed ~ 23.99% area (from TB 1.1)
 # 10 done, ~ 24.09% Completed ~ 24.48% area (from TB 1.1)
 # 12 done, ~ 24.55% Completed ~ 24.94% area (from TB 1.1)
 # 6 done, ~ 25.00% Completed ~ 25.43% area (from TB 1.1)
 # 15 done, ~ 25.45% Completed ~ 25.92% area (from TB 1.1)
 # 9 done, ~ 25.91% Completed ~ 26.41% area (from TB 1.1)
Client:22:1:INIT :N_TMPL=220 SOCK_CNT=22 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 8 done, ~ 26.36% Completed ~ 26.87% area (from TB 1.1)
Client:23:1:INIT :N_TMPL=220 SOCK_CNT=23 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 2 done, ~ 26.82% Completed ~ 27.33% area (from TB 1.1)
Client:24:1:INIT :N_TMPL=220 SOCK_CNT=24 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 11 done, ~ 27.27% Completed ~ 27.82% area (from TB 1.1)
Client:25:1:INIT :N_TMPL=220 SOCK_CNT=25 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 7 done, ~ 27.73% Completed ~ 28.31% area (from TB 1.1)
 # 4 done, ~ 28.18% Completed ~ 28.76% area (from TB 1.1)
 # 3 done, ~ 28.64% Completed ~ 29.22% area (from TB 1.1)
 # 139 done, ~ 29.09% Completed ~ 29.68% area (from TB 1.2)
 # 141 done, ~ 29.55% Completed ~ 30.14% area (from TB 1.2)
 # 140 done, ~ 30.00% Completed ~ 30.60% area (from TB 1.2)
 # 142 done, ~ 30.45% Completed ~ 31.06% area (from TB 1.2)
 # 5 done, ~ 30.91% Completed ~ 31.55% area (from TB 1.1)
 # 0 done, ~ 31.36% Completed ~ 31.98% area (from TB 1.1)
 # 143 done, ~ 31.82% Completed ~ 32.43% area (from TB 1.2)
 # 136 done, ~ 32.27% Completed ~ 32.92% area (from TB 1.2)
 # 135 done, ~ 32.73% Completed ~ 33.41% area (from TB 1.2)
 # 138 done, ~ 33.18% Completed ~ 33.90% area (from TB 1.2)
 # 1 done, ~ 33.64% Completed ~ 34.36% area (from TB 1.1)
 # 126 done, ~ 34.09% Completed ~ 34.85% area (from TB 1.2)
 # 133 done, ~ 34.55% Completed ~ 35.34% area (from TB 1.2)
 # 128 done, ~ 35.00% Completed ~ 35.83% area (from TB 1.2)
 # 137 done, ~ 35.45% Completed ~ 36.32% area (from TB 1.2)
 # 127 done, ~ 35.91% Completed ~ 36.81% area (from TB 1.2)
 # 134 done, ~ 36.36% Completed ~ 37.30% area (from TB 1.2)
 # 125 done, ~ 36.82% Completed ~ 37.79% area (from TB 1.2)
```

```
 # 131 done, ~ 37.27% Completed ~ 38.28% area (from TB 1.2)
 # 130 done, ~ 37.73% Completed ~ 38.77% area (from TB 1.2)
 # 132 done, ~ 38.18% Completed ~ 39.26% area (from TB 1.2)
 # 129 done, ~ 38.64% Completed ~ 39.75% area (from TB 1.2)
 # 121 done, ~ 39.09% Completed ~ 40.24% area (from TB 1.2)
 # 116 done, ~ 39.55% Completed ~ 40.73% area (from TB 1.2)
Client:26:1:INIT :N_TMPL=220 SOCK_CNT=26 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 124 done, ~ 40.00% Completed ~ 41.22% area (from TB 1.2)
 # 122 done, ~ 40.45% Completed ~ 41.71% area (from TB 1.2)
 # 106 done, ~ 40.91% Completed ~ 42.20% area (from TB 1.2)
 # 108 done, ~ 41.36% Completed ~ 42.68% area (from TB 1.2)
 # 111 done, ~ 41.82% Completed ~ 43.17% area (from TB 1.2)
 # 123 done, ~ 42.27% Completed ~ 43.66% area (from TB 1.2)
Client:27:1:INIT :N_TMPL=220 SOCK_CNT=27 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 107 done, ~ 42.73% Completed ~ 44.15% area (from TB 1.2)
 # 118 done, ~ 43.18% Completed ~ 44.64% area (from TB 1.2)
 # 120 done, ~ 43.64% Completed ~ 45.13% area (from TB 1.2)
Client:28:1:INIT :N_TMPL=220 SOCK_CNT=28 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:29:1:INIT :N_TMPL=220 SOCK_CNT=29 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 114 done, ~ 44.09% Completed ~ 45.62% area (from TB 1.2)
 # 117 done, ~ 44.55% Completed ~ 46.11% area (from TB 1.2)
 # 105 done, ~ 45.00% Completed ~ 46.60% area (from TB 1.2)
 # 104 done, ~ 45.45% Completed ~ 47.08% area (from TB 1.2)
 # 119 done, ~ 45.91% Completed ~ 47.57% area (from TB 1.2)
 # 103 done, ~ 46.36% Completed ~ 48.03% area (from TB 1.2)
 # 113 done, ~ 46.82% Completed ~ 48.52% area (from TB 1.2)
 # 102 done, ~ 47.27% Completed ~ 49.01% area (from TB 1.2)
Client:30:1:INIT :N_TMPL=220 SOCK_CNT=30 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 110 done, ~ 47.73% Completed ~ 49.50% area (from TB 1.2)
 # 98 done, ~ 48.18% Completed ~ 49.99% area (from TB 1.2)
 # 109 done, ~ 48.64% Completed ~ 50.48% area (from TB 1.2)
 # 115 done, ~ 49.09% Completed ~ 50.97% area (from TB 1.2)
Client:31:1:INIT :N_TMPL=220 SOCK_CNT=31 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 100 done, ~ 49.55% Completed ~ 51.46% area (from TB 1.2)
 # 99 done, ~ 50.00% Completed ~ 51.95% area (from TB 1.2)
 # 112 done, ~ 50.45% Completed ~ 52.44% area (from TB 1.2)
 # 101 done, ~ 50.91% Completed ~ 52.93% area (from TB 1.2)
```

```
Client:32:1:INIT :N_TMPL=220 SOCK_CNT=32 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 97 done, ~ 51.36% Completed ~ 53.42% area (from TB 1.2)
 # 93 done, ~ 51.82% Completed ~ 53.90% area (from TB 1.2)
Client:33:1:INIT :N_TMPL=220 SOCK_CNT=33 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 88 done, ~ 52.27% Completed ~ 54.39% area (from TB 1.2)
 # 96 done, ~ 52.73% Completed ~ 54.88% area (from TB 1.2)
 # 78 done, ~ 53.18% Completed ~ 55.13% area (from TB 1.2)
 # 83 done, ~ 53.64% Completed ~ 55.62% area (from TB 1.2)
 # 95 done, ~ 54.09% Completed ~ 56.11% area (from TB 1.2)
 # 94 done, ~ 54.55% Completed ~ 56.60% area (from TB 1.2)
 # 91 done, ~ 55.00% Completed ~ 57.09% area (from TB 1.2)
 # 73 done, ~ 55.45% Completed ~ 57.34% area (from TB 1.2)
 # 87 done, ~ 55.91% Completed ~ 57.83% area (from TB 1.2)
 # 92 done, ~ 56.36% Completed ~ 58.32% area (from TB 1.2)
 # 68 done, ~ 56.82% Completed ~ 58.55% area (from TB 1.2)
 # 90 done, ~ 57.27% Completed ~ 59.04% area (from TB 1.2)
 # 85 done, ~ 57.73% Completed ~ 59.53% area (from TB 1.2)
 # 84 done, ~ 58.18% Completed ~ 60.02% area (from TB 1.2)
 # 86 done, ~ 58.64% Completed ~ 60.51% area (from TB 1.2)
 # 89 done, ~ 59.09% Completed ~ 61.00% area (from TB 1.2)
 # 79 done, ~ 59.55% Completed ~ 61.49% area (from TB 1.2)
 # 82 done, ~ 60.00% Completed ~ 61.98% area (from TB 1.2)
 # 80 done, ~ 60.45% Completed ~ 62.47% area (from TB 1.2)
 # 72 done, ~ 60.91% Completed ~ 62.96% area (from TB 1.2)
 # 76 done, ~ 61.36% Completed ~ 63.45% area (from TB 1.2)
 # 75 done, ~ 61.82% Completed ~ 63.94% area (from TB 1.2)
 # 77 done, ~ 62.27% Completed ~ 64.43% area (from TB 1.2)
 # 67 done, ~ 62.73% Completed ~ 64.89% area (from TB 1.2)
 # 66 done, ~ 63.18% Completed ~ 65.35% area (from TB 1.2)
 # 201 done, ~ 63.64% Completed ~ 65.83% area (from TB 1.3)
 # 207 done, ~ 64.09% Completed ~ 66.29% area (from TB 1.3)
 # 202 done, ~ 64.55% Completed ~ 66.75% area (from TB 1.3)
 # 81 done, ~ 65.00% Completed ~ 67.24% area (from TB 1.2)
 # 200 done, ~ 65.45% Completed ~ 67.73% area (from TB 1.3)
 # 196 done, ~ 65.91% Completed ~ 67.84% area (from TB 1.3)
 # 71 done, ~ 66.36% Completed ~ 68.32% area (from TB 1.2)
 # 204 done, ~ 66.82% Completed ~ 68.81% area (from TB 1.3)
 # 65 done, ~ 67.27% Completed ~ 69.27% area (from TB 1.2)
 # 69 done, ~ 67.73% Completed ~ 69.76% area (from TB 1.2)
 # 205 done, ~ 68.18% Completed ~ 70.25% area (from TB 1.3)
 # 215 done, ~ 68.64% Completed ~ 70.71% area (from TB 1.3)
 # 64 done, ~ 69.09% Completed ~ 71.17% area (from TB 1.2)
 # 191 done, ~ 69.55% Completed ~ 71.27% area (from TB 1.3)
 # 70 done, ~ 70.00% Completed ~ 71.76% area (from TB 1.2)
 # 216 done, ~ 70.45% Completed ~ 72.22% area (from TB 1.3)
```

```
# 74 done, ~ 70.91% Completed ~ 72.71% area (from TB 1.2)
# 206 done, ~ 71.36% Completed ~ 73.20% area (from TB 1.3)
# 197 done, ~ 71.82% Completed ~ 73.66% area (from TB 1.3)
# 212 done, ~ 72.27% Completed ~ 74.12% area (from TB 1.3)
# 186 done, ~ 72.73% Completed ~ 74.22% area (from TB 1.3)
# 199 done, ~ 73.18% Completed ~ 74.71% area (from TB 1.3)
# 181 done, ~ 73.64% Completed ~ 74.80% area (from TB 1.3)
# 217 done, ~ 74.09% Completed ~ 75.23% area (from TB 1.3)
# 213 done, ~ 74.55% Completed ~ 75.69% area (from TB 1.3)
# 211 done, ~ 75.00% Completed ~ 76.18% area (from TB 1.3)
# 203 done, ~ 75.45% Completed ~ 76.67% area (from TB 1.3)
# 180 done, ~ 75.91% Completed ~ 76.87% area (from TB 1.3)
# 209 done, ~ 76.36% Completed ~ 77.36% area (from TB 1.3)
# 198 done, ~ 76.82% Completed ~ 77.85% area (from TB 1.3)
# 214 done, ~ 77.27% Completed ~ 78.31% area (from TB 1.3)
# 192 done, ~ 77.73% Completed ~ 78.77% area (from TB 1.3)
# 176 done, ~ 78.18% Completed ~ 79.26% area (from TB 1.3)
# 179 done, ~ 78.64% Completed ~ 79.54% area (from TB 1.3)
# 178 done, ~ 79.09% Completed ~ 79.99% area (from TB 1.3)
# 182 done, ~ 79.55% Completed ~ 80.45% area (from TB 1.3)
# 177 done, ~ 80.00% Completed ~ 80.91% area (from TB 1.3)
# 187 done, ~ 80.45% Completed ~ 81.37% area (from TB 1.3)
# 208 done, ~ 80.91% Completed ~ 81.86% area (from TB 1.3)
# 210 done, ~ 81.36% Completed ~ 82.35% area (from TB 1.3)
# 175 done, ~ 81.82% Completed ~ 82.72% area (from TB 1.3)
# 154 done, ~ 82.27% Completed ~ 82.76% area (from TB 1.3)
# 195 done, ~ 82.73% Completed ~ 83.12% area (from TB 1.3)
# 190 done, ~ 83.18% Completed ~ 83.48% area (from TB 1.3)
# 153 done, ~ 83.64% Completed ~ 83.52% area (from TB 1.3)
# 167 done, ~ 84.09% Completed ~ 83.98% area (from TB 1.3)
# 172 done, ~ 84.55% Completed ~ 84.44% area (from TB 1.3)
# 174 done, ~ 85.00% Completed ~ 84.82% area (from TB 1.3)
# 185 done, ~ 85.45% Completed ~ 85.18% area (from TB 1.3)
# 183 done, ~ 85.91% Completed ~ 85.67% area (from TB 1.3)
# 171 done, ~ 86.36% Completed ~ 86.16% area (from TB 1.3)
# 194 done, ~ 86.82% Completed ~ 86.65% area (from TB 1.3)
# 166 done, ~ 87.27% Completed ~ 87.14% area (from TB 1.3)
# 189 done, ~ 87.73% Completed ~ 87.62% area (from TB 1.3)
# 156 done, ~ 88.18% Completed ~ 88.11% area (from TB 1.3)
# 162 done, ~ 88.64% Completed ~ 88.57% area (from TB 1.3)
# 188 done, ~ 89.09% Completed ~ 89.06% area (from TB 1.3)
# 155 done, ~ 89.55% Completed ~ 89.55% area (from TB 1.3)
# 173 done, ~ 90.00% Completed ~ 90.01% area (from TB 1.3)
# 152 done, ~ 90.45% Completed ~ 90.47% area (from TB 1.3)
# 157 done, ~ 90.91% Completed ~ 90.93% area (from TB 1.3)
# 170 done, ~ 91.36% Completed ~ 91.42% area (from TB 1.3)
# 161 done, ~ 91.82% Completed ~ 91.91% area (from TB 1.3)
# 193 done, ~ 92.27% Completed ~ 92.39% area (from TB 1.3)
```

```
 # 150 done, ~ 92.73% Completed ~ 92.69% area (from TB 1.3)
 # 184 done, ~ 93.18% Completed ~ 93.18% area (from TB 1.3)
 # 147 done, ~ 93.64% Completed ~ 93.48% area (from TB 1.3)
 # 151 done, ~ 94.09% Completed ~ 93.97% area (from TB 1.3)
 # 168 done, ~ 94.55% Completed ~ 94.46% area (from TB 1.3)
 # 169 done, ~ 95.00% Completed ~ 94.95% area (from TB 1.3)
 # 163 done, ~ 95.45% Completed ~ 95.44% area (from TB 1.3)
 # 144 done, ~ 95.91% Completed ~ 95.71% area (from TB 1.3)
 # 160 done, ~ 96.36% Completed ~ 96.20% area (from TB 1.3)
 # 158 done, ~ 96.82% Completed ~ 96.69% area (from TB 1.3)
 # 149 done, ~ 97.27% Completed ~ 97.15% area (from TB 1.3)
 # 146 done, ~ 97.73% Completed ~ 97.58% area (from TB 1.3)
 # 165 done, ~ 98.18% Completed ~ 98.07% area (from TB 1.3)
 # 159 done, ~ 98.64% Completed ~ 98.56% area (from TB 1.3)
 # 164 done, ~ 99.09% Completed ~ 99.05% area (from TB 1.3)
 # 148 done, ~ 99.55% Completed ~ 99.54% area (from TB 1.3)
 # 145 done, ~ 100.00% Completed ~ 100.00% area (from TB 1.3)

Concurrent[1]: All data is present in the output for TB 1.

Pipeline flow complete for TB 1, Wed Feb 12 12:09:32 2014

 TB1 Times:User: 1.08 Sys: 0.36 Elapsed: 55.31 Memory: 280.094M

 Pipeline flow started for TB 2, Wed Feb 12 12:09:32 2014

 spatial context: 64 templates generated for stripe 2.1 of 3
Generating spatial templates for TB 3.
 spatial context: 80 templates generated for stripe 2.2 of 3
     Times: User: 0.96 Sys: 0.65 Elapsed: 55.19 Memory: 379.800M
 spatial context: 74 templates generated for stripe 2.3 of 3
     Times: User: 0.12 Sys: 0.00 Elapsed: 0.13 Memory: 376.850M
Calculating context for holding cells.
 holder context: 10%
 holder context: 20%
 holder context: 30%
 holder context: 40%
 holder context: 50%
 holder context: 60%
 holder context: 70%
 holder context: 80%
 holder context: 90%
 holder context: Done Wed Feb 12 12:09:34 2014

     Times: User: 0.00 Sys: 0.00 Elapsed: 0.03 Memory: 371.229M
 220 output templates (218 cluster templates) generated for TB2:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
```

```
 Total topcell bounding area: 85200.1608 square microns.
 Sum of all template bounding areas: 78141.6062 square microns.
 Sum of all ambit-biased template bounding areas: 99135.4797
square microns.

 Hierman BackEnd Times: User: 1.89 Sys: 0.76 Elapsed: 56.27 Memory:
379.800M
Client: executing mktop in a separate thread...
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 0.91% Completed ~ 0.00% area (from TB 2)
 # 48 done, ~ 1.36% Completed ~ 0.50% area (from TB 2.1)
 # 55 done, ~ 1.82% Completed ~ 0.68% area (from TB 2.1)
 # 44 done, ~ 2.27% Completed ~ 0.85% area (from TB 2.1)
 # 56 done, ~ 2.73% Completed ~ 1.20% area (from TB 2.1)
 # 39 done, ~ 3.18% Completed ~ 1.70% area (from TB 2.1)
 # 45 done, ~ 3.64% Completed ~ 2.16% area (from TB 2.1)
 # 40 done, ~ 4.09% Completed ~ 2.46% area (from TB 2.1)
 # 52 done, ~ 4.55% Completed ~ 2.93% area (from TB 2.1)
 # 41 done, ~ 5.00% Completed ~ 3.23% area (from TB 2.1)
 # 58 done, ~ 5.45% Completed ~ 3.72% area (from TB 2.1)
 # 60 done, ~ 5.91% Completed ~ 4.19% area (from TB 2.1)
 # 35 done, ~ 6.36% Completed ~ 4.69% area (from TB 2.1)
 # 31 done, ~ 6.82% Completed ~ 5.19% area (from TB 2.1)
 # 59 done, ~ 7.27% Completed ~ 5.68% area (from TB 2.1)
 # 49 done, ~ 7.73% Completed ~ 6.18% area (from TB 2.1)
 # 46 done, ~ 8.18% Completed ~ 6.65% area (from TB 2.1)
 # 53 done, ~ 8.64% Completed ~ 7.12% area (from TB 2.1)
 # 54 done, ~ 9.09% Completed ~ 7.57% area (from TB 2.1)
 # 51 done, ~ 9.55% Completed ~ 8.04% area (from TB 2.1)
 # 27 done, ~ 10.00% Completed ~ 8.53% area (from TB 2.1)
 # 50 done, ~ 10.45% Completed ~ 9.00% area (from TB 2.1)
 # 57 done, ~ 10.91% Completed ~ 9.46% area (from TB 2.1)
 # 63 done, ~ 11.36% Completed ~ 9.93% area (from TB 2.1)
 # 26 done, ~ 11.82% Completed ~ 10.42% area (from TB 2.1)
 # 47 done, ~ 12.27% Completed ~ 10.92% area (from TB 2.1)
 # 61 done, ~ 12.73% Completed ~ 11.42% area (from TB 2.1)
 # 34 done, ~ 13.18% Completed ~ 11.91% area (from TB 2.1)
 # 25 done, ~ 13.64% Completed ~ 12.41% area (from TB 2.1)
 # 30 done, ~ 14.09% Completed ~ 12.90% area (from TB 2.1)
 # 38 done, ~ 14.55% Completed ~ 13.40% area (from TB 2.1)
 # 62 done, ~ 15.00% Completed ~ 13.90% area (from TB 2.1)
 # 42 done, ~ 15.45% Completed ~ 14.39% area (from TB 2.1)
 # 24 done, ~ 15.91% Completed ~ 14.86% area (from TB 2.1)
 # 12 done, ~ 16.36% Completed ~ 15.33% area (from TB 2.1)
 # 11 done, ~ 16.82% Completed ~ 15.83% area (from TB 2.1)
 # 20 done, ~ 17.27% Completed ~ 16.30% area (from TB 2.1)
```

```
# 16 done, ~ 17.73% Completed ~ 16.77% area (from TB 2.1)
# 43 done, ~ 18.18% Completed ~ 17.26% area (from TB 2.1)
# 32 done, ~ 18.64% Completed ~ 17.76% area (from TB 2.1)
# 37 done, ~ 19.09% Completed ~ 18.23% area (from TB 2.1)
# 33 done, ~ 19.55% Completed ~ 18.70% area (from TB 2.1)
# 36 done, ~ 20.00% Completed ~ 19.20% area (from TB 2.1)
# 28 done, ~ 20.45% Completed ~ 19.69% area (from TB 2.1)
# 29 done, ~ 20.91% Completed ~ 20.16% area (from TB 2.1)
# 10 done, ~ 21.36% Completed ~ 20.66% area (from TB 2.1)
# 23 done, ~ 21.82% Completed ~ 21.15% area (from TB 2.1)
# 9 done, ~ 22.27% Completed ~ 21.65% area (from TB 2.1)
# 0 done, ~ 22.73% Completed ~ 22.12% area (from TB 2.1)
# 3 done, ~ 23.18% Completed ~ 22.56% area (from TB 2.1)
# 2 done, ~ 23.64% Completed ~ 23.03% area (from TB 2.1)
# 7 done, ~ 24.09% Completed ~ 23.50% area (from TB 2.1)
# 15 done, ~ 24.55% Completed ~ 24.00% area (from TB 2.1)
# 19 done, ~ 25.00% Completed ~ 24.50% area (from TB 2.1)
# 1 done, ~ 25.45% Completed ~ 24.97% area (from TB 2.1)
# 8 done, ~ 25.91% Completed ~ 25.46% area (from TB 2.1)
# 5 done, ~ 26.36% Completed ~ 25.96% area (from TB 2.1)
# 4 done, ~ 26.82% Completed ~ 26.46% area (from TB 2.1)
# 6 done, ~ 27.27% Completed ~ 26.95% area (from TB 2.1)
# 143 done, ~ 27.73% Completed ~ 27.42% area (from TB 2.2)
# 135 done, ~ 28.18% Completed ~ 27.92% area (from TB 2.2)
# 22 done, ~ 28.64% Completed ~ 28.41% area (from TB 2.1)
# 136 done, ~ 29.09% Completed ~ 28.91% area (from TB 2.2)
# 21 done, ~ 29.55% Completed ~ 29.40% area (from TB 2.1)
# 13 done, ~ 30.00% Completed ~ 29.90% area (from TB 2.1)
# 14 done, ~ 30.45% Completed ~ 30.39% area (from TB 2.1)
# 138 done, ~ 30.91% Completed ~ 30.89% area (from TB 2.2)
# 137 done, ~ 31.36% Completed ~ 31.38% area (from TB 2.2)
# 116 done, ~ 31.82% Completed ~ 31.62% area (from TB 2.2)
# 18 done, ~ 32.27% Completed ~ 32.12% area (from TB 2.1)
# 124 done, ~ 32.73% Completed ~ 32.61% area (from TB 2.2)
# 119 done, ~ 33.18% Completed ~ 33.08% area (from TB 2.2)
# 17 done, ~ 33.64% Completed ~ 33.57% area (from TB 2.1)
# 131 done, ~ 34.09% Completed ~ 34.07% area (from TB 2.2)
# 113 done, ~ 34.55% Completed ~ 34.33% area (from TB 2.2)
# 110 done, ~ 35.00% Completed ~ 34.57% area (from TB 2.2)
# 123 done, ~ 35.45% Completed ~ 35.06% area (from TB 2.2)
# 140 done, ~ 35.91% Completed ~ 35.56% area (from TB 2.2)
# 122 done, ~ 36.36% Completed ~ 36.02% area (from TB 2.2)
# 139 done, ~ 36.82% Completed ~ 36.52% area (from TB 2.2)
# 142 done, ~ 37.27% Completed ~ 37.01% area (from TB 2.2)
# 141 done, ~ 37.73% Completed ~ 37.51% area (from TB 2.2)
# 130 done, ~ 38.18% Completed ~ 38.00% area (from TB 2.2)
# 128 done, ~ 38.64% Completed ~ 38.47% area (from TB 2.2)
# 107 done, ~ 39.09% Completed ~ 38.97% area (from TB 2.2)
```

```
# 129 done, ~ 39.55% Completed ~ 39.47% area (from TB 2.2)
# 104 done, ~ 40.00% Completed ~ 39.96% area (from TB 2.2)
# 101 done, ~ 40.45% Completed ~ 40.46% area (from TB 2.2)
# 102 done, ~ 40.91% Completed ~ 40.96% area (from TB 2.2)
# 127 done, ~ 41.36% Completed ~ 41.45% area (from TB 2.2)
# 117 done, ~ 41.82% Completed ~ 41.92% area (from TB 2.2)
# 118 done, ~ 42.27% Completed ~ 42.39% area (from TB 2.2)
# 120 done, ~ 42.73% Completed ~ 42.89% area (from TB 2.2)
# 98 done, ~ 43.18% Completed ~ 43.35% area (from TB 2.2)
# 132 done, ~ 43.64% Completed ~ 43.85% area (from TB 2.2)
# 126 done, ~ 44.09% Completed ~ 44.34% area (from TB 2.2)
# 105 done, ~ 44.55% Completed ~ 44.84% area (from TB 2.2)
# 99 done, ~ 45.00% Completed ~ 45.31% area (from TB 2.2)
# 114 done, ~ 45.45% Completed ~ 45.80% area (from TB 2.2)
# 111 done, ~ 45.91% Completed ~ 46.30% area (from TB 2.2)
# 134 done, ~ 46.36% Completed ~ 46.79% area (from TB 2.2)
# 125 done, ~ 46.82% Completed ~ 47.29% area (from TB 2.2)
# 106 done, ~ 47.27% Completed ~ 47.79% area (from TB 2.2)
# 100 done, ~ 47.73% Completed ~ 48.28% area (from TB 2.2)
# 108 done, ~ 48.18% Completed ~ 48.78% area (from TB 2.2)
# 133 done, ~ 48.64% Completed ~ 49.27% area (from TB 2.2)
# 97 done, ~ 49.09% Completed ~ 49.77% area (from TB 2.2)
# 94 done, ~ 49.55% Completed ~ 50.26% area (from TB 2.2)
# 103 done, ~ 50.00% Completed ~ 50.76% area (from TB 2.2)
# 115 done, ~ 50.45% Completed ~ 51.25% area (from TB 2.2)
# 109 done, ~ 50.91% Completed ~ 51.75% area (from TB 2.2)
# 121 done, ~ 51.36% Completed ~ 52.25% area (from TB 2.2)
# 88 done, ~ 51.82% Completed ~ 52.74% area (from TB 2.2)
# 217 done, ~ 52.27% Completed ~ 52.78% area (from TB 2.3)
# 91 done, ~ 52.73% Completed ~ 53.28% area (from TB 2.2)
# 87 done, ~ 53.18% Completed ~ 53.77% area (from TB 2.2)
# 112 done, ~ 53.64% Completed ~ 54.27% area (from TB 2.2)
# 77 done, ~ 54.09% Completed ~ 54.76% area (from TB 2.2)
# 86 done, ~ 54.55% Completed ~ 55.26% area (from TB 2.2)
# 82 done, ~ 55.00% Completed ~ 55.73% area (from TB 2.2)
# 68 done, ~ 55.45% Completed ~ 56.22% area (from TB 2.2)
# 70 done, ~ 55.91% Completed ~ 56.72% area (from TB 2.2)
# 81 done, ~ 56.36% Completed ~ 57.19% area (from TB 2.2)
# 85 done, ~ 56.82% Completed ~ 57.69% area (from TB 2.2)
# 73 done, ~ 57.27% Completed ~ 58.18% area (from TB 2.2)
# 69 done, ~ 57.73% Completed ~ 58.68% area (from TB 2.2)
# 67 done, ~ 58.18% Completed ~ 59.15% area (from TB 2.2)
# 212 done, ~ 58.64% Completed ~ 59.44% area (from TB 2.3)
# 78 done, ~ 59.09% Completed ~ 59.93% area (from TB 2.2)
# 64 done, ~ 59.55% Completed ~ 60.40% area (from TB 2.2)
# 201 done, ~ 60.00% Completed ~ 60.51% area (from TB 2.3)
# 93 done, ~ 60.45% Completed ~ 61.00% area (from TB 2.2)
# 207 done, ~ 60.91% Completed ~ 61.49% area (from TB 2.3)
```

```
# 95 done, ~ 61.36% Completed ~ 61.99% area (from TB 2.2)
# 90 done, ~ 61.82% Completed ~ 62.48% area (from TB 2.2)
# 71 done, ~ 62.27% Completed ~ 62.98% area (from TB 2.2)
# 202 done, ~ 62.73% Completed ~ 63.14% area (from TB 2.3)
# 96 done, ~ 63.18% Completed ~ 63.64% area (from TB 2.2)
# 204 done, ~ 63.64% Completed ~ 64.13% area (from TB 2.3)
# 80 done, ~ 64.09% Completed ~ 64.60% area (from TB 2.2)
# 84 done, ~ 64.55% Completed ~ 65.10% area (from TB 2.2)
# 89 done, ~ 65.00% Completed ~ 65.59% area (from TB 2.2)
# 203 done, ~ 65.45% Completed ~ 66.09% area (from TB 2.3)
# 72 done, ~ 65.91% Completed ~ 66.58% area (from TB 2.2)
# 199 done, ~ 66.36% Completed ~ 67.08% area (from TB 2.3)
# 200 done, ~ 66.82% Completed ~ 67.57% area (from TB 2.3)
# 92 done, ~ 67.27% Completed ~ 68.06% area (from TB 2.2)
# 79 done, ~ 67.73% Completed ~ 68.56% area (from TB 2.2)
# 208 done, ~ 68.18% Completed ~ 69.05% area (from TB 2.3)
# 83 done, ~ 68.64% Completed ~ 69.55% area (from TB 2.2)
# 195 done, ~ 69.09% Completed ~ 69.85% area (from TB 2.3)
# 74 done, ~ 69.55% Completed ~ 70.35% area (from TB 2.2)
# 76 done, ~ 70.00% Completed ~ 70.84% area (from TB 2.2)
# 192 done, ~ 70.45% Completed ~ 71.15% area (from TB 2.3)
# 198 done, ~ 70.91% Completed ~ 71.44% area (from TB 2.3)
# 66 done, ~ 71.36% Completed ~ 71.93% area (from TB 2.2)
# 65 done, ~ 71.82% Completed ~ 72.43% area (from TB 2.2)
# 213 done, ~ 72.27% Completed ~ 72.81% area (from TB 2.3)
# 174 done, ~ 72.73% Completed ~ 72.92% area (from TB 2.3)
# 171 done, ~ 73.18% Completed ~ 73.03% area (from TB 2.3)
# 191 done, ~ 73.64% Completed ~ 73.52% area (from TB 2.3)
# 178 done, ~ 74.09% Completed ~ 74.02% area (from TB 2.3)
# 190 done, ~ 74.55% Completed ~ 74.49% area (from TB 2.3)
# 168 done, ~ 75.00% Completed ~ 74.60% area (from TB 2.3)
# 75 done, ~ 75.45% Completed ~ 75.09% area (from TB 2.2)
# 189 done, ~ 75.91% Completed ~ 75.56% area (from TB 2.3)
# 165 done, ~ 76.36% Completed ~ 76.05% area (from TB 2.3)
# 166 done, ~ 76.82% Completed ~ 76.54% area (from TB 2.3)
# 167 done, ~ 77.27% Completed ~ 77.01% area (from TB 2.3)
# 187 done, ~ 77.73% Completed ~ 77.51% area (from TB 2.3)
# 180 done, ~ 78.18% Completed ~ 77.98% area (from TB 2.3)
# 177 done, ~ 78.64% Completed ~ 78.45% area (from TB 2.3)
# 164 done, ~ 79.09% Completed ~ 78.91% area (from TB 2.3)
# 186 done, ~ 79.55% Completed ~ 79.38% area (from TB 2.3)
# 216 done, ~ 80.00% Completed ~ 79.88% area (from TB 2.3)
# 183 done, ~ 80.45% Completed ~ 80.35% area (from TB 2.3)
# 194 done, ~ 80.91% Completed ~ 80.85% area (from TB 2.3)
# 197 done, ~ 81.36% Completed ~ 81.32% area (from TB 2.3)
# 193 done, ~ 81.82% Completed ~ 81.79% area (from TB 2.3)
# 156 done, ~ 82.27% Completed ~ 81.84% area (from TB 2.3)
# 181 done, ~ 82.73% Completed ~ 82.33% area (from TB 2.3)
```

```
 # 170 done, ~ 83.18% Completed ~ 82.80% area (from TB 2.3)
 # 206 done, ~ 83.64% Completed ~ 83.30% area (from TB 2.3)
 # 196 done, ~ 84.09% Completed ~ 83.74% area (from TB 2.3)
 # 184 done, ~ 84.55% Completed ~ 84.24% area (from TB 2.3)
 # 205 done, ~ 85.00% Completed ~ 84.71% area (from TB 2.3)
 # 163 done, ~ 85.45% Completed ~ 85.18% area (from TB 2.3)
 # 215 done, ~ 85.91% Completed ~ 85.67% area (from TB 2.3)
 # 176 done, ~ 86.36% Completed ~ 86.14% area (from TB 2.3)
 # 173 done, ~ 86.82% Completed ~ 86.61% area (from TB 2.3)
 # 179 done, ~ 87.27% Completed ~ 87.00% area (from TB 2.3)
 # 175 done, ~ 87.73% Completed ~ 87.37% area (from TB 2.3)
 # 172 done, ~ 88.18% Completed ~ 87.73% area (from TB 2.3)
 # 214 done, ~ 88.64% Completed ~ 88.23% area (from TB 2.3)
 # 169 done, ~ 89.09% Completed ~ 88.59% area (from TB 2.3)
 # 210 done, ~ 89.55% Completed ~ 89.09% area (from TB 2.3)
 # 211 done, ~ 90.00% Completed ~ 89.58% area (from TB 2.3)
 # 151 done, ~ 90.45% Completed ~ 89.86% area (from TB 2.3)
 # 147 done, ~ 90.91% Completed ~ 90.36% area (from TB 2.3)
 # 146 done, ~ 91.36% Completed ~ 90.86% area (from TB 2.3)
 # 209 done, ~ 91.82% Completed ~ 91.35% area (from TB 2.3)
 # 188 done, ~ 92.27% Completed ~ 91.85% area (from TB 2.3)
 # 159 done, ~ 92.73% Completed ~ 92.32% area (from TB 2.3)
 # 160 done, ~ 93.18% Completed ~ 92.79% area (from TB 2.3)
 # 157 done, ~ 93.64% Completed ~ 93.23% area (from TB 2.3)
 # 185 done, ~ 94.09% Completed ~ 93.73% area (from TB 2.3)
 # 158 done, ~ 94.55% Completed ~ 94.20% area (from TB 2.3)
 # 152 done, ~ 95.00% Completed ~ 94.57% area (from TB 2.3)
 # 161 done, ~ 95.45% Completed ~ 95.07% area (from TB 2.3)
 # 162 done, ~ 95.91% Completed ~ 95.56% area (from TB 2.3)
 # 182 done, ~ 96.36% Completed ~ 96.06% area (from TB 2.3)
 # 144 done, ~ 96.82% Completed ~ 96.53% area (from TB 2.3)
 # 145 done, ~ 97.27% Completed ~ 97.03% area (from TB 2.3)
 # 155 done, ~ 97.73% Completed ~ 97.52% area (from TB 2.3)
 # 153 done, ~ 98.18% Completed ~ 98.02% area (from TB 2.3)
 # 148 done, ~ 98.64% Completed ~ 98.51% area (from TB 2.3)
 # 154 done, ~ 99.09% Completed ~ 99.01% area (from TB 2.3)
 # 150 done, ~ 99.55% Completed ~ 99.51% area (from TB 2.3)
 # 149 done, ~ 100.00% Completed ~ 100.00% area (from TB 2.3)

Concurrent[2]: All data is present in the output for TB 2.

Pipeline flow complete for TB 2, Wed Feb 12 12:10:46 2014

 TB2 Times:User: 2.23 Sys: 1.05 Elapsed: 128.82 Memory: 379.800M

 Pipeline flow started for TB 3, Wed Feb 12 12:10:46 2014

 spatial context: 64 templates generated for stripe 3.1 of 3
```

```
 # 55 done, ~ 0.47% Completed ~ 0.17% area (from TB 3.1)
 # 56 done, ~ 0.93% Completed ~ 0.50% area (from TB 3.1)
 # 33 done, ~ 1.40% Completed ~ 0.93% area (from TB 3.1)
 # 44 done, ~ 1.86% Completed ~ 1.09% area (from TB 3.1)
 # 41 done, ~ 2.33% Completed ~ 1.37% area (from TB 3.1)
 # 32 done, ~ 2.80% Completed ~ 1.81% area (from TB 3.1)
 # 36 done, ~ 3.26% Completed ~ 2.25% area (from TB 3.1)
 # 63 done, ~ 3.73% Completed ~ 2.68% area (from TB 3.1)
 # 35 done, ~ 4.19% Completed ~ 3.12% area (from TB 3.1)
 # 54 done, ~ 4.66% Completed ~ 3.53% area (from TB 3.1)
 # 53 done, ~ 5.12% Completed ~ 3.96% area (from TB 3.1)
 # 45 done, ~ 5.59% Completed ~ 4.38% area (from TB 3.1)
 # 58 done, ~ 6.06% Completed ~ 4.82% area (from TB 3.1)
 # 51 done, ~ 6.52% Completed ~ 5.25% area (from TB 3.1)
 # 47 done, ~ 6.99% Completed ~ 5.69% area (from TB 3.1)
 # 48 done, ~ 7.45% Completed ~ 6.13% area (from TB 3.1)
 # 42 done, ~ 7.92% Completed ~ 6.57% area (from TB 3.1)
 # 50 done, ~ 8.39% Completed ~ 7.00% area (from TB 3.1)
 # 37 done, ~ 8.85% Completed ~ 7.42% area (from TB 3.1)
 # 43 done, ~ 9.32% Completed ~ 7.86% area (from TB 3.1)
 # 59 done, ~ 9.78% Completed ~ 8.30% area (from TB 3.1)
 # 30 done, ~ 10.25% Completed ~ 8.75% area (from TB 3.1)
 # 21 done, ~ 10.71% Completed ~ 9.19% area (from TB 3.1)
 # 20 done, ~ 11.18% Completed ~ 9.61% area (from TB 3.1)
 # 23 done, ~ 11.65% Completed ~ 10.05% area (from TB 3.1)
 spatial context: 80 templates generated for stripe 3.2 of 3
 # 9 done, ~ 11.85% Completed ~ 10.50% area (from TB 3.1)
 # 7 done, ~ 12.31% Completed ~ 10.92% area (from TB 3.1)
 # 8 done, ~ 12.77% Completed ~ 11.36% area (from TB 3.1)
     Times: User: 0.91 Sys: 0.67 Elapsed: 73.26 Memory: 371.322M
 # 6 done, ~ 13.22% Completed ~ 11.80% area (from TB 3.1)
 spatial context: 74 templates generated for stripe 3.3 of 3
 # 5 done, ~ 13.68% Completed ~ 12.25% area (from TB 3.1)
     Times: User: 0.10 Sys: 0.01 Elapsed: 0.06 Memory: 266.395M
Calculating context for holding cells.
 holder context: 10%
 holder context: 20%
 holder context: 30%
 holder context: 40%
 holder context: 50%
 holder context: 60%
 holder context: 70%
 holder context: 80%
 holder context: 90%
 holder context: Done Wed Feb 12 12:10:47 2014

 # 3 done, ~ 14.22% Completed ~ 12.98% area (from TB 3.1)
     Times: User: 0.01 Sys: 0.00 Elapsed: 0.00 Memory: 261.712M
```

```
 220 output templates (218 cluster templates) generated for TB3:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 85598.4280 square microns.
 Sum of all template bounding areas: 78760.9214 square microns.
 Sum of all ambit-biased template bounding areas: 87266.8274
square microns.

 Hierman BackEnd Times: User: 2.91 Sys: 1.44 Elapsed: 129.61
Memory: 379.833M
 # 4 done, ~ 14.55% Completed ~ 14.60% area (from TB 3.1)
Client: executing mktop in a separate thread...
 # 2 done, ~ 15.00% Completed ~ 15.07% area (from TB 3.1)
 # 18 done, ~ 15.45% Completed ~ 15.57% area (from TB 3.1)
 # 57 done, ~ 15.91% Completed ~ 16.04% area (from TB 3.1)
 # 29 done, ~ 16.36% Completed ~ 16.51% area (from TB 3.1)
 # 24 done, ~ 16.82% Completed ~ 16.99% area (from TB 3.1)
 # 25 done, ~ 17.27% Completed ~ 17.48% area (from TB 3.1)
 # 22 done, ~ 17.73% Completed ~ 17.97% area (from TB 3.1)
 # 34 done, ~ 18.18% Completed ~ 18.46% area (from TB 3.1)
 # 28 done, ~ 18.64% Completed ~ 18.95% area (from TB 3.1)
 # 31 done, ~ 19.09% Completed ~ 19.45% area (from TB 3.1)
 # 52 done, ~ 19.55% Completed ~ 19.92% area (from TB 3.1)
 # 17 done, ~ 20.00% Completed ~ 20.41% area (from TB 3.1)
 # 49 done, ~ 20.45% Completed ~ 20.90% area (from TB 3.1)
 # 46 done, ~ 20.91% Completed ~ 21.38% area (from TB 3.1)
 # 19 done, ~ 21.36% Completed ~ 21.87% area (from TB 3.1)
 # 27 done, ~ 21.82% Completed ~ 22.36% area (from TB 3.1)
 # 14 done, ~ 22.27% Completed ~ 22.85% area (from TB 3.1)
 # 26 done, ~ 22.73% Completed ~ 23.35% area (from TB 3.1)
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 22.73% Completed ~ 23.35% area (from TB 3)
Client: executing mktop in a separate thread...
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # 0 done, ~ 23.18% Completed ~ 23.82% area (from TB 3.1)
 # mktop done, ~ 23.18% Completed ~ 23.82% area (from TB 3)
 # 12 done, ~ 23.64% Completed ~ 24.29% area (from TB 3.1)
 # 1 done, ~ 24.09% Completed ~ 24.77% area (from TB 3.1)
 # 11 done, ~ 24.55% Completed ~ 25.26% area (from TB 3.1)
Client: executing mktop in a separate thread...
 # 16 done, ~ 25.00% Completed ~ 25.74% area (from TB 3.1)
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
```

```
# 13 done, ~ 25.45% Completed ~ 26.23% area (from TB 3.1)
# mktop done, ~ 26.36% Completed ~ 26.23% area (from TB 3)
# 10 done, ~ 26.82% Completed ~ 26.72% area (from TB 3.1)
# 15 done, ~ 27.27% Completed ~ 27.21% area (from TB 3.1)
# 40 done, ~ 27.73% Completed ~ 27.51% area (from TB 3.1)
# 39 done, ~ 28.18% Completed ~ 28.00% area (from TB 3.1)
# 60 done, ~ 28.64% Completed ~ 28.48% area (from TB 3.1)
# 61 done, ~ 29.09% Completed ~ 28.97% area (from TB 3.1)
# 62 done, ~ 29.55% Completed ~ 29.46% area (from TB 3.1)
# 38 done, ~ 30.00% Completed ~ 29.96% area (from TB 3.1)
# 116 done, ~ 30.45% Completed ~ 30.21% area (from TB 3.2)
# 113 done, ~ 30.91% Completed ~ 30.47% area (from TB 3.2)
# 111 done, ~ 31.36% Completed ~ 30.96% area (from TB 3.2)
# 110 done, ~ 31.82% Completed ~ 31.22% area (from TB 3.2)
# 115 done, ~ 32.27% Completed ~ 31.71% area (from TB 3.2)
# 114 done, ~ 32.73% Completed ~ 32.20% area (from TB 3.2)
# 112 done, ~ 33.18% Completed ~ 32.69% area (from TB 3.2)
# 107 done, ~ 33.64% Completed ~ 33.19% area (from TB 3.2)
# 109 done, ~ 34.09% Completed ~ 33.68% area (from TB 3.2)
# 108 done, ~ 34.55% Completed ~ 34.17% area (from TB 3.2)
# 106 done, ~ 35.00% Completed ~ 34.66% area (from TB 3.2)
# 105 done, ~ 35.45% Completed ~ 35.16% area (from TB 3.2)
# 104 done, ~ 35.91% Completed ~ 35.65% area (from TB 3.2)
# 100 done, ~ 36.36% Completed ~ 36.14% area (from TB 3.2)
# 103 done, ~ 36.82% Completed ~ 36.63% area (from TB 3.2)
# 98 done, ~ 37.27% Completed ~ 37.10% area (from TB 3.2)
# 99 done, ~ 37.73% Completed ~ 37.59% area (from TB 3.2)
# 102 done, ~ 38.18% Completed ~ 38.08% area (from TB 3.2)
# 101 done, ~ 38.64% Completed ~ 38.58% area (from TB 3.2)
# 94 done, ~ 39.09% Completed ~ 39.07% area (from TB 3.2)
# 95 done, ~ 39.55% Completed ~ 39.56% area (from TB 3.2)
# 97 done, ~ 40.00% Completed ~ 40.05% area (from TB 3.2)
# 93 done, ~ 40.45% Completed ~ 40.54% area (from TB 3.2)
# 92 done, ~ 40.91% Completed ~ 41.03% area (from TB 3.2)
# 96 done, ~ 41.36% Completed ~ 41.52% area (from TB 3.2)
# 91 done, ~ 41.82% Completed ~ 42.01% area (from TB 3.2)
# 90 done, ~ 42.27% Completed ~ 42.51% area (from TB 3.2)
# 88 done, ~ 42.73% Completed ~ 43.00% area (from TB 3.2)
# 87 done, ~ 43.18% Completed ~ 43.49% area (from TB 3.2)
# 86 done, ~ 43.64% Completed ~ 43.98% area (from TB 3.2)
# 89 done, ~ 44.09% Completed ~ 44.47% area (from TB 3.2)
# 85 done, ~ 44.55% Completed ~ 44.96% area (from TB 3.2)
# 84 done, ~ 45.00% Completed ~ 45.45% area (from TB 3.2)
# 81 done, ~ 45.45% Completed ~ 45.93% area (from TB 3.2)
# 82 done, ~ 45.91% Completed ~ 46.40% area (from TB 3.2)
# 83 done, ~ 46.36% Completed ~ 46.90% area (from TB 3.2)
# 80 done, ~ 46.82% Completed ~ 47.37% area (from TB 3.2)
# 79 done, ~ 47.27% Completed ~ 47.86% area (from TB 3.2)
```

```
# 77 done, ~ 47.73% Completed ~ 48.35% area (from TB 3.2)
# 78 done, ~ 48.18% Completed ~ 48.85% area (from TB 3.2)
# 75 done, ~ 48.64% Completed ~ 49.34% area (from TB 3.2)
# 76 done, ~ 49.09% Completed ~ 49.83% area (from TB 3.2)
# 74 done, ~ 49.55% Completed ~ 50.32% area (from TB 3.2)
# 73 done, ~ 50.00% Completed ~ 50.81% area (from TB 3.2)
# 68 done, ~ 50.45% Completed ~ 51.30% area (from TB 3.2)
# 72 done, ~ 50.91% Completed ~ 51.79% area (from TB 3.2)
# 71 done, ~ 51.36% Completed ~ 52.29% area (from TB 3.2)
# 69 done, ~ 51.82% Completed ~ 52.78% area (from TB 3.2)
# 70 done, ~ 52.27% Completed ~ 53.27% area (from TB 3.2)
# 67 done, ~ 52.73% Completed ~ 53.74% area (from TB 3.2)
# 64 done, ~ 53.18% Completed ~ 54.22% area (from TB 3.2)
# 66 done, ~ 53.64% Completed ~ 54.71% area (from TB 3.2)
# 65 done, ~ 54.09% Completed ~ 55.20% area (from TB 3.2)
# 217 done, ~ 54.55% Completed ~ 55.25% area (from TB 3.3)
# 216 done, ~ 55.00% Completed ~ 55.74% area (from TB 3.3)
# 213 done, ~ 55.45% Completed ~ 56.13% area (from TB 3.3)
# 215 done, ~ 55.91% Completed ~ 56.62% area (from TB 3.3)
# 214 done, ~ 56.36% Completed ~ 57.12% area (from TB 3.3)
# 211 done, ~ 56.82% Completed ~ 57.61% area (from TB 3.3)
# 212 done, ~ 57.27% Completed ~ 57.90% area (from TB 3.3)
# 210 done, ~ 57.73% Completed ~ 58.39% area (from TB 3.3)
# 209 done, ~ 58.18% Completed ~ 58.88% area (from TB 3.3)
# 207 done, ~ 58.64% Completed ~ 59.38% area (from TB 3.3)
# 208 done, ~ 59.09% Completed ~ 59.87% area (from TB 3.3)
# 204 done, ~ 59.55% Completed ~ 60.36% area (from TB 3.3)
# 206 done, ~ 60.00% Completed ~ 60.85% area (from TB 3.3)
# 205 done, ~ 60.45% Completed ~ 61.32% area (from TB 3.3)
# 202 done, ~ 60.91% Completed ~ 61.50% area (from TB 3.3)
# 201 done, ~ 61.36% Completed ~ 61.61% area (from TB 3.3)
# 203 done, ~ 61.82% Completed ~ 62.10% area (from TB 3.3)
# 200 done, ~ 62.27% Completed ~ 62.59% area (from TB 3.3)
# 199 done, ~ 62.73% Completed ~ 63.08% area (from TB 3.3)
# 198 done, ~ 63.18% Completed ~ 63.38% area (from TB 3.3)
# 195 done, ~ 63.64% Completed ~ 63.69% area (from TB 3.3)
# 197 done, ~ 64.09% Completed ~ 64.17% area (from TB 3.3)
# 196 done, ~ 64.55% Completed ~ 64.62% area (from TB 3.3)
# 194 done, ~ 65.00% Completed ~ 65.12% area (from TB 3.3)
# 192 done, ~ 65.45% Completed ~ 65.43% area (from TB 3.3)
# 193 done, ~ 65.91% Completed ~ 65.90% area (from TB 3.3)
# 191 done, ~ 66.36% Completed ~ 66.39% area (from TB 3.3)
# 190 done, ~ 66.82% Completed ~ 66.87% area (from TB 3.3)
# 189 done, ~ 67.27% Completed ~ 67.34% area (from TB 3.3)
# 188 done, ~ 67.73% Completed ~ 67.83% area (from TB 3.3)
# 187 done, ~ 68.18% Completed ~ 68.32% area (from TB 3.3)
# 186 done, ~ 68.64% Completed ~ 68.80% area (from TB 3.3)
# 183 done, ~ 69.09% Completed ~ 69.27% area (from TB 3.3)
```

```
 # 185 done, ~ 69.55% Completed ~ 69.76% area (from TB 3.3)
 # 184 done, ~ 70.00% Completed ~ 70.26% area (from TB 3.3)
 # 181 done, ~ 70.45% Completed ~ 70.75% area (from TB 3.3)
 # 180 done, ~ 70.91% Completed ~ 71.22% area (from TB 3.3)
 # 182 done, ~ 71.36% Completed ~ 71.71% area (from TB 3.3)
 # 125 done, ~ 71.82% Completed ~ 72.21% area (from TB 3.2)
 # 178 done, ~ 72.27% Completed ~ 72.70% area (from TB 3.3)
 # 177 done, ~ 72.73% Completed ~ 73.17% area (from TB 3.3)
 # 179 done, ~ 73.18% Completed ~ 73.56% area (from TB 3.3)
 # 174 done, ~ 73.64% Completed ~ 73.67% area (from TB 3.3)
 # 176 done, ~ 74.09% Completed ~ 74.15% area (from TB 3.3)
 # 173 done, ~ 74.55% Completed ~ 74.62% area (from TB 3.3)
 # 171 done, ~ 75.00% Completed ~ 74.74% area (from TB 3.3)
 # 175 done, ~ 75.45% Completed ~ 75.11% area (from TB 3.3)
 # 170 done, ~ 75.91% Completed ~ 75.58% area (from TB 3.3)
 # 169 done, ~ 76.36% Completed ~ 75.95% area (from TB 3.3)
 # 168 done, ~ 76.82% Completed ~ 76.07% area (from TB 3.3)
 # 172 done, ~ 77.27% Completed ~ 76.44% area (from TB 3.3)
 # 165 done, ~ 77.73% Completed ~ 76.93% area (from TB 3.3)
 # 167 done, ~ 78.18% Completed ~ 77.40% area (from TB 3.3)
 # 166 done, ~ 78.64% Completed ~ 77.89% area (from TB 3.3)
 # 164 done, ~ 79.09% Completed ~ 78.37% area (from TB 3.3)
 # 163 done, ~ 79.55% Completed ~ 78.84% area (from TB 3.3)
 # 162 done, ~ 80.00% Completed ~ 79.33% area (from TB 3.3)
 # 160 done, ~ 80.45% Completed ~ 79.81% area (from TB 3.3)
 # 159 done, ~ 80.91% Completed ~ 80.28% area (from TB 3.3)
 # 161 done, ~ 81.36% Completed ~ 80.78% area (from TB 3.3)
 # 158 done, ~ 81.82% Completed ~ 81.25% area (from TB 3.3)
 # 157 done, ~ 82.27% Completed ~ 81.71% area (from TB 3.3)
 # 156 done, ~ 82.73% Completed ~ 81.77% area (from TB 3.3)
 # 154 done, ~ 83.18% Completed ~ 82.26% area (from TB 3.3)
 # 155 done, ~ 83.64% Completed ~ 82.75% area (from TB 3.3)
 # 153 done, ~ 84.09% Completed ~ 83.24% area (from TB 3.3)
 # 152 done, ~ 84.55% Completed ~ 83.62% area (from TB 3.3)
 # 151 done, ~ 85.00% Completed ~ 83.91% area (from TB 3.3)
 # 149 done, ~ 85.45% Completed ~ 84.40% area (from TB 3.3)
 # 150 done, ~ 85.91% Completed ~ 84.89% area (from TB 3.3)
 # 147 done, ~ 86.36% Completed ~ 85.38% area (from TB 3.3)
 # 146 done, ~ 86.82% Completed ~ 85.88% area (from TB 3.3)
 # 148 done, ~ 87.27% Completed ~ 86.37% area (from TB 3.3)
 server 2: ltgpe-05 exited: MSG_NUM 96 Total Times:User: 64.51
Sys: 0.13 Elapsed: 84.05 Memory: 83.870M
 # 145 done, ~ 87.73% Completed ~ 86.86% area (from TB 3.3)
 # 144 done, ~ 88.18% Completed ~ 87.33% area (from TB 3.3)
 server 3: ltgpe-05 exited: MSG_NUM 103 Total Times:User: 71.00
Sys: 0.12 Elapsed: 84.08 Memory: 87.410M
 server 1: ltgpe-03 exited: MSG_NUM 166 Total Times:User: 106.87
Sys: 0.14 Elapsed: 125.74 Memory: 93.616M
 server 4: ltgpe-05 exited: MSG_NUM 100 Total Times:User: 69.58
```

```
Sys: 0.12 Elapsed: 84.09 Memory: 84.481M
 server 6: ltgpe-05 exited: MSG_NUM 92 Total Times:User: 64.26
Sys: 0.08 Elapsed: 84.09 Memory: 83.393M
 server 5: ltgpe-05 exited: MSG_NUM 96 Total Times:User: 66.79
Sys: 0.13 Elapsed: 84.09 Memory: 91.276M
 # 131 done, ~ 88.64% Completed ~ 87.82% area (from TB 3.2)
 # 136 done, ~ 89.09% Completed ~ 88.32% area (from TB 3.2)
 server 7: ltgpe-05 exited: MSG_NUM 98 Total Times:User: 77.06
Sys: 0.11 Elapsed: 87.45 Memory: 82.753M
 server 16: ltgpe-73 exited: MSG_NUM 62 Total Times:User: 69.27
Sys: 0.08 Elapsed: 86.63 Memory: 80.194M
 # 134 done, ~ 89.55% Completed ~ 88.81% area (from TB 3.2)
 # 143 done, ~ 90.00% Completed ~ 89.28% area (from TB 3.2)
 # 133 done, ~ 90.45% Completed ~ 89.77% area (from TB 3.2)
 # 127 done, ~ 90.91% Completed ~ 90.27% area (from TB 3.2)
 # 126 done, ~ 91.36% Completed ~ 90.76% area (from TB 3.2)
 # 139 done, ~ 91.82% Completed ~ 91.25% area (from TB 3.2)
 # 137 done, ~ 92.27% Completed ~ 91.74% area (from TB 3.2)
 # 138 done, ~ 92.73% Completed ~ 92.23% area (from TB 3.2)
 server 25: ltgpe-master2 exited: MSG_NUM 52 Total Times:User:
75.19 Sys: 0.12 Elapsed: 84.45 Memory: 77.562M
 server 18: ltgpe-73 exited: MSG_NUM 56 Total Times:User: 67.30
Sys: 0.08 Elapsed: 86.58 Memory: 83.921M
 server 21: ltgpe-60 exited: MSG_NUM 54 Total Times:User: 73.96
Sys: 0.10 Elapsed: 85.65 Memory: 77.668M
 server 22: ltgpe-master2 exited: MSG_NUM 46 Total Times:User:
57.68 Sys: 0.13 Elapsed: 84.76 Memory: 76.345M
 server 13: ltgpe-73 exited: MSG_NUM 62 Total Times:User: 66.22
Sys: 0.08 Elapsed: 86.72 Memory: 77.703M
 server 15: ltgpe-73 exited: MSG_NUM 58 Total Times:User: 60.12
Sys: 0.25 Elapsed: 86.67 Memory: 81.540M
 server 14: ltgpe-73 exited: MSG_NUM 62 Total Times:User: 59.91
Sys: 0.11 Elapsed: 86.71 Memory: 88.968M
 # 132 done, ~ 93.18% Completed ~ 92.72% area (from TB 3.2)
 # 130 done, ~ 93.64% Completed ~ 93.22% area (from TB 3.2)
 # 129 done, ~ 94.09% Completed ~ 93.71% area (from TB 3.2)
 server 32: ltgpe-master2 exited: MSG_NUM 48 Total Times:User:
72.85 Sys: 0.12 Elapsed: 82.67 Memory: 74.287M
 # 135 done, ~ 94.55% Completed ~ 94.20% area (from TB 3.2)
 # 124 done, ~ 95.00% Completed ~ 94.69% area (from TB 3.2)
 # 122 done, ~ 95.45% Completed ~ 95.16% area (from TB 3.2)
 server 23: ltgpe-master1 exited: MSG_NUM 56 Total Times:User:
61.47 Sys: 0.15 Elapsed: 84.56 Memory: 79.205M
 server 29: ltgpe-master1 exited: MSG_NUM 48 Total Times:User:
59.29 Sys: 0.12 Elapsed: 83.12 Memory: 73.945M
 # 123 done, ~ 95.91% Completed ~ 95.65% area (from TB 3.2)
 server 28: ltgpe-master1 exited: MSG_NUM 36 Total Times:User:
66.24 Sys: 0.12 Elapsed: 83.12 Memory: 73.611M
 # 142 done, ~ 96.36% Completed ~ 96.14% area (from TB 3.2)
 # 128 done, ~ 96.82% Completed ~ 96.62% area (from TB 3.2)
```

```
 server 12: ltgpe-73 exited: MSG_NUM 64 Total Times:User: 60.93
Sys: 0.09 Elapsed: 86.75 Memory: 85.265M
 server 17: ltgpe-73 exited: MSG_NUM 60 Total Times:User: 65.10
Sys: 0.07 Elapsed: 86.61 Memory: 90.015M
 server 19: ltgpe-56 exited: MSG_NUM 62 Total Times:User: 65.55
Sys: 0.09 Elapsed: 86.14 Memory: 75.430M
 server 20: ltgpe-56 exited: MSG_NUM 62 Total Times:User: 67.14
Sys: 0.10 Elapsed: 86.14 Memory: 74.567M
 # 141 done, ~ 97.27% Completed ~ 97.11% area (from TB 3.2)
 # 140 done, ~ 97.73% Completed ~ 97.60% area (from TB 3.2)
 server 30: ltgpe-master2 exited: MSG_NUM 46 Total Times:User:
63.43 Sys: 0.10 Elapsed: 82.86 Memory: 73.754M
 server 10: ltgpe-73 exited: MSG_NUM 56 Total Times:User: 61.22
Sys: 0.10 Elapsed: 86.80 Memory: 88.805M
 server 11: ltgpe-73 exited: MSG_NUM 62 Total Times:User: 73.86
Sys: 0.24 Elapsed: 86.77 Memory: 81.832M
 server 9: rutro-04 exited: MSG_NUM 62 Total Times:User: 62.52
Sys: 0.31 Elapsed: 87.25 Memory: 76.538M
 # 119 done, ~ 98.18% Completed ~ 98.07% area (from TB 3.2)
 server 24: ltgpe-master1 exited: MSG_NUM 50 Total Times:User:
62.93 Sys: 0.12 Elapsed: 84.55 Memory: 74.715M
 # 120 done, ~ 98.64% Completed ~ 98.56% area (from TB 3.2)
 # 117 done, ~ 99.09% Completed ~ 99.03% area (from TB 3.2)
 server 27: ltgpe-master1 exited: MSG_NUM 48 Total Times:User:
72.51 Sys: 0.13 Elapsed: 83.27 Memory: 74.026M
 # 121 done, ~ 99.55% Completed ~ 99.52% area (from TB 3.2)
 server 26: ltgpe-master1 exited: MSG_NUM 50 Total Times:User:
67.58 Sys: 0.12 Elapsed: 83.50 Memory: 67.241M
 server 31: ltgpe-master2 exited: MSG_NUM 40 Total Times:User:
55.81 Sys: 0.11 Elapsed: 82.85 Memory: 73.628M
 # 118 done, ~ 100.00% Completed ~ 100.00% area (from TB 3.2)

 Final output file ./gds/TB1_out.gds is now complete, Wed Feb 12
12:10:53 2014


Prepare the holder cells for bound box data calculation for TB3.
 prepare holder cells: 10%
 prepare holder cells: 20%
 prepare holder cells: 30%
 prepare holder cells: 40%
 prepare holder cells: 50%
 prepare holder cells: 60%
 prepare holder cells: 70%
 prepare holder cells: 80%
 prepare holder cells: 90%
 prepare holder cells: Done Wed Feb 12 12:10:53 2014
```

```
Collecting cells bound box data for TB3.
 collect cell data: 10%
 collect cell data: 20%
 collect cell data: 30%
 collect cell data: 40%
 collect cell data: 50%
 collect cell data: 60%
 collect cell data: 70%
 collect cell data: 80%
 collect cell data: Done Wed Feb 12 12:10:53 2014


Finalizing OASIS output file ./gds/TB2_out.oas for TB3.
 finish output file: 10%
 finish output file: 20%
 finish output file: 30%
 finish output file: 40%
 finish output file: 50%
 finish output file: 60%
 finish output file: 70%
 finish output file: 80%
 finish output file: Done Wed Feb 12 12:10:53 2014


 Final output file ./gds/TB2_out.oas is now complete, Wed Feb 12
12:10:53 2014


 Final output file ./gds/MOF_BL_pcx2_out.gds is now complete, Wed
Feb 12 12:10:53 2014


Concurrent[3]: All data is present in the output for TB 3.
server 8: rutro-04 exited: MSG_NUM 60 Total Times:User: 63.69
Sys: 0.38 Elapsed: 87.47 Memory: 85.978M
 server 33: ltgpe-master2 exited: MSG_NUM 44 Total Times:User:
65.72 Sys: 0.13 Elapsed: 82.89 Memory: 75.764M

 Pipeline flow complete for TB 3, Wed Feb 12 12:10:53 2014

Template Generation Summary
 220 output templates (218 cluster templates) generated for TB1:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 84966.3180 square microns.
 Sum of all template bounding areas: 79470.0657 square microns.
 Sum of all ambit-biased template bounding areas: 93312.0891
square microns.
```

```
 220 output templates (218 cluster templates) generated for TB2:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 85200.1608 square microns.
 Sum of all template bounding areas: 78141.6062 square microns.
 Sum of all ambit-biased template bounding areas: 99135.4797
square microns.

 220 output templates (218 cluster templates) generated for TB3:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 85598.4280 square microns.
 Sum of all template bounding areas: 78760.9214 square microns.
 Sum of all ambit-biased template bounding areas: 87266.8274
square microns.


 Cleaning up intermediate recipe files at project finish
 rm -rf MOF_NONE_0210_TB1.pjx 2> /dev/null

 Cleaning up intermediate recipe files at project finish
 rm -rf MOF_NONE_0210_TB2.pjx 2> /dev/null

 Cleaning up intermediate recipe files at project finish
 rm -rf MOF_NONE_0210_TB3.pjx 2> /dev/null

 Cleaning up fragment directories at project finish
 rm -rf ./gds/TB1_out.gds.dir/ 2> /dev/null

 Pipeline flow complete for all template blocks, Wed Feb 12
12:10:53 2014

 TB3 Times:User: 4.63 Sys: 1.76 Elapsed: 135.81 Memory: 379.833M

Checking in PROTEUS_OPC...

Checking in PA...

 Server Processing Time Summary
 TB1 Elapsed: 0:00:47 [12:08:45-12:09:32]
 TB2 Elapsed: 0:01:12 [12:09:33-12:10:46]
 TB3 Elapsed: 0:00:05 [12:10:47-12:10:52]

 Total Times:User: 12.43 Sys: 2.79 Elapsed: 145.26 Memory: 379.833M
```

Log File Example 3

The following is an example of a log file from a proteus run with a `PROTEUS_JOB_FLOW` recipe with `PIPELINE_STRATEGY FRONTLOAD`.

```
qrsh -P bhigh -V -cwd -now no -l model=EMT2700|EMT3000 -l
os_version=WS5.0 proteus -s snps:32 MOF_NONE_0210.pjx
/remote/ms_integ3_us03/SCM/DailyBuildReleases/proteus_G-
2012.09-8_12Feb14/amd64/bin/proteus Release G-2012.09-8 Revision
Proteus_G-2012.09-8_12Feb14-2919019 (64f/64m LINUX_X86_64).
host: machine1

 Proteus (TM) / PROTEUS (TM)
Version G-2012.09-8

*** Copyright (C) 1995 - 2014 Synopsys, Inc. ***
*** This software and the associated documentation are ***
*** confidential and proprietary to Synopsys, Inc. ***
*** Your use or disclosure of this software is subject to ***
*** the terms and conditions of a written license agreement ***
*** between you, or your company, and Synopsys, Inc. ***
*** ***

Testing for license PROTEUS_OPC...

Checking out PROTEUS_OPC...
License PROTEUS_OPC checked out.

Testing for license PA...

Checking out PA...
License PA checked out.
hierman -fe MOF_NONE_0210.pjx

 Hierarchy management started for job HIERMAN; Wed Feb 12 11:51:50
2014
Initialization completed.
     Times: User: 1.03 Sys: 0.01 Elapsed: 1.07 Memory: 17.543M

Reading input file /remote/ltg_pe1_us03/usr/large_2/
everest_0130_00.oas
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................

Scanning for Topcell(s)
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................
 Topcell: everest_gem
```

```
 native hierarchy: 1 cells; 0 refs
     Times: User: 0.09 Sys: 0.00 Elapsed: 0.09 Memory: 19.499M
Separating graphics from SREFs & AREFs.
 1 holder cell added
     Times: User: 0.00 Sys: 0.01 Elapsed: 0.00 Memory: 19.466M
Calculating clustering statistics.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.722M
Cleaning 1D AREF transforms.
Choosing Smart Block Compression cells.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.05 Sys: 0.00 Elapsed: 0.04 Memory: 19.501M
Create OASIS decompression buffer file.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.17 Sys: 0.00 Elapsed: 0.24 Memory: 19.732M
Partitioning large graphic cells (multi-threaded).
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
 1 native cell divided into 217 smaller graphic cells (20000 x
20000)
     Times: User: 1.85 Sys: 0.02 Elapsed: 1.87 Memory: 29.707M
Subdividing large AREFs.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.534M
Constructing framework for revised hierarchy.
 revised hierarchy: 219 cells; 218 refs
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.01 Memory: 19.538M
Generating instances for processing.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
 219 instances (217 cluster instances) identified.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.981M
Generating spatial bins for TopCell.
 Bin Count : 16 x 14 (horizontal x vertical)
 Bin Dimensions : (19590,19684) (x,y)
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.818M
Generating spatial bin cells.
 revised hierarchy: 444 cells; 1244 refs
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.864M
Generating flat bin instances.
 226 instances (224 cluster instances) identified.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.865M

Writing partial hierarchy results.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.665M
     Total Times: User: 3.19 Sys: 0.04 Elapsed: 3.37 Memory:
```

```
29.707M

 Hierarchy management complete for job HIERMAN; Wed Feb 12 11:51:53
2014
 Hierman FrontEnd Times:User: 7.93 Sys: 0.94 Elapsed: 9.27 Memory:
17.630M

 Cleaning up double patterning files from a previous run
 rm -rf ./temp/HIERMAN_TB1.DPT 2> /dev/null

 Cleaning up TB 1 fragment files and outputs from a previous run
 rm -rf ./gds/TB1_out.gds.dir/TB1/ 2> /dev/null
 rm -f HIERMAN_TB1_out.oas 2> /dev/null

 mkdir -p ./gds/TB1_out.gds.dir/TB1/ 2> /dev/null
 mkdir ./gds/TB1_out.gds.dir/TB1/TINF 2> /dev/null
 Cleaning up double patterning files from a previous run
 rm -rf ./temp/HIERMAN_TB2.DPT 2> /dev/null

 Cleaning up TB 2 fragment files and outputs from a previous run
 rm -rf ./gds/TB1_out.gds.dir/TB2/ 2> /dev/null
 rm -f HIERMAN_TB2_out.oas 2> /dev/null

 mkdir -p ./gds/TB1_out.gds.dir/TB2/ 2> /dev/null
 mkdir ./gds/TB1_out.gds.dir/TB2/TINF 2> /dev/null
 Cleaning up double patterning files from a previous run
 rm -rf ./temp/HIERMAN_TB3.DPT 2> /dev/null

 Cleaning up TB 3 fragment files and outputs from a previous run
 rm -f ./gds/TB1_out.gds.dir/* 2> /dev/null
 rm -f ./gds/TB1_out.gds 2> /dev/null
 rm -f ./gds/TB2_out.oas 2> /dev/null
 rm -f ./gds/MOF_BL_pcx2_out.gds 2> /dev/null
 rm -f ./gds/TB1_out.gds_saved 2> /dev/null
 rm -f ./gds/TB1_out.gds_tmp 2> /dev/null

 mkdir -p ./gds/TB1_out.gds.dir/ 2> /dev/null
 mkdir ./gds/TB1_out.gds.dir/TINF 2> /dev/null
 ./gds/TB1_out.gds output file created, Wed Feb 12 11:51:53 2014
 ./gds/TB2_out.oas output file created, Wed Feb 12 11:51:53 2014
 ./gds/MOF_BL_pcx2_out.gds output file created, Wed Feb 12
11:51:53 2014

 Pipeline flow started for TB 1, Wed Feb 12 11:51:53 2014

Generating spatial templates for TB 1.
 spatial context: 64 templates generated for stripe 1.1 of 3
trying to open port:2346
Generating spatial templates for TB 2.
```

```
remote_server snps:32 2346 3 0 1
running remote_server for SGE access, in
/remote/us03home4/usr/bin/gridForProteus/remote_server
 spatial context: 80 templates generated for stripe 1.2 of 3
remote_server ltgpe-03 2346 3 1 1
 spatial context: 74 templates generated for stripe 1.3 of 3
     Times: User: 0.80 Sys: 0.29 Elapsed: 1.08 Memory: 279.543M
Calculating context for holding cells.
 holder context: 10%
 holder context: 20%
 holder context: 30%
 holder context: 40%
 holder context: 50%
 holder context: 60%
 holder context: 70%
 holder context: 80%
 holder context: 90%
 holder context: Done Wed Feb 12 11:51:54 2014

     Times: User: 0.00 Sys: 0.01 Elapsed: 0.01 Memory: 253.192M
 220 output templates (218 cluster templates) generated for TB1:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 84966.3180 square microns.
 Sum of all template bounding areas: 79470.0657 square microns.
 Sum of all ambit-biased template bounding areas: 93312.0891
square microns.

 Hierman BackEnd Times: User: 0.78 Sys: 0.13 Elapsed: 0.88 Memory:
279.543M
sh -c "{ { dpserver -p2346 -cltgpe-03 -V3 >>logfile.txt.ltgpe-
03.21050; } 3>&1 1>&2 2>&3 |tee logfile.txt.ltgpe-03.21050; }
3>&1 1>&2 2>&3"
Client: executing mktop in a separate thread...
qsub -P iheavy -V -cwd -N dpsvr.20952 -j y -o ./logfiles.20952 -
t 1-32 .dpserver.20952
Requesting 32 dpservers on snps...
Grid job number: 472478
dpserver log files are in logfiles.20952
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 0.91% Completed ~ 0.00% area (from TB 1)
Client: 1:1:INIT :N_TMPL=220 SOCK_CNT=1 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 63 done, ~ 1.36% Completed ~ 0.46% area (from TB 1.1)
 # 62 done, ~ 1.82% Completed ~ 0.92% area (from TB 1.1)
 # 61 done, ~ 2.27% Completed ~ 1.37% area (from TB 1.1)
```

```
 # 60 done, ~ 2.73% Completed ~ 1.80% area (from TB 1.1)
 # 59 done, ~ 3.18% Completed ~ 2.29% area (from TB 1.1)
 # 58 done, ~ 3.64% Completed ~ 2.78% area (from TB 1.1)
 # 57 done, ~ 4.09% Completed ~ 3.27% area (from TB 1.1)
 # 56 done, ~ 4.55% Completed ~ 3.73% area (from TB 1.1)
 # 55 done, ~ 5.00% Completed ~ 4.22% area (from TB 1.1)
 # 54 done, ~ 5.45% Completed ~ 4.71% area (from TB 1.1)
 # 53 done, ~ 5.91% Completed ~ 5.20% area (from TB 1.1)
 # 52 done, ~ 6.36% Completed ~ 5.66% area (from TB 1.1)
 # 51 done, ~ 6.82% Completed ~ 6.15% area (from TB 1.1)
 # 50 done, ~ 7.27% Completed ~ 6.64% area (from TB 1.1)
 # 49 done, ~ 7.73% Completed ~ 7.13% area (from TB 1.1)
 # 48 done, ~ 8.18% Completed ~ 7.59% area (from TB 1.1)
 # 47 done, ~ 8.64% Completed ~ 8.08% area (from TB 1.1)
 # 46 done, ~ 9.09% Completed ~ 8.57% area (from TB 1.1)
 # 45 done, ~ 9.55% Completed ~ 9.06% area (from TB 1.1)
 # 44 done, ~ 10.00% Completed ~ 9.52% area (from TB 1.1)
 # 43 done, ~ 10.45% Completed ~ 10.01% area (from TB 1.1)
 # 42 done, ~ 10.91% Completed ~ 10.50% area (from TB 1.1)
 # 41 done, ~ 11.36% Completed ~ 10.99% area (from TB 1.1)
 # 40 done, ~ 11.82% Completed ~ 11.44% area (from TB 1.1)
 # 39 done, ~ 12.27% Completed ~ 11.93% area (from TB 1.1)
 # 38 done, ~ 12.73% Completed ~ 12.42% area (from TB 1.1)
 # 37 done, ~ 13.18% Completed ~ 12.91% area (from TB 1.1)
 # 36 done, ~ 13.64% Completed ~ 13.37% area (from TB 1.1)
 # 35 done, ~ 14.09% Completed ~ 13.86% area (from TB 1.1)
 # 34 done, ~ 14.55% Completed ~ 14.35% area (from TB 1.1)
 # 33 done, ~ 15.00% Completed ~ 14.84% area (from TB 1.1)
 # 32 done, ~ 15.45% Completed ~ 15.30% area (from TB 1.1)
 # 31 done, ~ 15.91% Completed ~ 15.79% area (from TB 1.1)
 # 30 done, ~ 16.36% Completed ~ 16.28% area (from TB 1.1)
 # 29 done, ~ 16.82% Completed ~ 16.77% area (from TB 1.1)
 # 28 done, ~ 17.27% Completed ~ 17.23% area (from TB 1.1)
 # 27 done, ~ 17.73% Completed ~ 17.72% area (from TB 1.1)
 # 26 done, ~ 18.18% Completed ~ 18.21% area (from TB 1.1)
 # 25 done, ~ 18.64% Completed ~ 18.70% area (from TB 1.1)
 # 24 done, ~ 19.09% Completed ~ 19.16% area (from TB 1.1)
 # 23 done, ~ 19.55% Completed ~ 19.65% area (from TB 1.1)
 # 22 done, ~ 20.00% Completed ~ 20.13% area (from TB 1.1)
Client: 2:1:INIT :N_TMPL=220 SOCK_CNT=2 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 3:1:INIT :N_TMPL=220 SOCK_CNT=3 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 21 done, ~ 20.45% Completed ~ 20.62% area (from TB 1.1)
Client: 4:1:INIT :N_TMPL=220 SOCK_CNT=4 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
```

```
Client: 5:1:INIT :N_TMPL=220 SOCK_CNT=5 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 6:1:INIT :N_TMPL=220 SOCK_CNT=6 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 7:1:INIT :N_TMPL=220 SOCK_CNT=7 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 8:1:INIT :N_TMPL=220 SOCK_CNT=8 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 9:1:INIT :N_TMPL=220 SOCK_CNT=9 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:10:1:INIT :N_TMPL=220 SOCK_CNT=10 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:11:1:INIT :N_TMPL=220 SOCK_CNT=11 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:12:1:INIT :N_TMPL=220 SOCK_CNT=12 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:13:1:INIT :N_TMPL=220 SOCK_CNT=13 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 20 done, ~ 20.91% Completed ~ 21.08% area (from TB 1.1)
Client:14:1:INIT :N_TMPL=220 SOCK_CNT=14 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:15:1:INIT :N_TMPL=220 SOCK_CNT=15 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:16:1:INIT :N_TMPL=220 SOCK_CNT=16 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 17 done, ~ 21.36% Completed ~ 21.57% area (from TB 1.1)
 # 18 done, ~ 21.82% Completed ~ 22.06% area (from TB 1.1)
 # 19 done, ~ 22.27% Completed ~ 22.55% area (from TB 1.1)
 # 16 done, ~ 22.73% Completed ~ 23.01% area (from TB 1.1)
 # 14 done, ~ 23.18% Completed ~ 23.50% area (from TB 1.1)
 # 10 done, ~ 23.64% Completed ~ 23.99% area (from TB 1.1)
 # 6 done, ~ 24.09% Completed ~ 24.48% area (from TB 1.1)
 # 13 done, ~ 24.55% Completed ~ 24.97% area (from TB 1.1)
 # 9 done, ~ 25.00% Completed ~ 25.46% area (from TB 1.1)
 # 12 done, ~ 25.45% Completed ~ 25.92% area (from TB 1.1)
Client:17:1:INIT :N_TMPL=220 SOCK_CNT=17 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
```

```
Client:18:1:INIT :N_TMPL=220 SOCK_CNT=18 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 8 done, ~ 25.91% Completed ~ 26.38% area (from TB 1.1)
Client:19:1:INIT :N_TMPL=220 SOCK_CNT=19 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 15 done, ~ 26.36% Completed ~ 26.87% area (from TB 1.1)
 # 7 done, ~ 26.82% Completed ~ 27.36% area (from TB 1.1)
 # 5 done, ~ 27.27% Completed ~ 27.85% area (from TB 1.1)
 # 4 done, ~ 27.73% Completed ~ 28.31% area (from TB 1.1)
 # 11 done, ~ 28.18% Completed ~ 28.80% area (from TB 1.1)
 # 2 done, ~ 28.64% Completed ~ 29.25% area (from TB 1.1)
 # 3 done, ~ 29.09% Completed ~ 29.71% area (from TB 1.1)
 # 0 done, ~ 29.55% Completed ~ 30.14% area (from TB 1.1)
 # 1 done, ~ 30.00% Completed ~ 30.60% area (from TB 1.1)
 # 142 done, ~ 30.45% Completed ~ 31.06% area (from TB 1.2)
 # 143 done, ~ 30.91% Completed ~ 31.52% area (from TB 1.2)
 # 141 done, ~ 31.36% Completed ~ 31.98% area (from TB 1.2)
 # 140 done, ~ 31.82% Completed ~ 32.43% area (from TB 1.2)
 # 139 done, ~ 32.27% Completed ~ 32.89% area (from TB 1.2)
 # 135 done, ~ 32.73% Completed ~ 33.38% area (from TB 1.2)
Client:20:1:INIT :N_TMPL=220 SOCK_CNT=20 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 126 done, ~ 33.18% Completed ~ 33.87% area (from TB 1.2)
 # 136 done, ~ 33.64% Completed ~ 34.36% area (from TB 1.2)
Client:21:1:INIT :N_TMPL=220 SOCK_CNT=21 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 133 done, ~ 34.09% Completed ~ 34.85% area (from TB 1.2)
 # 128 done, ~ 34.55% Completed ~ 35.34% area (from TB 1.2)
 # 132 done, ~ 35.00% Completed ~ 35.83% area (from TB 1.2)
 # 137 done, ~ 35.45% Completed ~ 36.32% area (from TB 1.2)
 # 127 done, ~ 35.91% Completed ~ 36.81% area (from TB 1.2)
 # 138 done, ~ 36.36% Completed ~ 37.30% area (from TB 1.2)
Client:22:1:INIT :N_TMPL=220 SOCK_CNT=22 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 134 done, ~ 36.82% Completed ~ 37.79% area (from TB 1.2)
 # 125 done, ~ 37.27% Completed ~ 38.28% area (from TB 1.2)
 # 131 done, ~ 37.73% Completed ~ 38.77% area (from TB 1.2)
 # 129 done, ~ 38.18% Completed ~ 39.26% area (from TB 1.2)
 # 130 done, ~ 38.64% Completed ~ 39.75% area (from TB 1.2)
 # 124 done, ~ 39.09% Completed ~ 40.24% area (from TB 1.2)
 # 121 done, ~ 39.55% Completed ~ 40.73% area (from TB 1.2)
Client:23:1:INIT :N_TMPL=220 SOCK_CNT=23 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
```

```
Client:24:1:INIT :N_TMPL=220 SOCK_CNT=24 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:25:1:INIT :N_TMPL=220 SOCK_CNT=25 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 116 done, ~ 40.00% Completed ~ 41.22% area (from TB 1.2)
 # 123 done, ~ 40.45% Completed ~ 41.71% area (from TB 1.2)
 # 122 done, ~ 40.91% Completed ~ 42.20% area (from TB 1.2)
 # 120 done, ~ 41.36% Completed ~ 42.69% area (from TB 1.2)
 # 111 done, ~ 41.82% Completed ~ 43.18% area (from TB 1.2)
 # 119 done, ~ 42.27% Completed ~ 43.67% area (from TB 1.2)
 # 114 done, ~ 42.73% Completed ~ 44.16% area (from TB 1.2)
 # 108 done, ~ 43.18% Completed ~ 44.64% area (from TB 1.2)
 # 118 done, ~ 43.64% Completed ~ 45.13% area (from TB 1.2)
 # 117 done, ~ 44.09% Completed ~ 45.62% area (from TB 1.2)
 # 115 done, ~ 44.55% Completed ~ 46.11% area (from TB 1.2)
 # 106 done, ~ 45.00% Completed ~ 46.60% area (from TB 1.2)
 # 107 done, ~ 45.45% Completed ~ 47.09% area (from TB 1.2)
 # 110 done, ~ 45.91% Completed ~ 47.58% area (from TB 1.2)
 # 103 done, ~ 46.36% Completed ~ 48.03% area (from TB 1.2)
 # 113 done, ~ 46.82% Completed ~ 48.52% area (from TB 1.2)
 # 105 done, ~ 47.27% Completed ~ 49.01% area (from TB 1.2)
 # 104 done, ~ 47.73% Completed ~ 49.50% area (from TB 1.2)
 # 102 done, ~ 48.18% Completed ~ 49.99% area (from TB 1.2)
 # 109 done, ~ 48.64% Completed ~ 50.48% area (from TB 1.2)
 # 112 done, ~ 49.09% Completed ~ 50.97% area (from TB 1.2)
 # 98 done, ~ 49.55% Completed ~ 51.46% area (from TB 1.2)
 # 101 done, ~ 50.00% Completed ~ 51.95% area (from TB 1.2)
 # 97 done, ~ 50.45% Completed ~ 52.44% area (from TB 1.2)
 # 93 done, ~ 50.91% Completed ~ 52.93% area (from TB 1.2)
 # 100 done, ~ 51.36% Completed ~ 53.42% area (from TB 1.2)
 # 99 done, ~ 51.82% Completed ~ 53.90% area (from TB 1.2)
 # 96 done, ~ 52.27% Completed ~ 54.39% area (from TB 1.2)
 # 95 done, ~ 52.73% Completed ~ 54.88% area (from TB 1.2)
 # 94 done, ~ 53.18% Completed ~ 55.37% area (from TB 1.2)
 # 83 done, ~ 53.64% Completed ~ 55.86% area (from TB 1.2)
 # 92 done, ~ 54.09% Completed ~ 56.35% area (from TB 1.2)
 # 88 done, ~ 54.55% Completed ~ 56.84% area (from TB 1.2)
 # 78 done, ~ 55.00% Completed ~ 57.09% area (from TB 1.2)
 # 91 done, ~ 55.45% Completed ~ 57.58% area (from TB 1.2)
Client:26:1:INIT :N_TMPL=220 SOCK_CNT=26 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 90 done, ~ 55.91% Completed ~ 58.07% area (from TB 1.2)
 # 89 done, ~ 56.36% Completed ~ 58.56% area (from TB 1.2)
Client:27:1:INIT :N_TMPL=220 SOCK_CNT=27 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
```

```
# 87 done, ~ 56.82% Completed ~ 59.05% area (from TB 1.2)
# 86 done, ~ 57.27% Completed ~ 59.54% area (from TB 1.2)
# 68 done, ~ 57.73% Completed ~ 59.77% area (from TB 1.2)
# 73 done, ~ 58.18% Completed ~ 60.02% area (from TB 1.2)
# 84 done, ~ 58.64% Completed ~ 60.51% area (from TB 1.2)
# 81 done, ~ 59.09% Completed ~ 61.00% area (from TB 1.2)
# 80 done, ~ 59.55% Completed ~ 61.49% area (from TB 1.2)
# 85 done, ~ 60.00% Completed ~ 61.98% area (from TB 1.2)
# 82 done, ~ 60.45% Completed ~ 62.47% area (from TB 1.2)
# 79 done, ~ 60.91% Completed ~ 62.96% area (from TB 1.2)
# 77 done, ~ 61.36% Completed ~ 63.45% area (from TB 1.2)
# 72 done, ~ 61.82% Completed ~ 63.94% area (from TB 1.2)
# 76 done, ~ 62.27% Completed ~ 64.43% area (from TB 1.2)
# 75 done, ~ 62.73% Completed ~ 64.92% area (from TB 1.2)
# 66 done, ~ 63.18% Completed ~ 65.38% area (from TB 1.2)
# 65 done, ~ 63.64% Completed ~ 65.83% area (from TB 1.2)
# 67 done, ~ 64.09% Completed ~ 66.29% area (from TB 1.2)
# 215 done, ~ 64.55% Completed ~ 66.75% area (from TB 1.3)
# 201 done, ~ 65.00% Completed ~ 67.24% area (from TB 1.3)
# 70 done, ~ 65.45% Completed ~ 67.73% area (from TB 1.2)
# 207 done, ~ 65.91% Completed ~ 68.19% area (from TB 1.3)
# 202 done, ~ 66.36% Completed ~ 68.65% area (from TB 1.3)
# 196 done, ~ 66.82% Completed ~ 68.75% area (from TB 1.3)
# 64 done, ~ 67.27% Completed ~ 69.21% area (from TB 1.2)
# 204 done, ~ 67.73% Completed ~ 69.70% area (from TB 1.3)
# 200 done, ~ 68.18% Completed ~ 70.19% area (from TB 1.3)
# 205 done, ~ 68.64% Completed ~ 70.68% area (from TB 1.3)
# 217 done, ~ 69.09% Completed ~ 71.11% area (from TB 1.3)
# 216 done, ~ 69.55% Completed ~ 71.57% area (from TB 1.3)
# 191 done, ~ 70.00% Completed ~ 71.67% area (from TB 1.3)
# 212 done, ~ 70.45% Completed ~ 72.13% area (from TB 1.3)
# 74 done, ~ 70.91% Completed ~ 72.62% area (from TB 1.2)
# 206 done, ~ 71.36% Completed ~ 73.11% area (from TB 1.3)
# 197 done, ~ 71.82% Completed ~ 73.57% area (from TB 1.3)
# 186 done, ~ 72.27% Completed ~ 73.67% area (from TB 1.3)
# 213 done, ~ 72.73% Completed ~ 74.12% area (from TB 1.3)
# 199 done, ~ 73.18% Completed ~ 74.61% area (from TB 1.3)
# 181 done, ~ 73.64% Completed ~ 74.71% area (from TB 1.3)
# 209 done, ~ 74.09% Completed ~ 75.20% area (from TB 1.3)
# 71 done, ~ 74.55% Completed ~ 75.69% area (from TB 1.2)
# 203 done, ~ 75.00% Completed ~ 76.18% area (from TB 1.3)
# 69 done, ~ 75.45% Completed ~ 76.67% area (from TB 1.2)
# 214 done, ~ 75.91% Completed ~ 77.13% area (from TB 1.3)
# 192 done, ~ 76.36% Completed ~ 77.59% area (from TB 1.3)
# 210 done, ~ 76.82% Completed ~ 78.08% area (from TB 1.3)
# 198 done, ~ 77.27% Completed ~ 78.57% area (from TB 1.3)
# 208 done, ~ 77.73% Completed ~ 79.06% area (from TB 1.3)
# 180 done, ~ 78.18% Completed ~ 79.26% area (from TB 1.3)
```

```
 # 187 done, ~ 78.64% Completed ~ 79.72% area (from TB 1.3)
 # 211 done, ~ 79.09% Completed ~ 80.21% area (from TB 1.3)
 # 176 done, ~ 79.55% Completed ~ 80.70% area (from TB 1.3)
 # 177 done, ~ 80.00% Completed ~ 81.16% area (from TB 1.3)
 # 178 done, ~ 80.45% Completed ~ 81.61% area (from TB 1.3)
 # 179 done, ~ 80.91% Completed ~ 81.89% area (from TB 1.3)
 # 190 done, ~ 81.36% Completed ~ 82.25% area (from TB 1.3)
 # 185 done, ~ 81.82% Completed ~ 82.61% area (from TB 1.3)
 # 182 done, ~ 82.27% Completed ~ 83.07% area (from TB 1.3)
 # 195 done, ~ 82.73% Completed ~ 83.42% area (from TB 1.3)
 # 194 done, ~ 83.18% Completed ~ 83.91% area (from TB 1.3)
 # 175 done, ~ 83.64% Completed ~ 84.29% area (from TB 1.3)
 # 189 done, ~ 84.09% Completed ~ 84.78% area (from TB 1.3)
 # 172 done, ~ 84.55% Completed ~ 85.24% area (from TB 1.3)
 # 188 done, ~ 85.00% Completed ~ 85.73% area (from TB 1.3)
 # 171 done, ~ 85.45% Completed ~ 86.22% area (from TB 1.3)
 # 167 done, ~ 85.91% Completed ~ 86.68% area (from TB 1.3)
 # 153 done, ~ 86.36% Completed ~ 86.72% area (from TB 1.3)
 # 154 done, ~ 86.82% Completed ~ 86.76% area (from TB 1.3)
 # 193 done, ~ 87.27% Completed ~ 87.25% area (from TB 1.3)
 # 174 done, ~ 87.73% Completed ~ 87.62% area (from TB 1.3)
 # 162 done, ~ 88.18% Completed ~ 88.08% area (from TB 1.3)
 # 156 done, ~ 88.64% Completed ~ 88.57% area (from TB 1.3)
 # 155 done, ~ 89.09% Completed ~ 89.06% area (from TB 1.3)
 # 173 done, ~ 89.55% Completed ~ 89.52% area (from TB 1.3)
 # 169 done, ~ 90.00% Completed ~ 90.01% area (from TB 1.3)
 # 184 done, ~ 90.45% Completed ~ 90.50% area (from TB 1.3)
 # 161 done, ~ 90.91% Completed ~ 90.99% area (from TB 1.3)
 # 168 done, ~ 91.36% Completed ~ 91.48% area (from TB 1.3)
 # 170 done, ~ 91.82% Completed ~ 91.97% area (from TB 1.3)
 # 183 done, ~ 92.27% Completed ~ 92.46% area (from TB 1.3)
 # 157 done, ~ 92.73% Completed ~ 92.92% area (from TB 1.3)
 # 150 done, ~ 93.18% Completed ~ 93.21% area (from TB 1.3)
 # 166 done, ~ 93.64% Completed ~ 93.70% area (from TB 1.3)
 # 147 done, ~ 94.09% Completed ~ 94.00% area (from TB 1.3)
 # 152 done, ~ 94.55% Completed ~ 94.46% area (from TB 1.3)
 # 151 done, ~ 95.00% Completed ~ 94.95% area (from TB 1.3)
 # 163 done, ~ 95.45% Completed ~ 95.44% area (from TB 1.3)
 # 164 done, ~ 95.91% Completed ~ 95.93% area (from TB 1.3)
 # 165 done, ~ 96.36% Completed ~ 96.42% area (from TB 1.3)
 # 159 done, ~ 96.82% Completed ~ 96.91% area (from TB 1.3)
 # 160 done, ~ 97.27% Completed ~ 97.40% area (from TB 1.3)
 # 144 done, ~ 97.73% Completed ~ 97.67% area (from TB 1.3)
 # 148 done, ~ 98.18% Completed ~ 98.16% area (from TB 1.3)
 # 149 done, ~ 98.64% Completed ~ 98.62% area (from TB 1.3)
 # 146 done, ~ 99.09% Completed ~ 99.05% area (from TB 1.3)
 # 145 done, ~ 99.55% Completed ~ 99.51% area (from TB 1.3)
Client:28:1:INIT :N_TMPL=220 SOCK_CNT=28 JCF=MOF_NONE_0210.pjx
```

```
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 158 done, ~ 100.00% Completed ~ 100.00% area (from TB 1.3)

Concurrent[1]: All data is present in the output for TB 1.

Pipeline flow complete for TB 1, Wed Feb 12 11:52:50 2014

 TB1 Times:User: 1.02 Sys: 0.36 Elapsed: 56.94 Memory: 280.079M

 Pipeline flow started for TB 2, Wed Feb 12 11:52:50 2014

 spatial context: 64 templates generated for stripe 2.1 of 3
Client:29:1:INIT :N_TMPL=64 SOCK_CNT=29 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=2
LASTTB=3
Client:30:1:INIT :N_TMPL=64 SOCK_CNT=30 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=2
LASTTB=3
Client:31:1:INIT :N_TMPL=64 SOCK_CNT=31 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=2
LASTTB=3
Client:32:1:INIT :N_TMPL=64 SOCK_CNT=32 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=2
LASTTB=3
Client:33:1:INIT :N_TMPL=64 SOCK_CNT=33 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=2
LASTTB=3
Generating spatial templates for TB 3.
 spatial context: 80 templates generated for stripe 2.2 of 3
     Times: User: 1.07 Sys: 0.79 Elapsed: 57.24 Memory: 379.783M
 spatial context: 74 templates generated for stripe 2.3 of 3
     Times: User: 0.12 Sys: 0.00 Elapsed: 0.14 Memory: 376.850M
Calculating context for holding cells.
 holder context: 10%
 holder context: 20%
 holder context: 30%
 holder context: 40%
 holder context: 50%
 holder context: 60%
 holder context: 70%
 holder context: 80%
 holder context: 90%
 holder context: Done Wed Feb 12 11:52:51 2014

     Times: User: 0.00 Sys: 0.00 Elapsed: 0.01 Memory: 371.228M
 220 output templates (218 cluster templates) generated for TB2:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
```

```
 Total topcell bounding area: 85200.1608 square microns.
 Sum of all template bounding areas: 78141.6062 square microns.
 Sum of all ambit-biased template bounding areas: 99135.4797
square microns.

 Hierman BackEnd Times: User: 1.97 Sys: 0.92 Elapsed: 58.28 Memory:
379.783M
Client: executing mktop in a separate thread...
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 0.91% Completed ~ 0.00% area (from TB 2)
 # 55 done, ~ 1.36% Completed ~ 0.18% area (from TB 2.1)
 # 48 done, ~ 1.82% Completed ~ 0.68% area (from TB 2.1)
 # 44 done, ~ 2.27% Completed ~ 0.85% area (from TB 2.1)
 # 56 done, ~ 2.73% Completed ~ 1.20% area (from TB 2.1)
 # 39 done, ~ 3.18% Completed ~ 1.70% area (from TB 2.1)
 # 40 done, ~ 3.64% Completed ~ 2.00% area (from TB 2.1)
 # 45 done, ~ 4.09% Completed ~ 2.46% area (from TB 2.1)
 # 52 done, ~ 4.55% Completed ~ 2.93% area (from TB 2.1)
 # 58 done, ~ 5.00% Completed ~ 3.42% area (from TB 2.1)
 # 60 done, ~ 5.45% Completed ~ 3.89% area (from TB 2.1)
 # 41 done, ~ 5.91% Completed ~ 4.19% area (from TB 2.1)
 # 59 done, ~ 6.36% Completed ~ 4.69% area (from TB 2.1)
 # 35 done, ~ 6.82% Completed ~ 5.19% area (from TB 2.1)
 # 53 done, ~ 7.27% Completed ~ 5.66% area (from TB 2.1)
 # 31 done, ~ 7.73% Completed ~ 6.15% area (from TB 2.1)
 # 49 done, ~ 8.18% Completed ~ 6.65% area (from TB 2.1)
 # 27 done, ~ 8.64% Completed ~ 7.15% area (from TB 2.1)
 # 54 done, ~ 9.09% Completed ~ 7.59% area (from TB 2.1)
 # 57 done, ~ 9.55% Completed ~ 8.05% area (from TB 2.1)
 # 51 done, ~ 10.00% Completed ~ 8.52% area (from TB 2.1)
 # 46 done, ~ 10.45% Completed ~ 8.99% area (from TB 2.1)
 # 61 done, ~ 10.91% Completed ~ 9.48% area (from TB 2.1)
 # 50 done, ~ 11.36% Completed ~ 9.96% area (from TB 2.1)
 # 26 done, ~ 11.82% Completed ~ 10.45% area (from TB 2.1)
 # 63 done, ~ 12.27% Completed ~ 10.92% area (from TB 2.1)
 # 62 done, ~ 12.73% Completed ~ 11.42% area (from TB 2.1)
 # 47 done, ~ 13.18% Completed ~ 11.91% area (from TB 2.1)
 # 30 done, ~ 13.64% Completed ~ 12.41% area (from TB 2.1)
 # 25 done, ~ 14.09% Completed ~ 12.90% area (from TB 2.1)
 # 38 done, ~ 14.55% Completed ~ 13.40% area (from TB 2.1)
 # 34 done, ~ 15.00% Completed ~ 13.90% area (from TB 2.1)
 # 24 done, ~ 15.45% Completed ~ 14.37% area (from TB 2.1)
 # 42 done, ~ 15.91% Completed ~ 14.86% area (from TB 2.1)
 # 12 done, ~ 16.36% Completed ~ 15.33% area (from TB 2.1)
 # 11 done, ~ 16.82% Completed ~ 15.83% area (from TB 2.1)
 # 20 done, ~ 17.27% Completed ~ 16.30% area (from TB 2.1)
```

```
# 43 done, ~ 17.73% Completed ~ 16.79% area (from TB 2.1)
# 37 done, ~ 18.18% Completed ~ 17.26% area (from TB 2.1)
# 36 done, ~ 18.64% Completed ~ 17.76% area (from TB 2.1)
# 16 done, ~ 19.09% Completed ~ 18.23% area (from TB 2.1)
# 32 done, ~ 19.55% Completed ~ 18.72% area (from TB 2.1)
# 28 done, ~ 20.00% Completed ~ 19.22% area (from TB 2.1)
# 33 done, ~ 20.45% Completed ~ 19.69% area (from TB 2.1)
# 10 done, ~ 20.91% Completed ~ 20.19% area (from TB 2.1)
# 23 done, ~ 21.36% Completed ~ 20.68% area (from TB 2.1)
# 9 done, ~ 21.82% Completed ~ 21.18% area (from TB 2.1)
# 29 done, ~ 22.27% Completed ~ 21.65% area (from TB 2.1)
# 0 done, ~ 22.73% Completed ~ 22.12% area (from TB 2.1)
# 3 done, ~ 23.18% Completed ~ 22.56% area (from TB 2.1)
# 7 done, ~ 23.64% Completed ~ 23.03% area (from TB 2.1)
# 19 done, ~ 24.09% Completed ~ 23.53% area (from TB 2.1)
# 2 done, ~ 24.55% Completed ~ 24.00% area (from TB 2.1)
# 21 done, ~ 25.00% Completed ~ 24.50% area (from TB 2.1)
# 1 done, ~ 25.45% Completed ~ 24.97% area (from TB 2.1)
# 5 done, ~ 25.91% Completed ~ 25.46% area (from TB 2.1)
# 22 done, ~ 26.36% Completed ~ 25.96% area (from TB 2.1)
# 15 done, ~ 26.82% Completed ~ 26.45% area (from TB 2.1)
# 8 done, ~ 27.27% Completed ~ 26.95% area (from TB 2.1)
# 4 done, ~ 27.73% Completed ~ 27.45% area (from TB 2.1)
# 136 done, ~ 28.18% Completed ~ 27.94% area (from TB 2.2)
# 6 done, ~ 28.64% Completed ~ 28.44% area (from TB 2.1)
# 135 done, ~ 29.09% Completed ~ 28.93% area (from TB 2.2)
# 143 done, ~ 29.55% Completed ~ 29.40% area (from TB 2.2)
# 13 done, ~ 30.00% Completed ~ 29.90% area (from TB 2.1)
# 14 done, ~ 30.45% Completed ~ 30.39% area (from TB 2.1)
# 17 done, ~ 30.91% Completed ~ 30.89% area (from TB 2.1)
# 138 done, ~ 31.36% Completed ~ 31.38% area (from TB 2.2)
# 124 done, ~ 31.82% Completed ~ 31.88% area (from TB 2.2)
# 119 done, ~ 32.27% Completed ~ 32.34% area (from TB 2.2)
# 137 done, ~ 32.73% Completed ~ 32.84% area (from TB 2.2)
# 116 done, ~ 33.18% Completed ~ 33.08% area (from TB 2.2)
# 18 done, ~ 33.64% Completed ~ 33.57% area (from TB 2.1)
# 113 done, ~ 34.09% Completed ~ 33.83% area (from TB 2.2)
# 140 done, ~ 34.55% Completed ~ 34.32% area (from TB 2.2)
# 110 done, ~ 35.00% Completed ~ 34.57% area (from TB 2.2)
# 122 done, ~ 35.45% Completed ~ 35.03% area (from TB 2.2)
# 139 done, ~ 35.91% Completed ~ 35.53% area (from TB 2.2)
# 131 done, ~ 36.36% Completed ~ 36.02% area (from TB 2.2)
# 142 done, ~ 36.82% Completed ~ 36.52% area (from TB 2.2)
# 141 done, ~ 37.27% Completed ~ 37.01% area (from TB 2.2)
# 123 done, ~ 37.73% Completed ~ 37.51% area (from TB 2.2)
# 130 done, ~ 38.18% Completed ~ 38.00% area (from TB 2.2)
# 128 done, ~ 38.64% Completed ~ 38.47% area (from TB 2.2)
# 107 done, ~ 39.09% Completed ~ 38.97% area (from TB 2.2)
```

```
# 129 done, ~ 39.55% Completed ~ 39.47% area (from TB 2.2)
# 104 done, ~ 40.00% Completed ~ 39.96% area (from TB 2.2)
# 101 done, ~ 40.45% Completed ~ 40.46% area (from TB 2.2)
# 127 done, ~ 40.91% Completed ~ 40.96% area (from TB 2.2)
# 102 done, ~ 41.36% Completed ~ 41.45% area (from TB 2.2)
# 98 done, ~ 41.82% Completed ~ 41.92% area (from TB 2.2)
# 117 done, ~ 42.27% Completed ~ 42.39% area (from TB 2.2)
# 120 done, ~ 42.73% Completed ~ 42.88% area (from TB 2.2)
# 126 done, ~ 43.18% Completed ~ 43.38% area (from TB 2.2)
# 118 done, ~ 43.64% Completed ~ 43.85% area (from TB 2.2)
# 106 done, ~ 44.09% Completed ~ 44.34% area (from TB 2.2)
# 114 done, ~ 44.55% Completed ~ 44.84% area (from TB 2.2)
# 134 done, ~ 45.00% Completed ~ 45.34% area (from TB 2.2)
# 111 done, ~ 45.45% Completed ~ 45.83% area (from TB 2.2)
# 99 done, ~ 45.91% Completed ~ 46.30% area (from TB 2.2)
# 100 done, ~ 46.36% Completed ~ 46.79% area (from TB 2.2)
# 132 done, ~ 46.82% Completed ~ 47.29% area (from TB 2.2)
# 103 done, ~ 47.27% Completed ~ 47.78% area (from TB 2.2)
# 105 done, ~ 47.73% Completed ~ 48.28% area (from TB 2.2)
# 108 done, ~ 48.18% Completed ~ 48.78% area (from TB 2.2)
# 125 done, ~ 48.64% Completed ~ 49.27% area (from TB 2.2)
# 115 done, ~ 49.09% Completed ~ 49.77% area (from TB 2.2)
# 97 done, ~ 49.55% Completed ~ 50.26% area (from TB 2.2)
# 133 done, ~ 50.00% Completed ~ 50.76% area (from TB 2.2)
# 121 done, ~ 50.45% Completed ~ 51.25% area (from TB 2.2)
# 88 done, ~ 50.91% Completed ~ 51.75% area (from TB 2.2)
# 91 done, ~ 51.36% Completed ~ 52.24% area (from TB 2.2)
# 94 done, ~ 51.82% Completed ~ 52.74% area (from TB 2.2)
# 109 done, ~ 52.27% Completed ~ 53.24% area (from TB 2.2)
# 217 done, ~ 52.73% Completed ~ 53.28% area (from TB 2.3)
# 112 done, ~ 53.18% Completed ~ 53.77% area (from TB 2.2)
# 77 done, ~ 53.64% Completed ~ 54.27% area (from TB 2.2)
# 87 done, ~ 54.09% Completed ~ 54.76% area (from TB 2.2)
# 86 done, ~ 54.55% Completed ~ 55.26% area (from TB 2.2)
# 82 done, ~ 55.00% Completed ~ 55.73% area (from TB 2.2)
# 68 done, ~ 55.45% Completed ~ 56.22% area (from TB 2.2)
# 70 done, ~ 55.91% Completed ~ 56.72% area (from TB 2.2)
# 81 done, ~ 56.36% Completed ~ 57.19% area (from TB 2.2)
# 73 done, ~ 56.82% Completed ~ 57.69% area (from TB 2.2)
# 69 done, ~ 57.27% Completed ~ 58.18% area (from TB 2.2)
# 64 done, ~ 57.73% Completed ~ 58.65% area (from TB 2.2)
# 95 done, ~ 58.18% Completed ~ 59.15% area (from TB 2.2)
# 212 done, ~ 58.64% Completed ~ 59.44% area (from TB 2.3)
# 67 done, ~ 59.09% Completed ~ 59.91% area (from TB 2.2)
# 85 done, ~ 59.55% Completed ~ 60.40% area (from TB 2.2)
# 80 done, ~ 60.00% Completed ~ 60.87% area (from TB 2.2)
# 78 done, ~ 60.45% Completed ~ 61.37% area (from TB 2.2)
# 201 done, ~ 60.91% Completed ~ 61.47% area (from TB 2.3)
```

```
# 90 done, ~ 61.36% Completed ~ 61.96% area (from TB 2.2)
# 207 done, ~ 61.82% Completed ~ 62.46% area (from TB 2.3)
# 79 done, ~ 62.27% Completed ~ 62.96% area (from TB 2.2)
# 93 done, ~ 62.73% Completed ~ 63.45% area (from TB 2.2)
# 202 done, ~ 63.18% Completed ~ 63.61% area (from TB 2.3)
# 71 done, ~ 63.64% Completed ~ 64.11% area (from TB 2.2)
# 204 done, ~ 64.09% Completed ~ 64.60% area (from TB 2.3)
# 89 done, ~ 64.55% Completed ~ 65.10% area (from TB 2.2)
# 96 done, ~ 65.00% Completed ~ 65.59% area (from TB 2.2)
# 72 done, ~ 65.45% Completed ~ 66.09% area (from TB 2.2)
# 199 done, ~ 65.91% Completed ~ 66.58% area (from TB 2.3)
# 203 done, ~ 66.36% Completed ~ 67.08% area (from TB 2.3)
# 200 done, ~ 66.82% Completed ~ 67.57% area (from TB 2.3)
# 208 done, ~ 67.27% Completed ~ 68.06% area (from TB 2.3)
# 92 done, ~ 67.73% Completed ~ 68.56% area (from TB 2.2)
# 84 done, ~ 68.18% Completed ~ 69.05% area (from TB 2.2)
# 83 done, ~ 68.64% Completed ~ 69.55% area (from TB 2.2)
# 213 done, ~ 69.09% Completed ~ 69.93% area (from TB 2.3)
# 195 done, ~ 69.55% Completed ~ 70.24% area (from TB 2.3)
# 76 done, ~ 70.00% Completed ~ 70.73% area (from TB 2.2)
# 66 done, ~ 70.45% Completed ~ 71.23% area (from TB 2.2)
# 192 done, ~ 70.91% Completed ~ 71.54% area (from TB 2.3)
# 198 done, ~ 71.36% Completed ~ 71.82% area (from TB 2.3)
# 65 done, ~ 71.82% Completed ~ 72.32% area (from TB 2.2)
# 75 done, ~ 72.27% Completed ~ 72.81% area (from TB 2.2)
# 174 done, ~ 72.73% Completed ~ 72.92% area (from TB 2.3)
# 74 done, ~ 73.18% Completed ~ 73.41% area (from TB 2.2)
# 191 done, ~ 73.64% Completed ~ 73.91% area (from TB 2.3)
# 171 done, ~ 74.09% Completed ~ 74.02% area (from TB 2.3)
# 190 done, ~ 74.55% Completed ~ 74.49% area (from TB 2.3)
# 168 done, ~ 75.00% Completed ~ 74.60% area (from TB 2.3)
# 178 done, ~ 75.45% Completed ~ 75.09% area (from TB 2.3)
# 189 done, ~ 75.91% Completed ~ 75.56% area (from TB 2.3)
# 165 done, ~ 76.36% Completed ~ 76.05% area (from TB 2.3)
# 167 done, ~ 76.82% Completed ~ 76.52% area (from TB 2.3)
# 166 done, ~ 77.27% Completed ~ 77.01% area (from TB 2.3)
# 177 done, ~ 77.73% Completed ~ 77.48% area (from TB 2.3)
# 187 done, ~ 78.18% Completed ~ 77.98% area (from TB 2.3)
# 186 done, ~ 78.64% Completed ~ 78.44% area (from TB 2.3)
# 164 done, ~ 79.09% Completed ~ 78.91% area (from TB 2.3)
# 180 done, ~ 79.55% Completed ~ 79.38% area (from TB 2.3)
# 216 done, ~ 80.00% Completed ~ 79.88% area (from TB 2.3)
# 193 done, ~ 80.45% Completed ~ 80.35% area (from TB 2.3)
# 183 done, ~ 80.91% Completed ~ 80.82% area (from TB 2.3)
# 194 done, ~ 81.36% Completed ~ 81.32% area (from TB 2.3)
# 196 done, ~ 81.82% Completed ~ 81.76% area (from TB 2.3)
# 156 done, ~ 82.27% Completed ~ 81.81% area (from TB 2.3)
# 197 done, ~ 82.73% Completed ~ 82.28% area (from TB 2.3)
```

```
 # 170 done, ~ 83.18% Completed ~ 82.75% area (from TB 2.3)
 # 181 done, ~ 83.64% Completed ~ 83.25% area (from TB 2.3)
 # 184 done, ~ 84.09% Completed ~ 83.74% area (from TB 2.3)
 # 163 done, ~ 84.55% Completed ~ 84.21% area (from TB 2.3)
 # 215 done, ~ 85.00% Completed ~ 84.71% area (from TB 2.3)
 # 173 done, ~ 85.45% Completed ~ 85.18% area (from TB 2.3)
 # 206 done, ~ 85.91% Completed ~ 85.67% area (from TB 2.3)
 # 176 done, ~ 86.36% Completed ~ 86.14% area (from TB 2.3)
 # 214 done, ~ 86.82% Completed ~ 86.64% area (from TB 2.3)
 # 179 done, ~ 87.27% Completed ~ 87.02% area (from TB 2.3)
 # 172 done, ~ 87.73% Completed ~ 87.39% area (from TB 2.3)
 # 175 done, ~ 88.18% Completed ~ 87.76% area (from TB 2.3)
 # 211 done, ~ 88.64% Completed ~ 88.25% area (from TB 2.3)
 # 169 done, ~ 89.09% Completed ~ 88.62% area (from TB 2.3)
 # 151 done, ~ 89.55% Completed ~ 88.90% area (from TB 2.3)
 # 210 done, ~ 90.00% Completed ~ 89.39% area (from TB 2.3)
 # 209 done, ~ 90.45% Completed ~ 89.89% area (from TB 2.3)
 # 205 done, ~ 90.91% Completed ~ 90.36% area (from TB 2.3)
 # 146 done, ~ 91.36% Completed ~ 90.86% area (from TB 2.3)
 # 147 done, ~ 91.82% Completed ~ 91.35% area (from TB 2.3)
 # 159 done, ~ 92.27% Completed ~ 91.82% area (from TB 2.3)
 # 160 done, ~ 92.73% Completed ~ 92.29% area (from TB 2.3)
 # 158 done, ~ 93.18% Completed ~ 92.76% area (from TB 2.3)
 # 157 done, ~ 93.64% Completed ~ 93.21% area (from TB 2.3)
 # 185 done, ~ 94.09% Completed ~ 93.70% area (from TB 2.3)
 # 188 done, ~ 94.55% Completed ~ 94.20% area (from TB 2.3)
 # 152 done, ~ 95.00% Completed ~ 94.57% area (from TB 2.3)
 # 144 done, ~ 95.45% Completed ~ 95.04% area (from TB 2.3)
 # 161 done, ~ 95.91% Completed ~ 95.54% area (from TB 2.3)
 # 162 done, ~ 96.36% Completed ~ 96.03% area (from TB 2.3)
 # 182 done, ~ 96.82% Completed ~ 96.53% area (from TB 2.3)
 # 145 done, ~ 97.27% Completed ~ 97.03% area (from TB 2.3)
 # 155 done, ~ 97.73% Completed ~ 97.52% area (from TB 2.3)
 # 154 done, ~ 98.18% Completed ~ 98.02% area (from TB 2.3)
 # 150 done, ~ 98.64% Completed ~ 98.52% area (from TB 2.3)
 # 153 done, ~ 99.09% Completed ~ 99.01% area (from TB 2.3)
 # 148 done, ~ 99.55% Completed ~ 99.51% area (from TB 2.3)
 # 149 done, ~ 100.00% Completed ~ 100.00% area (from TB 2.3)

Concurrent[2]: All data is present in the output for TB 2.

Pipeline flow complete for TB 2, Wed Feb 12 11:54:04 2014

 TB2 Times:User: 2.24 Sys: 1.20 Elapsed: 130.75 Memory: 379.783M

 Pipeline flow started for TB 3, Wed Feb 12 11:54:04 2014

 spatial context: 64 templates generated for stripe 3.1 of 3
```

```
# 63 done, ~ 0.47% Completed ~ 0.43% area (from TB 3.1)
# 41 done, ~ 0.93% Completed ~ 0.70% area (from TB 3.1)
# 40 done, ~ 1.40% Completed ~ 0.97% area (from TB 3.1)
# 48 done, ~ 1.86% Completed ~ 1.42% area (from TB 3.1)
# 44 done, ~ 2.33% Completed ~ 1.58% area (from TB 3.1)
# 37 done, ~ 2.80% Completed ~ 2.00% area (from TB 3.1)
# 52 done, ~ 3.26% Completed ~ 2.43% area (from TB 3.1)
# 36 done, ~ 3.73% Completed ~ 2.87% area (from TB 3.1)
# 60 done, ~ 4.19% Completed ~ 3.30% area (from TB 3.1)
# 51 done, ~ 4.66% Completed ~ 3.72% area (from TB 3.1)
# 50 done, ~ 5.12% Completed ~ 4.15% area (from TB 3.1)
# 39 done, ~ 5.59% Completed ~ 4.59% area (from TB 3.1)
# 58 done, ~ 6.06% Completed ~ 5.03% area (from TB 3.1)
# 53 done, ~ 6.52% Completed ~ 5.46% area (from TB 3.1)
# 49 done, ~ 6.99% Completed ~ 5.90% area (from TB 3.1)
spatial context: 80 templates generated for stripe 3.2 of 3
# 61 done, ~ 7.45% Completed ~ 6.34% area (from TB 3.1)
# 33 done, ~ 7.92% Completed ~ 6.77% area (from TB 3.1)
# 42 done, ~ 8.39% Completed ~ 7.21% area (from TB 3.1)
# 35 done, ~ 8.85% Completed ~ 7.65% area (from TB 3.1)
# 59 done, ~ 9.32% Completed ~ 7.92% area (from TB 3.1)
# 32 done, ~ 9.57% Completed ~ 8.53% area (from TB 3.1)
# 56 done, ~ 10.03% Completed ~ 8.87% area (from TB 3.1)
# 34 done, ~ 10.49% Completed ~ 9.31% area (from TB 3.1)
# 57 done, ~ 10.94% Completed ~ 9.73% area (from TB 3.1)
# 43 done, ~ 11.40% Completed ~ 10.17% area (from TB 3.1)
# 38 done, ~ 11.85% Completed ~ 10.61% area (from TB 3.1)
# 30 done, ~ 12.31% Completed ~ 11.06% area (from TB 3.1)
# 27 done, ~ 12.77% Completed ~ 11.50% area (from TB 3.1)
# 24 done, ~ 13.22% Completed ~ 11.92% area (from TB 3.1)
# 26 done, ~ 13.68% Completed ~ 12.36% area (from TB 3.1)
# 29 done, ~ 14.13% Completed ~ 12.79% area (from TB 3.1)
# 25 done, ~ 14.59% Completed ~ 13.23% area (from TB 3.1)
# 28 done, ~ 15.05% Completed ~ 13.67% area (from TB 3.1)
# 62 done, ~ 15.50% Completed ~ 14.11% area (from TB 3.1)
# 20 done, ~ 15.96% Completed ~ 14.54% area (from TB 3.1)
# 18 done, ~ 16.41% Completed ~ 14.98% area (from TB 3.1)
# 16 done, ~ 16.87% Completed ~ 15.41% area (from TB 3.1)
# 23 done, ~ 17.33% Completed ~ 15.85% area (from TB 3.1)
# 22 done, ~ 17.78% Completed ~ 16.29% area (from TB 3.1)
# 12 done, ~ 18.24% Completed ~ 16.71% area (from TB 3.1)
    Times: User: 0.80 Sys: 0.60 Elapsed: 73.15 Memory: 371.325M
# 19 done, ~ 18.69% Completed ~ 17.16% area (from TB 3.1)
# 21 done, ~ 19.15% Completed ~ 17.60% area (from TB 3.1)
# 7 done, ~ 19.60% Completed ~ 18.02% area (from TB 3.1)
# 10 done, ~ 20.06% Completed ~ 18.46% area (from TB 3.1)
# 14 done, ~ 20.52% Completed ~ 18.90% area (from TB 3.1)
# 11 done, ~ 20.97% Completed ~ 19.34% area (from TB 3.1)
```

```
 # 17 done, ~ 21.43% Completed ~ 19.78% area (from TB 3.1)
 # 3 done, ~ 21.88% Completed ~ 20.20% area (from TB 3.1)
 # 13 done, ~ 22.34% Completed ~ 20.64% area (from TB 3.1)
 # 15 done, ~ 22.80% Completed ~ 21.08% area (from TB 3.1)
 # 9 done, ~ 23.25% Completed ~ 21.52% area (from TB 3.1)
 # 4 done, ~ 23.71% Completed ~ 21.96% area (from TB 3.1)
 # 0 done, ~ 24.16% Completed ~ 22.39% area (from TB 3.1)
 # 8 done, ~ 24.62% Completed ~ 22.83% area (from TB 3.1)
 # 1 done, ~ 25.08% Completed ~ 23.25% area (from TB 3.1)
 # 2 done, ~ 25.53% Completed ~ 23.68% area (from TB 3.1)
 # 5 done, ~ 25.99% Completed ~ 24.12% area (from TB 3.1)
 # 6 done, ~ 26.44% Completed ~ 24.56% area (from TB 3.1)
 spatial context: 74 templates generated for stripe 3.3 of 3
     Times: User: 0.10 Sys: 0.01 Elapsed: 0.06 Memory: 266.398M
Calculating context for holding cells.
 holder context: 10%
 holder context: 20%
 holder context: 30%
 holder context: 40%
 holder context: 50%
 holder context: 60%
 holder context: 70%
 holder context: 80%
 holder context: 90%
 holder context: Done Wed Feb 12 11:54:05 2014

     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 261.590M
 220 output templates (218 cluster templates) generated for TB3:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 85598.4280 square microns.
 Sum of all template bounding areas: 78760.9214 square microns.
 Sum of all ambit-biased template bounding areas: 87266.8274
square microns.

 Hierman BackEnd Times: User: 2.87 Sys: 1.53 Elapsed: 131.51
Memory: 379.816M
Client: executing mktop in a separate thread...
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 26.36% Completed ~ 27.38% area (from TB 3)
Client: executing mktop in a separate thread...
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 26.36% Completed ~ 27.38% area (from TB 3)
Client: executing mktop in a separate thread...
```

```
Building Instance reference list for full output hierarchy
Creating dummy TOPCELL_OUT: TOP ...
Building hierarchy cells ...
# mktop done, ~ 27.27% Completed ~ 27.38% area (from TB 3)
# 55 done, ~ 27.73% Completed ~ 27.57% area (from TB 3.1)
# 54 done, ~ 28.18% Completed ~ 28.03% area (from TB 3.1)
# 47 done, ~ 28.64% Completed ~ 28.52% area (from TB 3.1)
# 45 done, ~ 29.09% Completed ~ 28.99% area (from TB 3.1)
# 31 done, ~ 29.55% Completed ~ 29.48% area (from TB 3.1)
# 46 done, ~ 30.00% Completed ~ 29.96% area (from TB 3.1)
# 116 done, ~ 30.45% Completed ~ 30.21% area (from TB 3.2)
# 110 done, ~ 30.91% Completed ~ 30.46% area (from TB 3.2)
# 113 done, ~ 31.36% Completed ~ 30.73% area (from TB 3.2)
# 115 done, ~ 31.82% Completed ~ 31.22% area (from TB 3.2)
# 114 done, ~ 32.27% Completed ~ 31.71% area (from TB 3.2)
# 111 done, ~ 32.73% Completed ~ 32.20% area (from TB 3.2)
# 112 done, ~ 33.18% Completed ~ 32.69% area (from TB 3.2)
# 107 done, ~ 33.64% Completed ~ 33.19% area (from TB 3.2)
# 108 done, ~ 34.09% Completed ~ 33.68% area (from TB 3.2)
# 104 done, ~ 34.55% Completed ~ 34.17% area (from TB 3.2)
# 109 done, ~ 35.00% Completed ~ 34.66% area (from TB 3.2)
# 106 done, ~ 35.45% Completed ~ 35.16% area (from TB 3.2)
# 105 done, ~ 35.91% Completed ~ 35.65% area (from TB 3.2)
# 103 done, ~ 36.36% Completed ~ 36.14% area (from TB 3.2)
# 102 done, ~ 36.82% Completed ~ 36.63% area (from TB 3.2)
# 101 done, ~ 37.27% Completed ~ 37.12% area (from TB 3.2)
# 100 done, ~ 37.73% Completed ~ 37.62% area (from TB 3.2)
# 99 done, ~ 38.18% Completed ~ 38.11% area (from TB 3.2)
# 98 done, ~ 38.64% Completed ~ 38.58% area (from TB 3.2)
# 97 done, ~ 39.09% Completed ~ 39.07% area (from TB 3.2)
# 96 done, ~ 39.55% Completed ~ 39.56% area (from TB 3.2)
# 95 done, ~ 40.00% Completed ~ 40.05% area (from TB 3.2)
# 94 done, ~ 40.45% Completed ~ 40.54% area (from TB 3.2)
# 92 done, ~ 40.91% Completed ~ 41.03% area (from TB 3.2)
# 93 done, ~ 41.36% Completed ~ 41.52% area (from TB 3.2)
# 91 done, ~ 41.82% Completed ~ 42.01% area (from TB 3.2)
# 88 done, ~ 42.27% Completed ~ 42.51% area (from TB 3.2)
# 90 done, ~ 42.73% Completed ~ 43.00% area (from TB 3.2)
# 89 done, ~ 43.18% Completed ~ 43.49% area (from TB 3.2)
# 86 done, ~ 43.64% Completed ~ 43.98% area (from TB 3.2)
# 87 done, ~ 44.09% Completed ~ 44.47% area (from TB 3.2)
# 85 done, ~ 44.55% Completed ~ 44.96% area (from TB 3.2)
# 84 done, ~ 45.00% Completed ~ 45.45% area (from TB 3.2)
# 82 done, ~ 45.45% Completed ~ 45.93% area (from TB 3.2)
# 81 done, ~ 45.91% Completed ~ 46.40% area (from TB 3.2)
# 80 done, ~ 46.36% Completed ~ 46.88% area (from TB 3.2)
# 83 done, ~ 46.82% Completed ~ 47.37% area (from TB 3.2)
# 79 done, ~ 47.27% Completed ~ 47.86% area (from TB 3.2)
```

```
# 77 done, ~ 47.73% Completed ~ 48.35% area (from TB 3.2)
# 78 done, ~ 48.18% Completed ~ 48.85% area (from TB 3.2)
# 75 done, ~ 48.64% Completed ~ 49.34% area (from TB 3.2)
# 76 done, ~ 49.09% Completed ~ 49.83% area (from TB 3.2)
# 74 done, ~ 49.55% Completed ~ 50.32% area (from TB 3.2)
# 73 done, ~ 50.00% Completed ~ 50.81% area (from TB 3.2)
# 72 done, ~ 50.45% Completed ~ 51.30% area (from TB 3.2)
# 68 done, ~ 50.91% Completed ~ 51.79% area (from TB 3.2)
# 71 done, ~ 51.36% Completed ~ 52.29% area (from TB 3.2)
# 70 done, ~ 51.82% Completed ~ 52.78% area (from TB 3.2)
# 69 done, ~ 52.27% Completed ~ 53.27% area (from TB 3.2)
# 67 done, ~ 52.73% Completed ~ 53.74% area (from TB 3.2)
# 66 done, ~ 53.18% Completed ~ 54.24% area (from TB 3.2)
# 64 done, ~ 53.64% Completed ~ 54.71% area (from TB 3.2)
# 65 done, ~ 54.09% Completed ~ 55.20% area (from TB 3.2)
# 217 done, ~ 54.55% Completed ~ 55.25% area (from TB 3.3)
# 216 done, ~ 55.00% Completed ~ 55.74% area (from TB 3.3)
# 215 done, ~ 55.45% Completed ~ 56.23% area (from TB 3.3)
# 214 done, ~ 55.91% Completed ~ 56.73% area (from TB 3.3)
# 213 done, ~ 56.36% Completed ~ 57.12% area (from TB 3.3)
# 212 done, ~ 56.82% Completed ~ 57.41% area (from TB 3.3)
# 211 done, ~ 57.27% Completed ~ 57.90% area (from TB 3.3)
# 208 done, ~ 57.73% Completed ~ 58.39% area (from TB 3.3)
# 209 done, ~ 58.18% Completed ~ 58.88% area (from TB 3.3)
# 210 done, ~ 58.64% Completed ~ 59.37% area (from TB 3.3)
# 207 done, ~ 59.09% Completed ~ 59.87% area (from TB 3.3)
# 204 done, ~ 59.55% Completed ~ 60.36% area (from TB 3.3)
# 206 done, ~ 60.00% Completed ~ 60.85% area (from TB 3.3)
# 205 done, ~ 60.45% Completed ~ 61.32% area (from TB 3.3)
# 201 done, ~ 60.91% Completed ~ 61.43% area (from TB 3.3)
# 202 done, ~ 61.36% Completed ~ 61.61% area (from TB 3.3)
# 203 done, ~ 61.82% Completed ~ 62.10% area (from TB 3.3)
# 200 done, ~ 62.27% Completed ~ 62.59% area (from TB 3.3)
# 199 done, ~ 62.73% Completed ~ 63.08% area (from TB 3.3)
# 198 done, ~ 63.18% Completed ~ 63.38% area (from TB 3.3)
# 195 done, ~ 63.64% Completed ~ 63.69% area (from TB 3.3)
# 197 done, ~ 64.09% Completed ~ 64.17% area (from TB 3.3)
# 196 done, ~ 64.55% Completed ~ 64.62% area (from TB 3.3)
# 194 done, ~ 65.00% Completed ~ 65.12% area (from TB 3.3)
# 192 done, ~ 65.45% Completed ~ 65.43% area (from TB 3.3)
# 193 done, ~ 65.91% Completed ~ 65.90% area (from TB 3.3)
# 191 done, ~ 66.36% Completed ~ 66.39% area (from TB 3.3)
# 189 done, ~ 66.82% Completed ~ 66.87% area (from TB 3.3)
# 190 done, ~ 67.27% Completed ~ 67.34% area (from TB 3.3)
# 186 done, ~ 67.73% Completed ~ 67.81% area (from TB 3.3)
# 187 done, ~ 68.18% Completed ~ 68.31% area (from TB 3.3)
# 188 done, ~ 68.64% Completed ~ 68.80% area (from TB 3.3)
# 183 done, ~ 69.09% Completed ~ 69.27% area (from TB 3.3)
```

```
 # 184 done, ~ 69.55% Completed ~ 69.76% area (from TB 3.3)
 # 185 done, ~ 70.00% Completed ~ 70.26% area (from TB 3.3)
 # 180 done, ~ 70.45% Completed ~ 70.73% area (from TB 3.3)
 # 181 done, ~ 70.91% Completed ~ 71.22% area (from TB 3.3)
 # 182 done, ~ 71.36% Completed ~ 71.71% area (from TB 3.3)
 # 120 done, ~ 71.82% Completed ~ 72.21% area (from TB 3.2)
 # 178 done, ~ 72.27% Completed ~ 72.70% area (from TB 3.3)
 # 179 done, ~ 72.73% Completed ~ 73.09% area (from TB 3.3)
 # 177 done, ~ 73.18% Completed ~ 73.56% area (from TB 3.3)
 # 174 done, ~ 73.64% Completed ~ 73.67% area (from TB 3.3)
 # 171 done, ~ 74.09% Completed ~ 73.79% area (from TB 3.3)
 # 175 done, ~ 74.55% Completed ~ 74.16% area (from TB 3.3)
 # 173 done, ~ 75.00% Completed ~ 74.64% area (from TB 3.3)
 # 170 done, ~ 75.45% Completed ~ 75.11% area (from TB 3.3)
 # 172 done, ~ 75.91% Completed ~ 75.48% area (from TB 3.3)
 # 168 done, ~ 76.36% Completed ~ 75.59% area (from TB 3.3)
 # 169 done, ~ 76.82% Completed ~ 75.96% area (from TB 3.3)
 # 167 done, ~ 77.27% Completed ~ 76.44% area (from TB 3.3)
 # 166 done, ~ 77.73% Completed ~ 76.93% area (from TB 3.3)
 # 165 done, ~ 78.18% Completed ~ 77.42% area (from TB 3.3)
 # 164 done, ~ 78.64% Completed ~ 77.89% area (from TB 3.3)
 # 163 done, ~ 79.09% Completed ~ 78.37% area (from TB 3.3)
 # 162 done, ~ 79.55% Completed ~ 78.86% area (from TB 3.3)
 # 160 done, ~ 80.00% Completed ~ 79.33% area (from TB 3.3)
 # 161 done, ~ 80.45% Completed ~ 79.83% area (from TB 3.3)
 # 159 done, ~ 80.91% Completed ~ 80.30% area (from TB 3.3)
 # 158 done, ~ 81.36% Completed ~ 80.77% area (from TB 3.3)
 # 157 done, ~ 81.82% Completed ~ 81.23% area (from TB 3.3)
 # 156 done, ~ 82.27% Completed ~ 81.29% area (from TB 3.3)
 # 155 done, ~ 82.73% Completed ~ 81.78% area (from TB 3.3)
 # 154 done, ~ 83.18% Completed ~ 82.27% area (from TB 3.3)
 # 153 done, ~ 83.64% Completed ~ 82.77% area (from TB 3.3)
 # 152 done, ~ 84.09% Completed ~ 83.15% area (from TB 3.3)
 # 151 done, ~ 84.55% Completed ~ 83.43% area (from TB 3.3)
 # 150 done, ~ 85.00% Completed ~ 83.93% area (from TB 3.3)
 # 147 done, ~ 85.45% Completed ~ 84.42% area (from TB 3.3)
 # 146 done, ~ 85.91% Completed ~ 84.91% area (from TB 3.3)
 # 149 done, ~ 86.36% Completed ~ 85.40% area (from TB 3.3)
 # 148 done, ~ 86.82% Completed ~ 85.89% area (from TB 3.3)
 # 145 done, ~ 87.27% Completed ~ 86.38% area (from TB 3.3)
 server 23: ltgpe-05 exited: MSG_NUM 90 Total Times:User: 63.35
Sys: 0.10 Elapsed: 83.77 Memory: 76.500M
 server 14: ltgpe-05 exited: MSG_NUM 94 Total Times:User: 67.08
Sys: 0.12 Elapsed: 83.78 Memory: 82.989M
 server 15: ltgpe-05 exited: MSG_NUM 104 Total Times:User: 71.35
Sys: 0.09 Elapsed: 83.78 Memory: 83.266M
 # 144 done, ~ 87.73% Completed ~ 86.86% area (from TB 3.3)
 server 24: ltgpe-05 exited: MSG_NUM 92 Total Times:User: 58.57
Sys: 0.09 Elapsed: 83.80 Memory: 73.598M
```

```
 # 176 done, ~ 88.18% Completed ~ 87.33% area (from TB 3.3)
 server 1: ltgpe-03 exited: MSG_NUM 158 Total Times:User: 111.14
Sys: 0.13 Elapsed: 127.87 Memory: 91.200M
 server 16: ltgpe-05 exited: MSG_NUM 94 Total Times:User: 71.20
Sys: 0.09 Elapsed: 83.82 Memory: 83.267M
 # 135 done, ~ 88.64% Completed ~ 87.82% area (from TB 3.2)
 server 6: ltgpe-73 exited: MSG_NUM 66 Total Times:User: 68.77
Sys: 0.08 Elapsed: 87.97 Memory: 85.020M
 # 136 done, ~ 89.09% Completed ~ 88.32% area (from TB 3.2)
 # 137 done, ~ 89.55% Completed ~ 88.81% area (from TB 3.2)
 # 139 done, ~ 90.00% Completed ~ 89.30% area (from TB 3.2)
 server 11: ltgpe-73 exited: MSG_NUM 62 Total Times:User: 75.54
Sys: 0.08 Elapsed: 87.91 Memory: 90.954M
 # 131 done, ~ 90.45% Completed ~ 89.79% area (from TB 3.2)
 server 4: ltgpe-73 exited: MSG_NUM 69 Total Times:User: 68.34
Sys: 0.10 Elapsed: 88.01 Memory: 83.430M
 server 5: ltgpe-73 exited: MSG_NUM 74 Total Times:User: 70.50
Sys: 0.12 Elapsed: 88.01 Memory: 89.957M
 # 143 done, ~ 90.91% Completed ~ 90.26% area (from TB 3.2)
 # 140 done, ~ 91.36% Completed ~ 90.76% area (from TB 3.2)
 server 10: ltgpe-73 exited: MSG_NUM 62 Total Times:User: 62.63
Sys: 0.10 Elapsed: 87.92 Memory: 85.002M
 # 134 done, ~ 91.82% Completed ~ 91.25% area (from TB 3.2)
 # 132 done, ~ 92.27% Completed ~ 91.74% area (from TB 3.2)
 # 138 done, ~ 92.73% Completed ~ 92.23% area (from TB 3.2)
 # 128 done, ~ 93.18% Completed ~ 92.71% area (from TB 3.2)
 # 133 done, ~ 93.64% Completed ~ 93.20% area (from TB 3.2)
 # 130 done, ~ 94.09% Completed ~ 93.69% area (from TB 3.2)
 # 124 done, ~ 94.55% Completed ~ 94.18% area (from TB 3.2)
 # 129 done, ~ 95.00% Completed ~ 94.67% area (from TB 3.2)
 server 20: ltgpe-master2 exited: MSG_NUM 50 Total Times:User:
74.96 Sys: 0.11 Elapsed: 84.56 Memory: 79.861M
 server 25: ltgpe-05 exited: MSG_NUM 96 Total Times:User: 74.22
Sys: 0.08 Elapsed: 86.99 Memory: 74.017M
 # 142 done, ~ 95.45% Completed ~ 95.16% area (from TB 3.2)
 server 31: ltgpe-master1 exited: MSG_NUM 38 Total Times:User:
57.94 Sys: 0.13 Elapsed: 79.80 Memory: 73.069M
 server 7: ltgpe-73 exited: MSG_NUM 64 Total Times:User: 68.92
Sys: 0.10 Elapsed: 87.97 Memory: 91.655M
 server 9: ltgpe-73 exited: MSG_NUM 62 Total Times:User: 64.38
Sys: 0.19 Elapsed: 87.93 Memory: 86.714M
 server 19: ltgpe-master2 exited: MSG_NUM 52 Total Times:User:
59.77 Sys: 0.10 Elapsed: 85.38 Memory: 76.943M
 server 12: ltgpe-73 exited: MSG_NUM 66 Total Times:User: 75.91
Sys: 0.09 Elapsed: 87.90 Memory: 84.688M
 server 8: ltgpe-73 exited: MSG_NUM 64 Total Times:User: 63.03
Sys: 0.21 Elapsed: 87.93 Memory: 83.733M
 # 122 done, ~ 95.91% Completed ~ 95.63% area (from TB 3.2)
 # 125 done, ~ 96.36% Completed ~ 96.12% area (from TB 3.2)
 server 17: ltgpe-master2 exited: MSG_NUM 60 Total Times:User:
```

```
67.10 Sys: 0.12 Elapsed: 85.46 Memory: 80.136M
 server 21: ltgpe-master2 exited: MSG_NUM 52 Total Times:User:
74.19 Sys: 0.10 Elapsed: 84.51 Memory: 74.060M
 # 127 done, ~ 96.82% Completed ~ 96.62% area (from TB 3.2)
 # 141 done, ~ 97.27% Completed ~ 97.11% area (from TB 3.2)
 server 30: ltgpe-master1 exited: MSG_NUM 42 Total Times:User:
64.65 Sys: 0.09 Elapsed: 79.81 Memory: 73.409M
 # 123 done, ~ 97.73% Completed ~ 97.60% area (from TB 3.2)
 server 2: rutro-04 exited: MSG_NUM 62 Total Times:User: 77.56
Sys: 0.54 Elapsed: 89.02 Memory: 81.876M
 server 13: ltgpe-73 exited: MSG_NUM 66 Total Times:User: 70.60
Sys: 0.08 Elapsed: 87.90 Memory: 85.904M
 server 27: ltgpe-master1 exited: MSG_NUM 42 Total Times:User:
63.32 Sys: 0.13 Elapsed: 82.43 Memory: 69.790M
 # 126 done, ~ 98.18% Completed ~ 98.09% area (from TB 3.2)
 server 32: ltgpe-master1 exited: MSG_NUM 32 Total Times:User:
57.98 Sys: 0.26 Elapsed: 79.77 Memory: 73.439M
 server 28: ltgpe-master1 exited: MSG_NUM 35 Total Times:User:
56.28 Sys: 0.13 Elapsed: 79.98 Memory: 70.920M
 server 3: rutro-04 exited: MSG_NUM 64 Total Times:User: 65.14
Sys: 0.34 Elapsed: 88.91 Memory: 87.736M
 # 119 done, ~ 98.64% Completed ~ 98.56% area (from TB 3.2)
 server 29: ltgpe-master1 exited: MSG_NUM 42 Total Times:User:
57.11 Sys: 0.17 Elapsed: 79.88 Memory: 63.829M
 # 121 done, ~ 99.09% Completed ~ 99.05% area (from TB 3.2)
 server 22: ltgpe-master2 exited: MSG_NUM 46 Total Times:User:
66.48 Sys: 0.12 Elapsed: 84.47 Memory: 80.937M
 # 117 done, ~ 99.55% Completed ~ 99.52% area (from TB 3.2)
 # 118 done, ~ 100.00% Completed ~ 100.00% area (from TB 3.2)

 Final output file ./gds/TB1_out.gds is now complete, Wed Feb 12
11:54:10 2014


Prepare the holder cells for bound box data calculation for TB3.
 prepare holder cells: 10%
 prepare holder cells: 20%
 prepare holder cells: 30%
 prepare holder cells: 40%
 prepare holder cells: 50%
 prepare holder cells: 60%
 prepare holder cells: 70%
 prepare holder cells: 80%
 prepare holder cells: 90%
 prepare holder cells: Done Wed Feb 12 11:54:10 2014


Collecting cells bound box data for TB3.
 collect cell data: 10%
```

```
 collect cell data: 20%
 collect cell data: 30%
 collect cell data: 40%
 collect cell data: 50%
 collect cell data: 60%
 collect cell data: 70%
 collect cell data: 80%
 collect cell data: Done Wed Feb 12 11:54:10 2014


Finalizing OASIS output file ./gds/TB2_out.oas for TB3.
 finish output file: 10%
 finish output file: 20%
 finish output file: 30%
 finish output file: 40%
 finish output file: 50%
 finish output file: 60%
 finish output file: 70%
 finish output file: 80%
 finish output file: Done Wed Feb 12 11:54:10 2014


 Final output file ./gds/TB2_out.oas is now complete, Wed Feb 12
11:54:10 2014


 Final output file ./gds/MOF_BL_pcx2_out.gds is now complete, Wed
Feb 12 11:54:10 2014


Concurrent[3]: All data is present in the output for TB 3.
server 18: ltgpe-master2 exited: MSG_NUM 58 Total Times:User:
63.83 Sys: 0.11 Elapsed: 85.49 Memory: 79.797M
 server 26: ltgpe-master1 exited: MSG_NUM 46 Total Times:User:
57.22 Sys: 0.14 Elapsed: 82.67 Memory: 70.187M
 server 33: ltgpe-master1 exited: MSG_NUM 34 Total Times:User:
55.43 Sys: 0.18 Elapsed: 80.05 Memory: 75.589M

 Pipeline flow complete for TB 3, Wed Feb 12 11:54:10 2014

Template Generation Summary
 220 output templates (218 cluster templates) generated for TB1:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 84966.3180 square microns.
 Sum of all template bounding areas: 79470.0657 square microns.
 Sum of all ambit-biased template bounding areas: 93312.0891
square microns.
```

```
 220 output templates (218 cluster templates) generated for TB2:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 85200.1608 square microns.
 Sum of all template bounding areas: 78141.6062 square microns.
 Sum of all ambit-biased template bounding areas: 99135.4797
square microns.

 220 output templates (218 cluster templates) generated for TB3:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 85598.4280 square microns.
 Sum of all template bounding areas: 78760.9214 square microns.
 Sum of all ambit-biased template bounding areas: 87266.8274
square microns.


 Cleaning up intermediate recipe files at project finish
 rm -rf MOF_NONE_0210_TB1.pjx 2> /dev/null

 Cleaning up intermediate recipe files at project finish
 rm -rf MOF_NONE_0210_TB2.pjx 2> /dev/null

 Cleaning up intermediate recipe files at project finish
 rm -rf MOF_NONE_0210_TB3.pjx 2> /dev/null

 Cleaning up fragment directories at project finish
 rm -rf ./gds/TB1_out.gds.dir/ 2> /dev/null

 Pipeline flow complete for all template blocks, Wed Feb 12
11:54:10 2014

 TB3 Times:User: 4.62 Sys: 1.86 Elapsed: 137.32 Memory: 379.816M

Checking in PROTEUS_OPC...

Checking in PA...

 Server Processing Time Summary
 TB1 Elapsed: 0:00:49 [11:52:00-11:52:50]
 TB2 Elapsed: 0:01:12 [11:52:51-11:54:04]
 TB3 Elapsed: 0:00:05 [11:54:04-11:54:10]

 Total Times:User: 12.55 Sys: 2.80 Elapsed: 146.61 Memory: 379.816M
```

Log File Example 4

The following is an example of a log file from a proteus run with a `PROTEUS_JOB_FLOW` recipe with `PIPELINE_STRATEGY SINGLETONS`.

```
qrsh -P bhigh -V -cwd -now no -l model=EMT2700|EMT3000 -l
os_version=WS5.0 proteus -s snps:32 MOF_NONE_0210.pjx
/remote/ms_integ3_us03/SCM/DailyBuildReleases/proteus_G-
2012.09-8_12Feb14/amd64/bin/proteus Release G-2012.09-8 Revision
Proteus_G-2012.09-8_12Feb14-2919019 (64f/64m LINUX_X86_64).
host: machine1

 Proteus (TM) / PROTEUS (TM)
Version G-2012.09-8

*** Copyright (C) 1995 - 2014 Synopsys, Inc. ***
*** This software and the associated documentation are ***
*** confidential and proprietary to Synopsys, Inc. ***
*** Your use or disclosure of this software is subject to ***
*** the terms and conditions of a written license agreement ***
*** between you, or your company, and Synopsys, Inc. ***
*** ***

Testing for license PROTEUS_OPC...

Checking out PROTEUS_OPC...
License PROTEUS_OPC checked out.

Testing for license PA...

Checking out PA...
License PA checked out.
hierman -fe MOF_NONE_0210.pjx

 Hierarchy management started for job HIERMAN; Wed Feb 12 11:40:01
2014
Initialization completed.
     Times: User: 1.02 Sys: 0.02 Elapsed: 1.06 Memory: 17.543M

Reading input file /remote/ltg_pe1_us03/usr/large_2/
everest_0130_00.oas
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................

Scanning for Topcell(s)
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................
 Topcell: everest_gem
```

```
 native hierarchy: 1 cells; 0 refs
     Times: User: 0.09 Sys: 0.00 Elapsed: 0.09 Memory: 19.499M
Separating graphics from SREFs & AREFs.
 1 holder cell added
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.466M
Calculating clustering statistics.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.722M
Cleaning 1D AREF transforms.
Choosing Smart Block Compression cells.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.05 Sys: 0.00 Elapsed: 0.04 Memory: 19.501M
Create OASIS decompression buffer file.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.17 Sys: 0.00 Elapsed: 0.24 Memory: 19.732M
Partitioning large graphic cells (multi-threaded).
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
 1 native cell divided into 217 smaller graphic cells (20000 x
20000)
     Times: User: 1.82 Sys: 0.02 Elapsed: 1.84 Memory: 29.707M
Subdividing large AREFs.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.534M
Constructing framework for revised hierarchy.
 revised hierarchy: 219 cells; 218 refs
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.534M
Generating instances for processing.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
 219 instances (217 cluster instances) identified.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.01 Memory: 19.977M
Generating spatial bins for TopCell.
 Bin Count : 16 x 14 (horizontal x vertical)
 Bin Dimensions : (19590,19684) (x,y)
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.814M
Generating spatial bin cells.
 revised hierarchy: 444 cells; 1244 refs
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.859M
Generating flat bin instances.
 226 instances (224 cluster instances) identified.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.860M

Writing partial hierarchy results.
     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 19.661M
     Total Times: User: 3.15 Sys: 0.04 Elapsed: 3.32 Memory:
```

```
29.707M

 Hierarchy management complete for job HIERMAN; Wed Feb 12 11:40:03
2014
 Hierman FrontEnd Times:User: 7.85 Sys: 0.95 Elapsed: 9.53 Memory:
17.630M
 Cleaning up double patterning files from a previous run
 rm -rf ./temp/HIERMAN_TB1.DPT 2> /dev/null

 Cleaning up TB 1 fragment files and outputs from a previous run
 rm -rf ./gds/TB1_out.gds.dir/TB1/ 2> /dev/null
 rm -f HIERMAN_TB1_out.oas 2> /dev/null

 mkdir -p ./gds/TB1_out.gds.dir/TB1/ 2> /dev/null
 mkdir ./gds/TB1_out.gds.dir/TB1/TINF 2> /dev/null
 Cleaning up double patterning files from a previous run
 rm -rf ./temp/HIERMAN_TB2.DPT 2> /dev/null

 Cleaning up TB 2 fragment files and outputs from a previous run
 rm -rf ./gds/TB1_out.gds.dir/TB2/ 2> /dev/null
 rm -f HIERMAN_TB2_out.oas 2> /dev/null

 mkdir -p ./gds/TB1_out.gds.dir/TB2/ 2> /dev/null
 mkdir ./gds/TB1_out.gds.dir/TB2/TINF 2> /dev/null
 Cleaning up double patterning files from a previous run
 rm -rf ./temp/HIERMAN_TB3.DPT 2> /dev/null

 Cleaning up TB 3 fragment files and outputs from a previous run
 rm -f ./gds/TB1_out.gds.dir/* 2> /dev/null
 rm -f ./gds/TB1_out.gds 2> /dev/null
 rm -f ./gds/TB2_out.oas 2> /dev/null
 rm -f ./gds/MOF_BL_pcx2_out.gds 2> /dev/null
 rm -f ./gds/TB1_out.gds_saved 2> /dev/null
 rm -f ./gds/TB1_out.gds_tmp 2> /dev/null

 mkdir -p ./gds/TB1_out.gds.dir/ 2> /dev/null
 mkdir ./gds/TB1_out.gds.dir/TINF 2> /dev/null
 ./gds/TB1_out.gds output file created, Wed Feb 12 11:40:03 2014
 ./gds/TB2_out.oas output file created, Wed Feb 12 11:40:03 2014
 ./gds/MOF_BL_pcx2_out.gds output file created, Wed Feb 12
11:40:03 2014

 Correction job started for TB 1, Wed Feb 12 11:40:03 2014

 Generating templates from singleton instances.
 +0%----------+25%--------+50%--------+75%---------+100%
 ..................................................
 218 templates generated.
     Times: User: 0.05 Sys: 0.11 Elapsed: 0.22 Memory: 131.077M
```

```
trying to open port:2346
remote_server snps:32 2346 3 0 1
running remote_server for SGE access, in
/remote/us03home4/usr/bin/gridForProteus/remote_server
remote_server ltgpe-03 2346 3 1 1
sh -c "{ { dpserver -p2346 -cltgpe-03 -V3 >>logfile.txt.ltgpe-
03.18626; } 3>&1 1>&2 2>&3 |tee logfile.txt.ltgpe-03.18626; }
3>&1 1>&2 2>&3"
qsub -P iheavy -V -cwd -N dpsvr.18528 -j y -o ./logfiles.18528 -
t 1-32 .dpserver.18528
Requesting 32 dpservers on snps...
Grid job number: 472424
dpserver log files are in logfiles.18528
Calculating context for leaf cells.
 leaf context: 10%
 leaf context: 20%
 leaf context: 30%
 leaf context: 40%
 leaf context: 50%
 leaf context: 60%
 leaf context: 70%
 leaf context: 80%
 leaf context: Done Wed Feb 12 11:40:05 2014

     Times: User: 0.04 Sys: 0.04 Elapsed: 2.08 Memory: 136.485M
Calculating context for holding cells.
 holder context: 10%
 holder context: 20%
 holder context: 30%
 holder context: 40%
 holder context: 50%
 holder context: 60%
 holder context: 70%
 holder context: 80%
 holder context: 90%
 holder context: Done Wed Feb 12 11:40:05 2014

     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 131.871M
Generating output files for correction.
 output files: 10%
 output files: 20%
 output files: 30%
 output files: 60%
 output files: 70%
 output files: 80%
 output files: 90%
 output files: Done Wed Feb 12 11:40:05 2014

     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 131.843M
```

```
 220 output templates (218 cluster templates) generated for TB1:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 84966.3180 square microns.
 Sum of all template bounding areas: 79470.0657 square microns.
 Sum of all ambit-biased template bounding areas: 93312.0891
square microns.

 Hierman BackEnd Times: User: 0.06 Sys: 0.06 Elapsed: 2.13 Memory:
136.485M
Client: executing mktop in a separate thread...
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 0.91% Completed ~ 0.00% area (of TB 1)
Client: 1:1:INIT :N_TMPL=220 SOCK_CNT=1 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 217 done, ~ 1.36% Completed ~ 0.43% area (of TB 1)
 # 216 done, ~ 1.82% Completed ~ 0.89% area (of TB 1)
 # 215 done, ~ 2.27% Completed ~ 1.35% area (of TB 1)
 # 214 done, ~ 2.73% Completed ~ 1.80% area (of TB 1)
 # 213 done, ~ 3.18% Completed ~ 2.26% area (of TB 1)
 # 212 done, ~ 3.64% Completed ~ 2.72% area (of TB 1)
 # 211 done, ~ 4.09% Completed ~ 3.18% area (of TB 1)
 # 210 done, ~ 4.55% Completed ~ 3.64% area (of TB 1)
 # 209 done, ~ 5.00% Completed ~ 4.10% area (of TB 1)
 # 208 done, ~ 5.45% Completed ~ 4.55% area (of TB 1)
 # 207 done, ~ 5.91% Completed ~ 5.01% area (of TB 1)
 # 206 done, ~ 6.36% Completed ~ 5.47% area (of TB 1)
 # 205 done, ~ 6.82% Completed ~ 5.93% area (of TB 1)
 # 204 done, ~ 7.27% Completed ~ 6.36% area (of TB 1)
 # 203 done, ~ 7.73% Completed ~ 6.82% area (of TB 1)
 # 202 done, ~ 8.18% Completed ~ 7.31% area (of TB 1)
 # 201 done, ~ 8.64% Completed ~ 7.80% area (of TB 1)
 # 200 done, ~ 9.09% Completed ~ 8.29% area (of TB 1)
 # 199 done, ~ 9.55% Completed ~ 8.78% area (of TB 1)
 # 198 done, ~ 10.00% Completed ~ 9.27% area (of TB 1)
 # 197 done, ~ 10.45% Completed ~ 9.76% area (of TB 1)
 # 196 done, ~ 10.91% Completed ~ 10.25% area (of TB 1)
 # 195 done, ~ 11.36% Completed ~ 10.74% area (of TB 1)
 # 194 done, ~ 11.82% Completed ~ 11.23% area (of TB 1)
 # 193 done, ~ 12.27% Completed ~ 11.71% area (of TB 1)
 # 192 done, ~ 12.73% Completed ~ 12.20% area (of TB 1)
 # 191 done, ~ 13.18% Completed ~ 12.69% area (of TB 1)
 # 190 done, ~ 13.64% Completed ~ 13.15% area (of TB 1)
 # 189 done, ~ 14.09% Completed ~ 13.61% area (of TB 1)
 # 188 done, ~ 14.55% Completed ~ 14.10% area (of TB 1)
```

```
 # 187 done, ~ 15.00% Completed ~ 14.59% area (of TB 1)
 # 186 done, ~ 15.45% Completed ~ 15.08% area (of TB 1)
 # 185 done, ~ 15.91% Completed ~ 15.57% area (of TB 1)
 # 184 done, ~ 16.36% Completed ~ 16.06% area (of TB 1)
 # 183 done, ~ 16.82% Completed ~ 16.55% area (of TB 1)
 # 182 done, ~ 17.27% Completed ~ 17.04% area (of TB 1)
 # 181 done, ~ 17.73% Completed ~ 17.53% area (of TB 1)
 # 180 done, ~ 18.18% Completed ~ 18.02% area (of TB 1)
 # 179 done, ~ 18.64% Completed ~ 18.51% area (of TB 1)
 # 178 done, ~ 19.09% Completed ~ 19.00% area (of TB 1)
 # 177 done, ~ 19.55% Completed ~ 19.49% area (of TB 1)
 # 176 done, ~ 20.00% Completed ~ 19.95% area (of TB 1)
 # 175 done, ~ 20.45% Completed ~ 20.41% area (of TB 1)
 # 174 done, ~ 20.91% Completed ~ 20.90% area (of TB 1)
 # 173 done, ~ 21.36% Completed ~ 21.39% area (of TB 1)
 # 172 done, ~ 21.82% Completed ~ 21.87% area (of TB 1)
 # 171 done, ~ 22.27% Completed ~ 22.36% area (of TB 1)
Client: 2:1:INIT :N_TMPL=220 SOCK_CNT=2 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 3:1:INIT :N_TMPL=220 SOCK_CNT=3 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 4:1:INIT :N_TMPL=220 SOCK_CNT=4 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 5:1:INIT :N_TMPL=220 SOCK_CNT=5 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 6:1:INIT :N_TMPL=220 SOCK_CNT=6 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 7:1:INIT :N_TMPL=220 SOCK_CNT=7 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 8:1:INIT :N_TMPL=220 SOCK_CNT=8 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client: 9:1:INIT :N_TMPL=220 SOCK_CNT=9 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 170 done, ~ 22.73% Completed ~ 22.85% area (of TB 1)
Client:10:1:INIT :N_TMPL=220 SOCK_CNT=10 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:11:1:INIT :N_TMPL=220 SOCK_CNT=11 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:12:1:INIT :N_TMPL=220 SOCK_CNT=12 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
```

```
LASTTB=3
Client:13:1:INIT :N_TMPL=220 SOCK_CNT=13 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:14:1:INIT :N_TMPL=220 SOCK_CNT=14 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:15:1:INIT :N_TMPL=220 SOCK_CNT=15 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:16:1:INIT :N_TMPL=220 SOCK_CNT=16 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:17:1:INIT :N_TMPL=220 SOCK_CNT=17 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:18:1:INIT :N_TMPL=220 SOCK_CNT=18 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:19:1:INIT :N_TMPL=220 SOCK_CNT=19 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:20:1:INIT :N_TMPL=220 SOCK_CNT=20 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 169 done, ~ 23.18% Completed ~ 23.34% area (of TB 1)
 # 168 done, ~ 23.64% Completed ~ 23.83% area (of TB 1)
 # 161 done, ~ 24.09% Completed ~ 24.29% area (of TB 1)
 # 160 done, ~ 24.55% Completed ~ 24.40% area (of TB 1)
 # 163 done, ~ 25.00% Completed ~ 24.89% area (of TB 1)
 # 167 done, ~ 25.45% Completed ~ 25.37% area (of TB 1)
 # 164 done, ~ 25.91% Completed ~ 25.86% area (of TB 1)
 # 162 done, ~ 26.36% Completed ~ 26.32% area (of TB 1)
 # 165 done, ~ 26.82% Completed ~ 26.81% area (of TB 1)
 # 166 done, ~ 27.27% Completed ~ 27.30% area (of TB 1)
 # 159 done, ~ 27.73% Completed ~ 27.66% area (of TB 1)
 # 154 done, ~ 28.18% Completed ~ 28.15% area (of TB 1)
 # 146 done, ~ 28.64% Completed ~ 28.25% area (of TB 1)
 # 147 done, ~ 29.09% Completed ~ 28.71% area (of TB 1)
 # 156 done, ~ 29.55% Completed ~ 29.20% area (of TB 1)
 # 153 done, ~ 30.00% Completed ~ 29.69% area (of TB 1)
 # 157 done, ~ 30.45% Completed ~ 30.18% area (of TB 1)
 # 155 done, ~ 30.91% Completed ~ 30.67% area (of TB 1)
 # 148 done, ~ 31.36% Completed ~ 31.13% area (of TB 1)
 # 158 done, ~ 31.82% Completed ~ 31.62% area (of TB 1)
 # 151 done, ~ 32.27% Completed ~ 32.11% area (of TB 1)
 # 132 done, ~ 32.73% Completed ~ 32.20% area (of TB 1)
 # 140 done, ~ 33.18% Completed ~ 32.69% area (of TB 1)
 # 152 done, ~ 33.64% Completed ~ 33.18% area (of TB 1)
 # 149 done, ~ 34.09% Completed ~ 33.67% area (of TB 1)
```

```
# 145 done, ~ 34.55% Completed ~ 34.03% area (of TB 1)
# 150 done, ~ 35.00% Completed ~ 34.52% area (of TB 1)
# 139 done, ~ 35.45% Completed ~ 35.01% area (of TB 1)
# 142 done, ~ 35.91% Completed ~ 35.50% area (of TB 1)
# 143 done, ~ 36.36% Completed ~ 35.99% area (of TB 1)
# 144 done, ~ 36.82% Completed ~ 36.48% area (of TB 1)
# 141 done, ~ 37.27% Completed ~ 36.97% area (of TB 1)
# 133 done, ~ 37.73% Completed ~ 37.43% area (of TB 1)
# 138 done, ~ 38.18% Completed ~ 37.92% area (of TB 1)
# 137 done, ~ 38.64% Completed ~ 38.41% area (of TB 1)
# 118 done, ~ 39.09% Completed ~ 38.51% area (of TB 1)
# 134 done, ~ 39.55% Completed ~ 38.97% area (of TB 1)
# 131 done, ~ 40.00% Completed ~ 39.33% area (of TB 1)
# 117 done, ~ 40.45% Completed ~ 39.53% area (of TB 1)
# 136 done, ~ 40.91% Completed ~ 40.02% area (of TB 1)
# 126 done, ~ 41.36% Completed ~ 40.51% area (of TB 1)
# 135 done, ~ 41.82% Completed ~ 41.00% area (of TB 1)
# 128 done, ~ 42.27% Completed ~ 41.49% area (of TB 1)
# 127 done, ~ 42.73% Completed ~ 41.98% area (of TB 1)
# 116 done, ~ 43.18% Completed ~ 42.26% area (of TB 1)
# 129 done, ~ 43.64% Completed ~ 42.75% area (of TB 1)
# 115 done, ~ 44.09% Completed ~ 43.20% area (of TB 1)
# 119 done, ~ 44.55% Completed ~ 43.66% area (of TB 1)
# 120 done, ~ 45.00% Completed ~ 44.12% area (of TB 1)
# 112 done, ~ 45.45% Completed ~ 44.61% area (of TB 1)
# 130 done, ~ 45.91% Completed ~ 45.10% area (of TB 1)
# 114 done, ~ 46.36% Completed ~ 45.58% area (of TB 1)
# 125 done, ~ 46.82% Completed ~ 46.07% area (of TB 1)
# 124 done, ~ 47.27% Completed ~ 46.56% area (of TB 1)
# 113 done, ~ 47.73% Completed ~ 47.05% area (of TB 1)
# 121 done, ~ 48.18% Completed ~ 47.54% area (of TB 1)
# 123 done, ~ 48.64% Completed ~ 48.03% area (of TB 1)
# 104 done, ~ 49.09% Completed ~ 48.52% area (of TB 1)
# 122 done, ~ 49.55% Completed ~ 49.01% area (of TB 1)
# 111 done, ~ 50.00% Completed ~ 49.50% area (of TB 1)
# 105 done, ~ 50.45% Completed ~ 49.95% area (of TB 1)
# 110 done, ~ 50.91% Completed ~ 50.44% area (of TB 1)
# 106 done, ~ 51.36% Completed ~ 50.90% area (of TB 1)
# 108 done, ~ 51.82% Completed ~ 51.39% area (of TB 1)
# 109 done, ~ 52.27% Completed ~ 51.88% area (of TB 1)
# 100 done, ~ 52.73% Completed ~ 52.34% area (of TB 1)
# 103 done, ~ 53.18% Completed ~ 52.71% area (of TB 1)
# 107 done, ~ 53.64% Completed ~ 53.20% area (of TB 1)
# 101 done, ~ 54.09% Completed ~ 53.66% area (of TB 1)
# 98 done, ~ 54.55% Completed ~ 54.15% area (of TB 1)
# 102 done, ~ 55.00% Completed ~ 54.52% area (of TB 1)
# 99 done, ~ 55.45% Completed ~ 55.01% area (of TB 1)
# 91 done, ~ 55.91% Completed ~ 55.47% area (of TB 1)
```

```
 # 97 done, ~ 56.36% Completed ~ 55.96% area (of TB 1)
 # 96 done, ~ 56.82% Completed ~ 56.45% area (of TB 1)
 # 94 done, ~ 57.27% Completed ~ 56.94% area (of TB 1)
 # 95 done, ~ 57.73% Completed ~ 57.43% area (of TB 1)
 # 86 done, ~ 58.18% Completed ~ 57.92% area (of TB 1)
 # 90 done, ~ 58.64% Completed ~ 58.41% area (of TB 1)
 # 92 done, ~ 59.09% Completed ~ 58.87% area (of TB 1)
 # 85 done, ~ 59.55% Completed ~ 59.36% area (of TB 1)
 # 93 done, ~ 60.00% Completed ~ 59.85% area (of TB 1)
 # 88 done, ~ 60.45% Completed ~ 60.34% area (of TB 1)
 # 87 done, ~ 60.91% Completed ~ 60.83% area (of TB 1)
 # 89 done, ~ 61.36% Completed ~ 61.32% area (of TB 1)
 # 77 done, ~ 61.82% Completed ~ 61.78% area (of TB 1)
 # 84 done, ~ 62.27% Completed ~ 62.26% area (of TB 1)
 # 80 done, ~ 62.73% Completed ~ 62.75% area (of TB 1)
Client:21:1:INIT :N_TMPL=220 SOCK_CNT=21 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 82 done, ~ 63.18% Completed ~ 63.24% area (of TB 1)
 # 76 done, ~ 63.64% Completed ~ 63.73% area (of TB 1)
 # 83 done, ~ 64.09% Completed ~ 64.22% area (of TB 1)
 # 72 done, ~ 64.55% Completed ~ 64.71% area (of TB 1)
 # 81 done, ~ 65.00% Completed ~ 65.20% area (of TB 1)
 # 78 done, ~ 65.45% Completed ~ 65.66% area (of TB 1)
 # 79 done, ~ 65.91% Completed ~ 66.15% area (of TB 1)
 # 74 done, ~ 66.36% Completed ~ 66.64% area (of TB 1)
 # 63 done, ~ 66.82% Completed ~ 67.10% area (of TB 1)
 # 75 done, ~ 67.27% Completed ~ 67.59% area (of TB 1)
Client:22:1:INIT :N_TMPL=220 SOCK_CNT=22 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:23:1:INIT :N_TMPL=220 SOCK_CNT=23 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 73 done, ~ 67.73% Completed ~ 68.08% area (of TB 1)
 # 62 done, ~ 68.18% Completed ~ 68.57% area (of TB 1)
Client:24:1:INIT :N_TMPL=220 SOCK_CNT=24 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 71 done, ~ 68.64% Completed ~ 69.06% area (of TB 1)
 # 58 done, ~ 69.09% Completed ~ 69.55% area (of TB 1)
Client:25:1:INIT :N_TMPL=220 SOCK_CNT=25 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:26:1:INIT :N_TMPL=220 SOCK_CNT=26 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 66 done, ~ 69.55% Completed ~ 70.04% area (of TB 1)
Client:27:1:INIT :N_TMPL=220 SOCK_CNT=27 JCF=MOF_NONE_0210.pjx
```

```
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 70 done, ~ 70.00% Completed ~ 70.53% area (of TB 1)
 # 68 done, ~ 70.45% Completed ~ 71.02% area (of TB 1)
 # 46 done, ~ 70.91% Completed ~ 71.05% area (of TB 1)
 # 64 done, ~ 71.36% Completed ~ 71.51% area (of TB 1)
 # 61 done, ~ 71.82% Completed ~ 72.00% area (of TB 1)
 # 69 done, ~ 72.27% Completed ~ 72.49% area (of TB 1)
 # 65 done, ~ 72.73% Completed ~ 72.98% area (of TB 1)
 # 48 done, ~ 73.18% Completed ~ 73.47% area (of TB 1)
 # 45 done, ~ 73.64% Completed ~ 73.52% area (of TB 1)
 # 52 done, ~ 74.09% Completed ~ 74.01% area (of TB 1)
 # 59 done, ~ 74.55% Completed ~ 74.50% area (of TB 1)
 # 67 done, ~ 75.00% Completed ~ 74.99% area (of TB 1)
 # 47 done, ~ 75.45% Completed ~ 75.48% area (of TB 1)
 # 49 done, ~ 75.91% Completed ~ 75.94% area (of TB 1)
 # 60 done, ~ 76.36% Completed ~ 76.43% area (of TB 1)
 # 50 done, ~ 76.82% Completed ~ 76.88% area (of TB 1)
 # 51 done, ~ 77.27% Completed ~ 77.37% area (of TB 1)
 # 57 done, ~ 77.73% Completed ~ 77.86% area (of TB 1)
 # 44 done, ~ 78.18% Completed ~ 78.35% area (of TB 1)
 # 55 done, ~ 78.64% Completed ~ 78.84% area (of TB 1)
 # 38 done, ~ 79.09% Completed ~ 79.33% area (of TB 1)
 # 32 done, ~ 79.55% Completed ~ 79.58% area (of TB 1)
 # 33 done, ~ 80.00% Completed ~ 79.88% area (of TB 1)
 # 35 done, ~ 80.45% Completed ~ 80.34% area (of TB 1)
 # 53 done, ~ 80.91% Completed ~ 80.83% area (of TB 1)
 # 56 done, ~ 81.36% Completed ~ 81.32% area (of TB 1)
 # 54 done, ~ 81.82% Completed ~ 81.81% area (of TB 1)
 # 34 done, ~ 82.27% Completed ~ 82.30% area (of TB 1)
 # 37 done, ~ 82.73% Completed ~ 82.79% area (of TB 1)
 # 43 done, ~ 83.18% Completed ~ 83.27% area (of TB 1)
 # 26 done, ~ 83.64% Completed ~ 83.76% area (of TB 1)
 # 36 done, ~ 84.09% Completed ~ 84.22% area (of TB 1)
 # 21 done, ~ 84.55% Completed ~ 84.52% area (of TB 1)
 # 20 done, ~ 85.00% Completed ~ 84.77% area (of TB 1)
 # 42 done, ~ 85.45% Completed ~ 85.26% area (of TB 1)
 # 31 done, ~ 85.91% Completed ~ 85.75% area (of TB 1)
 # 39 done, ~ 86.36% Completed ~ 86.24% area (of TB 1)
 # 40 done, ~ 86.82% Completed ~ 86.73% area (of TB 1)
 # 25 done, ~ 87.27% Completed ~ 87.22% area (of TB 1)
 # 41 done, ~ 87.73% Completed ~ 87.71% area (of TB 1)
 # 23 done, ~ 88.18% Completed ~ 88.17% area (of TB 1)
 # 22 done, ~ 88.64% Completed ~ 88.65% area (of TB 1)
 # 14 done, ~ 89.09% Completed ~ 89.14% area (of TB 1)
 # 29 done, ~ 89.55% Completed ~ 89.63% area (of TB 1)
 # 19 done, ~ 90.00% Completed ~ 90.12% area (of TB 1)
 # 30 done, ~ 90.45% Completed ~ 90.61% area (of TB 1)
```

```
 # 8 done, ~ 90.91% Completed ~ 90.85% area (of TB 1)
 # 28 done, ~ 91.36% Completed ~ 91.34% area (of TB 1)
 # 24 done, ~ 91.82% Completed ~ 91.79% area (of TB 1)
Client:28:1:INIT :N_TMPL=220 SOCK_CNT=28 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 27 done, ~ 92.27% Completed ~ 92.28% area (of TB 1)
 # 9 done, ~ 92.73% Completed ~ 92.56% area (of TB 1)
Client:29:1:INIT :N_TMPL=220 SOCK_CNT=29 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:30:1:INIT :N_TMPL=220 SOCK_CNT=30 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
Client:31:1:INIT :N_TMPL=220 SOCK_CNT=31 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 11 done, ~ 93.18% Completed ~ 92.99% area (of TB 1)
 # 18 done, ~ 93.64% Completed ~ 93.48% area (of TB 1)
Client:32:1:INIT :N_TMPL=220 SOCK_CNT=32 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 16 done, ~ 94.09% Completed ~ 93.97% area (of TB 1)
 # 10 done, ~ 94.55% Completed ~ 94.43% area (of TB 1)
 # 15 done, ~ 95.00% Completed ~ 94.92% area (of TB 1)
Client:33:1:INIT :N_TMPL=220 SOCK_CNT=33 JCF=MOF_NONE_0210.pjx
DPROTEUS_CFG_FILE=/remote/us03home4/usr/bin/dproteus.cfg TB=1
LASTTB=3
 # 2 done, ~ 95.45% Completed ~ 95.38% area (of TB 1)
 # 7 done, ~ 95.91% Completed ~ 95.84% area (of TB 1)
 # 5 done, ~ 96.36% Completed ~ 96.30% area (of TB 1)
 # 17 done, ~ 96.82% Completed ~ 96.79% area (of TB 1)
 # 12 done, ~ 97.27% Completed ~ 97.25% area (of TB 1)
 # 13 done, ~ 97.73% Completed ~ 97.73% area (of TB 1)
 # 4 done, ~ 98.18% Completed ~ 98.19% area (of TB 1)
 # 3 done, ~ 98.64% Completed ~ 98.65% area (of TB 1)
 # 6 done, ~ 99.09% Completed ~ 99.11% area (of TB 1)
 # 1 done, ~ 99.55% Completed ~ 99.57% area (of TB 1)
 # 0 done, ~ 100.00% Completed ~ 100.00% area (of TB 1)

Concurrent[1]: All data is present in the output for TB 1.

Correction job complete for TB 1, Wed Feb 12 11:40:59 2014

 TB1 Times:User: 0.25 Sys: 0.24 Elapsed: 56.36 Memory: 141.081M

 Correction job started for TB 2, Wed Feb 12 11:40:59 2014

 Scanning cell graphics from previous template block.
```

```
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.18 Sys: 0.14 Elapsed: 0.16 Memory: 77.694M
 Performing Overflow Processing for TB 2.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
     Times: User: 0.08 Sys: 0.05 Elapsed: 0.07 Memory: 77.873M
 Generating templates from singleton instances.
 +0%----------+25%--------+50%--------+75%---------+100%
 ....................................................
 218 templates generated.
     Times: User: 0.05 Sys: 0.07 Elapsed: 0.07 Memory: 185.866M
 # 195 done, ~ 0.46% Completed ~ 0.10% area (of TB 2)
 # 192 done, ~ 0.92% Completed ~ 0.55% area (of TB 2)
 # 199 done, ~ 1.38% Completed ~ 0.65% area (of TB 2)
 # 203 done, ~ 1.83% Completed ~ 0.74% area (of TB 2)
 # 207 done, ~ 2.29% Completed ~ 0.84% area (of TB 2)
 # 189 done, ~ 2.75% Completed ~ 0.88% area (of TB 2)
 # 191 done, ~ 3.21% Completed ~ 1.33% area (of TB 2)
 # 193 done, ~ 3.67% Completed ~ 1.76% area (of TB 2)
 # 194 done, ~ 4.13% Completed ~ 2.19% area (of TB 2)
 # 208 done, ~ 4.59% Completed ~ 2.34% area (of TB 2)
Calculating context for leaf cells.
 leaf context: 10%
 leaf context: 20%
 leaf context: 30%
 leaf context: 40%
 leaf context: 50%
 leaf context: 60%
 leaf context: 70%
 leaf context: 80%
 leaf context: Done Wed Feb 12 11:41:02 2014

     Times: User: 0.07 Sys: 0.06 Elapsed: 2.07 Memory: 190.670M
Calculating context for holding cells.
 holder context: 10%
 holder context: 20%
 holder context: 30%
 holder context: 40%
 holder context: 50%
 holder context: 60%
 holder context: 70%
 holder context: 80%
 holder context: 90%
 holder context: Done Wed Feb 12 11:41:02 2014

     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 186.014M
Generating output files for correction.
```

```
 output files: 10%
 output files: 20%
 output files: 30%
 output files: 60%
 output files: 70%
 output files: 80%
 output files: 90%
 output files: Done Wed Feb 12 11:41:02 2014

     Times: User: 0.01 Sys: 0.00 Elapsed: 0.00 Memory: 185.980M

 220 output templates (218 cluster templates) generated for TB2:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 85200.1608 square microns.
 Sum of all template bounding areas: 78141.6062 square microns.
 Sum of all ambit-biased template bounding areas: 99135.4797
square microns.

 Hierman BackEnd Times: User: 0.39 Sys: 0.32 Elapsed: 2.40 Memory:
190.743M
Client: executing mktop in a separate thread...
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 5.45% Completed ~ 2.55% area (of TB 2)
 # 217 done, ~ 5.91% Completed ~ 3.05% area (of TB 2)
 # 214 done, ~ 6.36% Completed ~ 3.51% area (of TB 2)
 # 216 done, ~ 6.82% Completed ~ 4.01% area (of TB 2)
 # 187 done, ~ 7.27% Completed ~ 4.50% area (of TB 2)
 # 215 done, ~ 7.73% Completed ~ 5.00% area (of TB 2)
 # 188 done, ~ 8.18% Completed ~ 5.49% area (of TB 2)
 # 198 done, ~ 8.64% Completed ~ 5.96% area (of TB 2)
 # 209 done, ~ 9.09% Completed ~ 6.25% area (of TB 2)
 # 172 done, ~ 9.55% Completed ~ 6.74% area (of TB 2)
 # 190 done, ~ 10.00% Completed ~ 7.24% area (of TB 2)
 # 186 done, ~ 10.45% Completed ~ 7.70% area (of TB 2)
 # 183 done, ~ 10.91% Completed ~ 8.20% area (of TB 2)
 # 202 done, ~ 11.36% Completed ~ 8.67% area (of TB 2)
 # 206 done, ~ 11.82% Completed ~ 9.14% area (of TB 2)
 # 182 done, ~ 12.27% Completed ~ 9.61% area (of TB 2)
 # 173 done, ~ 12.73% Completed ~ 10.08% area (of TB 2)
 # 175 done, ~ 13.18% Completed ~ 10.57% area (of TB 2)
 # 174 done, ~ 13.64% Completed ~ 11.04% area (of TB 2)
 # 178 done, ~ 14.09% Completed ~ 11.51% area (of TB 2)
 # 204 done, ~ 14.55% Completed ~ 11.88% area (of TB 2)
 # 200 done, ~ 15.00% Completed ~ 12.25% area (of TB 2)
 # 196 done, ~ 15.45% Completed ~ 12.61% area (of TB 2)
```

```
# 179 done, ~ 15.91% Completed ~ 13.11% area (of TB 2)
# 169 done, ~ 16.36% Completed ~ 13.40% area (of TB 2)
# 166 done, ~ 16.82% Completed ~ 13.70% area (of TB 2)
# 161 done, ~ 17.27% Completed ~ 14.01% area (of TB 2)
# 163 done, ~ 17.73% Completed ~ 14.48% area (of TB 2)
# 162 done, ~ 18.18% Completed ~ 14.97% area (of TB 2)
# 210 done, ~ 18.64% Completed ~ 15.44% area (of TB 2)
# 213 done, ~ 19.09% Completed ~ 15.94% area (of TB 2)
# 171 done, ~ 19.55% Completed ~ 16.32% area (of TB 2)
# 168 done, ~ 20.00% Completed ~ 16.79% area (of TB 2)
# 167 done, ~ 20.45% Completed ~ 17.24% area (of TB 2)
# 155 done, ~ 20.91% Completed ~ 17.74% area (of TB 2)
# 164 done, ~ 21.36% Completed ~ 18.21% area (of TB 2)
# 165 done, ~ 21.82% Completed ~ 18.70% area (of TB 2)
# 211 done, ~ 22.27% Completed ~ 19.20% area (of TB 2)
# 212 done, ~ 22.73% Completed ~ 19.69% area (of TB 2)
# 170 done, ~ 23.18% Completed ~ 20.08% area (of TB 2)
# 146 done, ~ 23.64% Completed ~ 20.58% area (of TB 2)
# 148 done, ~ 24.09% Completed ~ 21.07% area (of TB 2)
# 147 done, ~ 24.55% Completed ~ 21.57% area (of TB 2)
# 184 done, ~ 25.00% Completed ~ 22.06% area (of TB 2)
# 180 done, ~ 25.45% Completed ~ 22.56% area (of TB 2)
# 197 done, ~ 25.91% Completed ~ 23.05% area (of TB 2)
# 145 done, ~ 26.36% Completed ~ 23.55% area (of TB 2)
# 143 done, ~ 26.82% Completed ~ 23.83% area (of TB 2)
# 201 done, ~ 27.27% Completed ~ 24.33% area (of TB 2)
# 205 done, ~ 27.73% Completed ~ 24.82% area (of TB 2)
# 158 done, ~ 28.18% Completed ~ 25.32% area (of TB 2)
# 144 done, ~ 28.64% Completed ~ 25.78% area (of TB 2)
# 159 done, ~ 29.09% Completed ~ 26.28% area (of TB 2)
# 176 done, ~ 29.55% Completed ~ 26.77% area (of TB 2)
# 185 done, ~ 30.00% Completed ~ 27.27% area (of TB 2)
# 157 done, ~ 30.45% Completed ~ 27.76% area (of TB 2)
# 177 done, ~ 30.91% Completed ~ 28.26% area (of TB 2)
# 181 done, ~ 31.36% Completed ~ 28.75% area (of TB 2)
# 153 done, ~ 31.82% Completed ~ 29.25% area (of TB 2)
# 154 done, ~ 32.27% Completed ~ 29.75% area (of TB 2)
# 149 done, ~ 32.73% Completed ~ 30.24% area (of TB 2)
# 160 done, ~ 33.18% Completed ~ 30.74% area (of TB 2)
# 140 done, ~ 33.64% Completed ~ 31.24% area (of TB 2)
# 152 done, ~ 34.09% Completed ~ 31.73% area (of TB 2)
# 113 done, ~ 34.55% Completed ~ 31.78% area (of TB 2)
# 156 done, ~ 35.00% Completed ~ 32.28% area (of TB 2)
# 114 done, ~ 35.45% Completed ~ 32.74% area (of TB 2)
# 134 done, ~ 35.91% Completed ~ 33.24% area (of TB 2)
# 121 done, ~ 36.36% Completed ~ 33.73% area (of TB 2)
# 151 done, ~ 36.82% Completed ~ 34.23% area (of TB 2)
# 124 done, ~ 37.27% Completed ~ 34.72% area (of TB 2)
```

```
# 128 done, ~ 37.73% Completed ~ 35.22% area (of TB 2)
# 150 done, ~ 38.18% Completed ~ 35.71% area (of TB 2)
# 120 done, ~ 38.64% Completed ~ 36.21% area (of TB 2)
# 123 done, ~ 39.09% Completed ~ 36.70% area (of TB 2)
# 122 done, ~ 39.55% Completed ~ 37.20% area (of TB 2)
# 142 done, ~ 40.00% Completed ~ 37.70% area (of TB 2)
# 108 done, ~ 40.45% Completed ~ 38.19% area (of TB 2)
# 141 done, ~ 40.91% Completed ~ 38.69% area (of TB 2)
# 115 done, ~ 41.36% Completed ~ 39.18% area (of TB 2)
# 119 done, ~ 41.82% Completed ~ 39.68% area (of TB 2)
# 137 done, ~ 42.27% Completed ~ 40.17% area (of TB 2)
# 139 done, ~ 42.73% Completed ~ 40.67% area (of TB 2)
# 138 done, ~ 43.18% Completed ~ 41.17% area (of TB 2)
# 102 done, ~ 43.64% Completed ~ 41.66% area (of TB 2)
# 116 done, ~ 44.09% Completed ~ 42.16% area (of TB 2)
# 117 done, ~ 44.55% Completed ~ 42.65% area (of TB 2)
# 109 done, ~ 45.00% Completed ~ 43.15% area (of TB 2)
# 118 done, ~ 45.45% Completed ~ 43.64% area (of TB 2)
# 136 done, ~ 45.91% Completed ~ 44.14% area (of TB 2)
# 135 done, ~ 46.36% Completed ~ 44.63% area (of TB 2)
# 129 done, ~ 46.82% Completed ~ 45.13% area (of TB 2)
# 106 done, ~ 47.27% Completed ~ 45.63% area (of TB 2)
# 112 done, ~ 47.73% Completed ~ 46.12% area (of TB 2)
# 133 done, ~ 48.18% Completed ~ 46.62% area (of TB 2)
# 104 done, ~ 48.64% Completed ~ 47.11% area (of TB 2)
# 96 done, ~ 49.09% Completed ~ 47.61% area (of TB 2)
# 125 done, ~ 49.55% Completed ~ 48.10% area (of TB 2)
# 131 done, ~ 50.00% Completed ~ 48.60% area (of TB 2)
# 127 done, ~ 50.45% Completed ~ 49.09% area (of TB 2)
# 97 done, ~ 50.91% Completed ~ 49.59% area (of TB 2)
# 111 done, ~ 51.36% Completed ~ 50.08% area (of TB 2)
# 130 done, ~ 51.82% Completed ~ 50.58% area (of TB 2)
# 105 done, ~ 52.27% Completed ~ 51.07% area (of TB 2)
# 126 done, ~ 52.73% Completed ~ 51.57% area (of TB 2)
# 132 done, ~ 53.18% Completed ~ 52.06% area (of TB 2)
# 110 done, ~ 53.64% Completed ~ 52.56% area (of TB 2)
# 84 done, ~ 54.09% Completed ~ 52.80% area (of TB 2)
# 103 done, ~ 54.55% Completed ~ 53.30% area (of TB 2)
# 78 done, ~ 55.00% Completed ~ 53.54% area (of TB 2)
# 93 done, ~ 55.45% Completed ~ 54.01% area (of TB 2)
# 98 done, ~ 55.91% Completed ~ 54.50% area (of TB 2)
# 79 done, ~ 56.36% Completed ~ 54.76% area (of TB 2)
# 99 done, ~ 56.82% Completed ~ 55.25% area (of TB 2)
# 100 done, ~ 57.27% Completed ~ 55.75% area (of TB 2)
# 92 done, ~ 57.73% Completed ~ 56.22% area (of TB 2)
# 90 done, ~ 58.18% Completed ~ 56.71% area (of TB 2)
# 91 done, ~ 58.64% Completed ~ 57.21% area (of TB 2)
# 66 done, ~ 59.09% Completed ~ 57.57% area (of TB 2)
```

```
# 65 done, ~ 59.55% Completed ~ 57.75% area (of TB 2)
# 89 done, ~ 60.00% Completed ~ 58.24% area (of TB 2)
# 94 done, ~ 60.45% Completed ~ 58.62% area (of TB 2)
# 70 done, ~ 60.91% Completed ~ 59.09% area (of TB 2)
# 68 done, ~ 61.36% Completed ~ 59.58% area (of TB 2)
# 58 done, ~ 61.82% Completed ~ 60.08% area (of TB 2)
# 87 done, ~ 62.27% Completed ~ 60.55% area (of TB 2)
# 88 done, ~ 62.73% Completed ~ 61.02% area (of TB 2)
# 62 done, ~ 63.18% Completed ~ 61.49% area (of TB 2)
# 85 done, ~ 63.64% Completed ~ 61.96% area (of TB 2)
# 86 done, ~ 64.09% Completed ~ 62.43% area (of TB 2)
# 107 done, ~ 64.55% Completed ~ 62.92% area (of TB 2)
# 69 done, ~ 65.00% Completed ~ 63.42% area (of TB 2)
# 54 done, ~ 65.45% Completed ~ 63.59% area (of TB 2)
# 80 done, ~ 65.91% Completed ~ 64.08% area (of TB 2)
# 77 done, ~ 66.36% Completed ~ 64.58% area (of TB 2)
# 71 done, ~ 66.82% Completed ~ 65.08% area (of TB 2)
# 81 done, ~ 67.27% Completed ~ 65.57% area (of TB 2)
# 55 done, ~ 67.73% Completed ~ 66.04% area (of TB 2)
# 67 done, ~ 68.18% Completed ~ 66.49% area (of TB 2)
# 63 done, ~ 68.64% Completed ~ 66.96% area (of TB 2)
# 73 done, ~ 69.09% Completed ~ 67.43% area (of TB 2)
# 49 done, ~ 69.55% Completed ~ 67.93% area (of TB 2)
# 101 done, ~ 70.00% Completed ~ 68.43% area (of TB 2)
# 64 done, ~ 70.45% Completed ~ 68.87% area (of TB 2)
# 72 done, ~ 70.91% Completed ~ 69.37% area (of TB 2)
# 59 done, ~ 71.36% Completed ~ 69.86% area (of TB 2)
# 61 done, ~ 71.82% Completed ~ 70.33% area (of TB 2)
# 51 done, ~ 72.27% Completed ~ 70.63% area (of TB 2)
# 50 done, ~ 72.73% Completed ~ 70.93% area (of TB 2)
# 82 done, ~ 73.18% Completed ~ 71.42% area (of TB 2)
# 83 done, ~ 73.64% Completed ~ 71.92% area (of TB 2)
# 95 done, ~ 74.09% Completed ~ 72.41% area (of TB 2)
# 76 done, ~ 74.55% Completed ~ 72.91% area (of TB 2)
# 75 done, ~ 75.00% Completed ~ 73.40% area (of TB 2)
# 56 done, ~ 75.45% Completed ~ 73.87% area (of TB 2)
# 45 done, ~ 75.91% Completed ~ 74.37% area (of TB 2)
# 74 done, ~ 76.36% Completed ~ 74.86% area (of TB 2)
# 60 done, ~ 76.82% Completed ~ 75.33% area (of TB 2)
# 41 done, ~ 77.27% Completed ~ 75.83% area (of TB 2)
# 57 done, ~ 77.73% Completed ~ 76.33% area (of TB 2)
# 37 done, ~ 78.18% Completed ~ 76.82% area (of TB 2)
# 36 done, ~ 78.64% Completed ~ 77.32% area (of TB 2)
# 35 done, ~ 79.09% Completed ~ 77.81% area (of TB 2)
# 34 done, ~ 79.55% Completed ~ 78.28% area (of TB 2)
# 22 done, ~ 80.00% Completed ~ 78.75% area (of TB 2)
# 48 done, ~ 80.45% Completed ~ 79.25% area (of TB 2)
# 52 done, ~ 80.91% Completed ~ 79.74% area (of TB 2)
```

```
# 21 done, ~ 81.36% Completed ~ 80.24% area (of TB 2)
# 44 done, ~ 81.82% Completed ~ 80.73% area (of TB 2)
# 40 done, ~ 82.27% Completed ~ 81.23% area (of TB 2)
# 26 done, ~ 82.73% Completed ~ 81.70% area (of TB 2)
# 30 done, ~ 83.18% Completed ~ 82.17% area (of TB 2)
# 53 done, ~ 83.64% Completed ~ 82.67% area (of TB 2)
# 43 done, ~ 84.09% Completed ~ 83.14% area (of TB 2)
# 47 done, ~ 84.55% Completed ~ 83.61% area (of TB 2)
# 46 done, ~ 85.00% Completed ~ 84.10% area (of TB 2)
# 20 done, ~ 85.45% Completed ~ 84.60% area (of TB 2)
# 19 done, ~ 85.91% Completed ~ 85.09% area (of TB 2)
# 42 done, ~ 86.36% Completed ~ 85.59% area (of TB 2)
# 17 done, ~ 86.82% Completed ~ 86.06% area (of TB 2)
# 39 done, ~ 87.27% Completed ~ 86.53% area (of TB 2)
# 38 done, ~ 87.73% Completed ~ 87.03% area (of TB 2)
# 10 done, ~ 88.18% Completed ~ 87.50% area (of TB 2)
# 33 done, ~ 88.64% Completed ~ 87.99% area (of TB 2)
# 25 done, ~ 89.09% Completed ~ 88.49% area (of TB 2)
# 18 done, ~ 89.55% Completed ~ 88.98% area (of TB 2)
# 13 done, ~ 90.00% Completed ~ 89.43% area (of TB 2)
# 29 done, ~ 90.45% Completed ~ 89.93% area (of TB 2)
# 11 done, ~ 90.91% Completed ~ 90.40% area (of TB 2)
# 16 done, ~ 91.36% Completed ~ 90.89% area (of TB 2)
# 14 done, ~ 91.82% Completed ~ 91.39% area (of TB 2)
# 12 done, ~ 92.27% Completed ~ 91.86% area (of TB 2)
# 9 done, ~ 92.73% Completed ~ 92.33% area (of TB 2)
# 31 done, ~ 93.18% Completed ~ 92.83% area (of TB 2)
# 15 done, ~ 93.64% Completed ~ 93.32% area (of TB 2)
# 32 done, ~ 94.09% Completed ~ 93.82% area (of TB 2)
# 23 done, ~ 94.55% Completed ~ 94.31% area (of TB 2)
# 8 done, ~ 95.00% Completed ~ 94.78% area (of TB 2)
# 28 done, ~ 95.45% Completed ~ 95.28% area (of TB 2)
# 27 done, ~ 95.91% Completed ~ 95.77% area (of TB 2)
# 7 done, ~ 96.36% Completed ~ 96.24% area (of TB 2)
# 6 done, ~ 96.82% Completed ~ 96.71% area (of TB 2)
# 24 done, ~ 97.27% Completed ~ 97.21% area (of TB 2)
# 4 done, ~ 97.73% Completed ~ 97.68% area (of TB 2)
# 2 done, ~ 98.18% Completed ~ 98.14% area (of TB 2)
# 3 done, ~ 98.64% Completed ~ 98.61% area (of TB 2)
# 0 done, ~ 99.09% Completed ~ 99.06% area (of TB 2)
# 1 done, ~ 99.55% Completed ~ 99.53% area (of TB 2)
# 5 done, ~ 100.00% Completed ~ 100.00% area (of TB 2)

Concurrent[2]: All data is present in the output for TB 2.

Correction job complete for TB 2, Wed Feb 12 11:42:08 2014

 TB2 Times:User: 0.80 Sys: 0.66 Elapsed: 124.61 Memory: 191.608M
```

```
Correction job started for TB 3, Wed Feb 12 11:42:08 2014

Scanning cell graphics from previous template block.
+0%----------+25%--------+50%--------+75%---------+100%
....................................................
    Times: User: 0.20 Sys: 0.14 Elapsed: 0.17 Memory: 75.755M
Performing Overflow Processing for TB 3.
+0%----------+25%--------+50%--------+75%---------+100%
....................................................
    Times: User: 0.08 Sys: 0.06 Elapsed: 0.09 Memory: 78.059M
Generating templates from singleton instances.
+0%----------+25%--------+50%--------+75%---------+100%
....................................................
218 templates generated.
    Times: User: 0.06 Sys: 0.05 Elapsed: 0.05 Memory: 186.150M
# 217 done, ~ 0.46% Completed ~ 0.45% area (of TB 3)
# 208 done, ~ 0.92% Completed ~ 0.61% area (of TB 3)
# 215 done, ~ 1.38% Completed ~ 1.06% area (of TB 3)
# 214 done, ~ 1.83% Completed ~ 1.50% area (of TB 3)
# 209 done, ~ 2.29% Completed ~ 1.77% area (of TB 3)
# 192 done, ~ 2.75% Completed ~ 2.22% area (of TB 3)
# 207 done, ~ 3.21% Completed ~ 2.32% area (of TB 3)
# 191 done, ~ 3.67% Completed ~ 2.78% area (of TB 3)
# 203 done, ~ 4.13% Completed ~ 2.88% area (of TB 3)
# 216 done, ~ 4.59% Completed ~ 3.33% area (of TB 3)
# 210 done, ~ 5.05% Completed ~ 3.77% area (of TB 3)
# 199 done, ~ 5.50% Completed ~ 3.87% area (of TB 3)
# 206 done, ~ 5.96% Completed ~ 4.31% area (of TB 3)
# 204 done, ~ 6.42% Completed ~ 4.66% area (of TB 3)
# 195 done, ~ 6.88% Completed ~ 4.76% area (of TB 3)
# 202 done, ~ 7.34% Completed ~ 5.20% area (of TB 3)
# 200 done, ~ 7.80% Completed ~ 5.54% area (of TB 3)
# 198 done, ~ 8.26% Completed ~ 5.97% area (of TB 3)
# 182 done, ~ 8.72% Completed ~ 6.41% area (of TB 3)
# 189 done, ~ 9.17% Completed ~ 6.45% area (of TB 3)
# 183 done, ~ 9.63% Completed ~ 6.91% area (of TB 3)
# 213 done, ~ 10.09% Completed ~ 7.36% area (of TB 3)
# 205 done, ~ 10.55% Completed ~ 7.81% area (of TB 3)
# 184 done, ~ 11.01% Completed ~ 8.26% area (of TB 3)
# 197 done, ~ 11.47% Completed ~ 8.71% area (of TB 3)
# 211 done, ~ 11.93% Completed ~ 9.17% area (of TB 3)
# 201 done, ~ 12.39% Completed ~ 9.62% area (of TB 3)
# 178 done, ~ 12.84% Completed ~ 10.06% area (of TB 3)
# 172 done, ~ 13.30% Completed ~ 10.51% area (of TB 3)
# 173 done, ~ 13.76% Completed ~ 10.95% area (of TB 3)
# 174 done, ~ 14.22% Completed ~ 11.38% area (of TB 3)
# 181 done, ~ 14.68% Completed ~ 11.83% area (of TB 3)
```

```
# 193 done, ~ 15.14% Completed ~ 12.27% area (of TB 3)
# 179 done, ~ 15.60% Completed ~ 12.72% area (of TB 3)
# 187 done, ~ 16.06% Completed ~ 13.17% area (of TB 3)
# 194 done, ~ 16.51% Completed ~ 13.61% area (of TB 3)
# 171 done, ~ 16.97% Completed ~ 13.97% area (of TB 3)
# 180 done, ~ 17.43% Completed ~ 14.42% area (of TB 3)
# 176 done, ~ 17.89% Completed ~ 14.87% area (of TB 3)
# 177 done, ~ 18.35% Completed ~ 15.33% area (of TB 3)
# 212 done, ~ 18.81% Completed ~ 15.78% area (of TB 3)
# 196 done, ~ 19.27% Completed ~ 16.12% area (of TB 3)
# 175 done, ~ 19.72% Completed ~ 16.57% area (of TB 3)
# 166 done, ~ 20.18% Completed ~ 16.86% area (of TB 3)
# 169 done, ~ 20.64% Completed ~ 17.13% area (of TB 3)
# 161 done, ~ 21.10% Completed ~ 17.42% area (of TB 3)
# 163 done, ~ 21.56% Completed ~ 17.85% area (of TB 3)
# 188 done, ~ 22.02% Completed ~ 18.31% area (of TB 3)
# 186 done, ~ 22.48% Completed ~ 18.74% area (of TB 3)
# 167 done, ~ 22.94% Completed ~ 19.16% area (of TB 3)
# 165 done, ~ 23.39% Completed ~ 19.62% area (of TB 3)
# 168 done, ~ 23.85% Completed ~ 20.05% area (of TB 3)
# 170 done, ~ 24.31% Completed ~ 20.41% area (of TB 3)
# 162 done, ~ 24.77% Completed ~ 20.86% area (of TB 3)
# 164 done, ~ 25.23% Completed ~ 21.30% area (of TB 3)
# 159 done, ~ 25.69% Completed ~ 21.75% area (of TB 3)
# 190 done, ~ 26.15% Completed ~ 22.21% area (of TB 3)
# 185 done, ~ 26.61% Completed ~ 22.66% area (of TB 3)
# 155 done, ~ 27.06% Completed ~ 23.11% area (of TB 3)
# 146 done, ~ 27.52% Completed ~ 23.56% area (of TB 3)
# 158 done, ~ 27.98% Completed ~ 24.02% area (of TB 3)
# 160 done, ~ 28.44% Completed ~ 24.47% area (of TB 3)
# 153 done, ~ 28.90% Completed ~ 24.92% area (of TB 3)
# 143 done, ~ 29.36% Completed ~ 25.19% area (of TB 3)
# 147 done, ~ 29.82% Completed ~ 25.64% area (of TB 3)
# 145 done, ~ 30.28% Completed ~ 26.09% area (of TB 3)
# 157 done, ~ 30.73% Completed ~ 26.55% area (of TB 3)
# 156 done, ~ 31.19% Completed ~ 27.00% area (of TB 3)
# 152 done, ~ 31.65% Completed ~ 27.45% area (of TB 3)
# 144 done, ~ 32.11% Completed ~ 27.88% area (of TB 3)
# 148 done, ~ 32.57% Completed ~ 28.33% area (of TB 3)
# 154 done, ~ 33.03% Completed ~ 28.79% area (of TB 3)
# 141 done, ~ 33.49% Completed ~ 29.24% area (of TB 3)
# 151 done, ~ 33.94% Completed ~ 29.69% area (of TB 3)
# 140 done, ~ 34.40% Completed ~ 30.14% area (of TB 3)
# 149 done, ~ 34.86% Completed ~ 30.60% area (of TB 3)
# 136 done, ~ 35.32% Completed ~ 31.05% area (of TB 3)
# 138 done, ~ 35.78% Completed ~ 31.50% area (of TB 3)
# 134 done, ~ 36.24% Completed ~ 31.95% area (of TB 3)
# 142 done, ~ 36.70% Completed ~ 32.41% area (of TB 3)
```

```
# 150 done, ~ 37.16% Completed ~ 32.86% area (of TB 3)
# 139 done, ~ 37.61% Completed ~ 33.31% area (of TB 3)
# 137 done, ~ 38.07% Completed ~ 33.76% area (of TB 3)
# 132 done, ~ 38.53% Completed ~ 34.21% area (of TB 3)
# 135 done, ~ 38.99% Completed ~ 34.67% area (of TB 3)
# 130 done, ~ 39.45% Completed ~ 35.12% area (of TB 3)
# 133 done, ~ 39.91% Completed ~ 35.57% area (of TB 3)
# 128 done, ~ 40.37% Completed ~ 36.02% area (of TB 3)
# 131 done, ~ 40.83% Completed ~ 36.47% area (of TB 3)
# 127 done, ~ 41.28% Completed ~ 36.92% area (of TB 3)
# 129 done, ~ 41.74% Completed ~ 37.38% area (of TB 3)
# 113 done, ~ 42.20% Completed ~ 37.43% area (of TB 3)
# 124 done, ~ 42.66% Completed ~ 37.88% area (of TB 3)
# 121 done, ~ 43.12% Completed ~ 38.33% area (of TB 3)
# 123 done, ~ 43.58% Completed ~ 38.78% area (of TB 3)
# 126 done, ~ 44.04% Completed ~ 39.24% area (of TB 3)
# 125 done, ~ 44.50% Completed ~ 39.69% area (of TB 3)
# 119 done, ~ 44.95% Completed ~ 40.14% area (of TB 3)
# 114 done, ~ 45.41% Completed ~ 40.57% area (of TB 3)
# 120 done, ~ 45.87% Completed ~ 41.02% area (of TB 3)
# 122 done, ~ 46.33% Completed ~ 41.48% area (of TB 3)
# 108 done, ~ 46.79% Completed ~ 41.93% area (of TB 3)
# 116 done, ~ 47.25% Completed ~ 42.38% area (of TB 3)
# 112 done, ~ 47.71% Completed ~ 42.84% area (of TB 3)
# 117 done, ~ 48.17% Completed ~ 43.29% area (of TB 3)
# 115 done, ~ 48.62% Completed ~ 43.74% area (of TB 3)
# 107 done, ~ 49.08% Completed ~ 44.19% area (of TB 3)
# 102 done, ~ 49.54% Completed ~ 44.65% area (of TB 3)
# 111 done, ~ 50.00% Completed ~ 45.10% area (of TB 3)
# 118 done, ~ 50.46% Completed ~ 45.55% area (of TB 3)
# 101 done, ~ 50.92% Completed ~ 46.00% area (of TB 3)
# 109 done, ~ 51.38% Completed ~ 46.46% area (of TB 3)
# 110 done, ~ 51.83% Completed ~ 46.91% area (of TB 3)
# 103 done, ~ 52.29% Completed ~ 47.36% area (of TB 3)
# 105 done, ~ 52.75% Completed ~ 47.81% area (of TB 3)
# 106 done, ~ 53.21% Completed ~ 48.27% area (of TB 3)
# 94 done, ~ 53.67% Completed ~ 48.62% area (of TB 3)
# 104 done, ~ 54.13% Completed ~ 49.07% area (of TB 3)
# 95 done, ~ 54.59% Completed ~ 49.52% area (of TB 3)
# 96 done, ~ 55.05% Completed ~ 49.97% area (of TB 3)
# 97 done, ~ 55.50% Completed ~ 50.43% area (of TB 3)
# 100 done, ~ 55.96% Completed ~ 50.88% area (of TB 3)
# 93 done, ~ 56.42% Completed ~ 51.31% area (of TB 3)
# 98 done, ~ 56.88% Completed ~ 51.76% area (of TB 3)
# 99 done, ~ 57.34% Completed ~ 52.22% area (of TB 3)
# 92 done, ~ 57.80% Completed ~ 52.67% area (of TB 3)
# 91 done, ~ 58.26% Completed ~ 53.12% area (of TB 3)
# 89 done, ~ 58.72% Completed ~ 53.57% area (of TB 3)
```

```
# 90 done, ~ 59.17% Completed ~ 54.03% area (of TB 3)
# 84 done, ~ 59.63% Completed ~ 54.26% area (of TB 3)
# 87 done, ~ 60.09% Completed ~ 54.70% area (of TB 3)
# 88 done, ~ 60.55% Completed ~ 55.13% area (of TB 3)
# 86 done, ~ 61.01% Completed ~ 55.57% area (of TB 3)
# 85 done, ~ 61.47% Completed ~ 56.01% area (of TB 3)
# 78 done, ~ 61.93% Completed ~ 56.24% area (of TB 3)
# 79 done, ~ 62.39% Completed ~ 56.48% area (of TB 3)
# 83 done, ~ 62.84% Completed ~ 56.93% area (of TB 3)
# 82 done, ~ 63.30% Completed ~ 57.38% area (of TB 3)
# 81 done, ~ 63.76% Completed ~ 57.84% area (of TB 3)
# 80 done, ~ 64.22% Completed ~ 58.29% area (of TB 3)
# 70 done, ~ 64.68% Completed ~ 58.73% area (of TB 3)
# 73 done, ~ 65.14% Completed ~ 59.16% area (of TB 3)
# 68 done, ~ 65.60% Completed ~ 59.61% area (of TB 3)
# 65 done, ~ 66.06% Completed ~ 59.79% area (of TB 3)
# 66 done, ~ 66.51% Completed ~ 60.13% area (of TB 3)
# 71 done, ~ 66.97% Completed ~ 60.59% area (of TB 3)
# 77 done, ~ 67.43% Completed ~ 61.04% area (of TB 3)
# 58 done, ~ 67.89% Completed ~ 61.49% area (of TB 3)
# 69 done, ~ 68.35% Completed ~ 61.94% area (of TB 3)
# 72 done, ~ 68.81% Completed ~ 62.40% area (of TB 3)
# 67 done, ~ 69.27% Completed ~ 62.83% area (of TB 3)
# 62 done, ~ 69.72% Completed ~ 63.27% area (of TB 3)
# 76 done, ~ 70.18% Completed ~ 63.72% area (of TB 3)
# 74 done, ~ 70.64% Completed ~ 64.17% area (of TB 3)
# 54 done, ~ 71.10% Completed ~ 64.34% area (of TB 3)
# 61 done, ~ 71.56% Completed ~ 64.77% area (of TB 3)
# 64 done, ~ 72.02% Completed ~ 65.19% area (of TB 3)
# 75 done, ~ 72.48% Completed ~ 65.65% area (of TB 3)
# 60 done, ~ 72.94% Completed ~ 66.08% area (of TB 3)
# 57 done, ~ 73.39% Completed ~ 66.53% area (of TB 3)
# 63 done, ~ 73.85% Completed ~ 66.97% area (of TB 3)
# 59 done, ~ 74.31% Completed ~ 67.42% area (of TB 3)
# 55 done, ~ 74.77% Completed ~ 67.86% area (of TB 3)
# 51 done, ~ 75.23% Completed ~ 68.14% area (of TB 3)
# 56 done, ~ 75.69% Completed ~ 68.58% area (of TB 3)
# 50 done, ~ 76.15% Completed ~ 68.85% area (of TB 3)
# 53 done, ~ 76.61% Completed ~ 69.31% area (of TB 3)
# 49 done, ~ 77.06% Completed ~ 69.76% area (of TB 3)
# 47 done, ~ 77.52% Completed ~ 70.20% area (of TB 3)
# 52 done, ~ 77.98% Completed ~ 70.65% area (of TB 3)
# 46 done, ~ 78.44% Completed ~ 71.10% area (of TB 3)
# 43 done, ~ 78.90% Completed ~ 71.54% area (of TB 3)
# 48 done, ~ 79.36% Completed ~ 71.99% area (of TB 3)
# 35 done, ~ 79.82% Completed ~ 72.44% area (of TB 3)
# 45 done, ~ 80.28% Completed ~ 72.90% area (of TB 3)
# 37 done, ~ 80.73% Completed ~ 73.35% area (of TB 3)
```

```
 # 39 done, ~ 81.19% Completed ~ 73.79% area (of TB 3)
 # 41 done, ~ 81.65% Completed ~ 74.24% area (of TB 3)
 # 30 done, ~ 82.11% Completed ~ 74.68% area (of TB 3)
 # 34 done, ~ 82.57% Completed ~ 75.11% area (of TB 3)
 # 38 done, ~ 83.03% Completed ~ 75.57% area (of TB 3)
 # 36 done, ~ 83.49% Completed ~ 76.02% area (of TB 3)
 # 44 done, ~ 83.94% Completed ~ 76.47% area (of TB 3)
 # 33 done, ~ 84.40% Completed ~ 76.93% area (of TB 3)
 # 42 done, ~ 84.86% Completed ~ 77.38% area (of TB 3)
 # 32 done, ~ 85.32% Completed ~ 77.83% area (of TB 3)
 # 22 done, ~ 85.78% Completed ~ 78.27% area (of TB 3)
 # 31 done, ~ 86.24% Completed ~ 78.72% area (of TB 3)
 # 40 done, ~ 86.70% Completed ~ 79.17% area (of TB 3)
 # 29 done, ~ 87.16% Completed ~ 79.63% area (of TB 3)
 # 21 done, ~ 87.61% Completed ~ 80.08% area (of TB 3)
 # 25 done, ~ 88.07% Completed ~ 80.53% area (of TB 3)
 # 24 done, ~ 88.53% Completed ~ 80.98% area (of TB 3)
 # 20 done, ~ 88.99% Completed ~ 81.43% area (of TB 3)
 # 19 done, ~ 89.45% Completed ~ 81.89% area (of TB 3)
 # 26 done, ~ 89.91% Completed ~ 82.32% area (of TB 3)
 # 28 done, ~ 90.37% Completed ~ 82.78% area (of TB 3)
 # 23 done, ~ 90.83% Completed ~ 83.23% area (of TB 3)
 # 27 done, ~ 91.28% Completed ~ 83.68% area (of TB 3)
 # 18 done, ~ 91.74% Completed ~ 84.13% area (of TB 3)
 # 17 done, ~ 92.20% Completed ~ 84.57% area (of TB 3)
 # 13 done, ~ 92.66% Completed ~ 84.99% area (of TB 3)
 # 14 done, ~ 93.12% Completed ~ 85.44% area (of TB 3)
 # 12 done, ~ 93.58% Completed ~ 85.88% area (of TB 3)
 # 16 done, ~ 94.04% Completed ~ 86.33% area (of TB 3)
 # 9 done, ~ 94.50% Completed ~ 86.77% area (of TB 3)
 # 10 done, ~ 94.95% Completed ~ 87.21% area (of TB 3)
 # 11 done, ~ 95.41% Completed ~ 87.64% area (of TB 3)
 # 15 done, ~ 95.87% Completed ~ 88.10% area (of TB 3)
 # 7 done, ~ 96.33% Completed ~ 88.53% area (of TB 3)
 # 8 done, ~ 96.79% Completed ~ 88.97% area (of TB 3)
 # 6 done, ~ 97.25% Completed ~ 89.41% area (of TB 3)
 # 4 done, ~ 97.71% Completed ~ 89.84% area (of TB 3)
 # 2 done, ~ 98.17% Completed ~ 90.28% area (of TB 3)
 # 3 done, ~ 98.62% Completed ~ 90.72% area (of TB 3)
 # 5 done, ~ 99.08% Completed ~ 91.15% area (of TB 3)
 # 1 done, ~ 99.54% Completed ~ 91.59% area (of TB 3)
 # 0 done, ~ 100.00% Completed ~ 92.01% area (of TB 3)
Calculating context for leaf cells.
 leaf context: 10%
 leaf context: 20%
 leaf context: 30%
 leaf context: 40%
 leaf context: 50%
```

```
 leaf context: 60%
 leaf context: 70%
 leaf context: 80%
 leaf context: Done Wed Feb 12 11:42:10 2014

     Times: User: 1.55 Sys: 0.99 Elapsed: 2.07 Memory: 196.511M
Calculating context for holding cells.
 holder context: 10%
 holder context: 20%
 holder context: 30%
 holder context: 40%
 holder context: 50%
 holder context: 60%
 holder context: 70%
 holder context: 80%
 holder context: 90%
 holder context: Done Wed Feb 12 11:42:10 2014

     Times: User: 0.00 Sys: 0.00 Elapsed: 0.00 Memory: 191.798M
Generating output files for correction.
 output files: 10%
 output files: 20%
 output files: 30%
 output files: 60%
 output files: 70%
 output files: 80%
 output files: 90%
 output files: Done Wed Feb 12 11:42:10 2014

     Times: User: 0.00 Sys: 0.01 Elapsed: 0.00 Memory: 191.759M

 220 output templates (218 cluster templates) generated for TB3:
 flat: 218 templates (224 instances)
 holder: 2 templates ( 2 instances)
 Total topcell bounding area: 85598.4280 square microns.
 Sum of all template bounding areas: 78760.9214 square microns.
 Sum of all ambit-biased template bounding areas: 87266.8274
square microns.

 Hierman BackEnd Times: User: 1.89 Sys: 1.25 Elapsed: 2.40 Memory:
196.585M
Client: executing mktop in a separate thread...
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 99.09% Completed ~ 100.00% area (of TB 3)
Client: executing mktop in a separate thread...
 Building Instance reference list for full output hierarchy
```

```
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 99.09% Completed ~ 100.00% area (of TB 3)
Client: executing mktop in a separate thread...
 Building Instance reference list for full output hierarchy
 Creating dummy TOPCELL_OUT: TOP ...
 Building hierarchy cells ...
 # mktop done, ~ 100.00% Completed ~ 100.00% area (of TB 3)
 server 1: ltgpe-03 exited: MSG_NUM 172 Total Times:User: 110.45
Sys: 0.12 Elapsed: 121.65 Memory: 93.614M
 server 2: ltgpe-05 exited: MSG_NUM 73 Total Times:User: 69.02
Sys: 0.08 Elapsed: 79.42 Memory: 85.591M
 server 3: ltgpe-05 exited: MSG_NUM 71 Total Times:User: 73.52
Sys: 0.08 Elapsed: 79.42 Memory: 92.962M
 server 4: ltgpe-05 exited: MSG_NUM 71 Total Times:User: 71.01
Sys: 0.06 Elapsed: 79.41 Memory: 81.874M
 server 5: ltgpe-05 exited: MSG_NUM 67 Total Times:User: 66.71
Sys: 0.08 Elapsed: 79.39 Memory: 88.712M
 server 6: rutro-04 exited: MSG_NUM 64 Total Times:User: 68.37
Sys: 0.46 Elapsed: 79.36 Memory: 85.834M
 server 7: rutro-04 exited: MSG_NUM 63 Total Times:User: 71.55
Sys: 0.25 Elapsed: 79.32 Memory: 91.427M
 server 8: ltgpe-05 exited: MSG_NUM 67 Total Times:User: 67.08
Sys: 0.08 Elapsed: 79.27 Memory: 88.721M
 server 9: ltgpe-05 exited: MSG_NUM 68 Total Times:User: 74.64
Sys: 0.04 Elapsed: 79.21 Memory: 86.926M
 server 10: ltgpe-73 exited: MSG_NUM 63 Total Times:User: 65.88
Sys: 0.09 Elapsed: 78.68 Memory: 81.614M
 server 11: ltgpe-73 exited: MSG_NUM 65 Total Times:User: 67.35
Sys: 0.16 Elapsed: 78.65 Memory: 85.505M
 server 12: ltgpe-73 exited: MSG_NUM 59 Total Times:User: 63.95
Sys: 0.09 Elapsed: 78.64 Memory: 87.307M
 server 13: ltgpe-73 exited: MSG_NUM 71 Total Times:User: 66.93
Sys: 0.10 Elapsed: 78.64 Memory: 81.535M
 server 14: ltgpe-73 exited: MSG_NUM 61 Total Times:User: 70.90
Sys: 0.08 Elapsed: 78.63 Memory: 86.519M
 server 15: ltgpe-73 exited: MSG_NUM 61 Total Times:User: 67.23
Sys: 0.12 Elapsed: 78.63 Memory: 90.523M
 server 16: ltgpe-73 exited: MSG_NUM 65 Total Times:User: 67.59
Sys: 0.10 Elapsed: 78.58 Memory: 84.435M
 server 17: ltgpe-73 exited: MSG_NUM 63 Total Times:User: 65.10
Sys: 0.17 Elapsed: 78.57 Memory: 83.730M
 server 18: ltgpe-73 exited: MSG_NUM 59 Total Times:User: 69.17
Sys: 0.13 Elapsed: 78.57 Memory: 88.214M
 server 19: ltgpe-73 exited: MSG_NUM 59 Total Times:User: 67.60
Sys: 0.38 Elapsed: 78.53 Memory: 90.456M
 server 20: ltgpe-43 exited: MSG_NUM 67 Total Times:User: 68.21
Sys: 0.10 Elapsed: 78.45 Memory: 90.494M
 server 21: ltgpe-master1 exited: MSG_NUM 51 Total Times:User:
59.69 Sys: 0.12 Elapsed: 74.01 Memory: 81.292M
```

```
 server 22: ltgpe-master2 exited: MSG_NUM 43 Total Times:User:
61.65 Sys: 0.14 Elapsed: 73.47 Memory: 73.932M
 server 23: ltgpe-master2 exited: MSG_NUM 45 Total Times:User:
60.32 Sys: 0.12 Elapsed: 73.47 Memory: 72.725M
 server 24: ltgpe-master2 exited: MSG_NUM 49 Total Times:User:
68.82 Sys: 0.13 Elapsed: 73.44 Memory: 72.482M
 server 25: ltgpe-master2 exited: MSG_NUM 45 Total Times:User:
62.03 Sys: 0.13 Elapsed: 73.41 Memory: 73.869M
 server 26: ltgpe-master2 exited: MSG_NUM 45 Total Times:User:
64.71 Sys: 0.11 Elapsed: 73.39 Memory: 74.647M
 server 27: ltgpe-master2 exited: MSG_NUM 45 Total Times:User:
66.46 Sys: 0.11 Elapsed: 73.38 Memory: 74.837M
 server 28: ltgpe-master1 exited: MSG_NUM 46 Total Times:User:
63.26 Sys: 0.11 Elapsed: 71.67 Memory: 75.188M
 server 29: ltgpe-master1 exited: MSG_NUM 42 Total Times:User:
60.85 Sys: 0.11 Elapsed: 71.55 Memory: 74.189M
 server 30: ltgpe-master1 exited: MSG_NUM 46 Total Times:User:
65.39 Sys: 0.09 Elapsed: 71.46 Memory: 67.424M
 server 31: ltgpe-master1 exited: MSG_NUM 46 Total Times:User:
57.88 Sys: 0.14 Elapsed: 71.45 Memory: 72.459M
 server 32: ltgpe-master1 exited: MSG_NUM 40 Total Times:User:
57.88 Sys: 0.10 Elapsed: 71.42 Memory: 73.904M

 Final output file ./gds/TB1_out.gds is now complete, Wed Feb 12
11:42:11 2014


Prepare the holder cells for bound box data calculation for TB3.
 prepare holder cells: 10%
 prepare holder cells: 20%
 prepare holder cells: 30%
 prepare holder cells: 40%
 prepare holder cells: 50%
 prepare holder cells: 60%
 prepare holder cells: 70%
 prepare holder cells: 80%
 prepare holder cells: 90%
 prepare holder cells: Done Wed Feb 12 11:42:11 2014


Collecting cells bound box data for TB3.
 collect cell data: 10%
 collect cell data: 20%
 collect cell data: 30%
 collect cell data: 40%
 collect cell data: 50%
 collect cell data: 60%
 collect cell data: 70%
 collect cell data: 80%
```

```
 collect cell data: Done Wed Feb 12 11:42:11 2014


Finalizing OASIS output file ./gds/TB2_out.oas for TB3.
 finish output file: 10%
 finish output file: 20%
 finish output file: 30%
 finish output file: 40%
 finish output file: 50%
 finish output file: 60%
 finish output file: 70%
 finish output file: 80%
 finish output file: Done Wed Feb 12 11:42:11 2014


 Final output file ./gds/TB2_out.oas is now complete, Wed Feb 12
11:42:11 2014


 Final output file ./gds/MOF_BL_pcx2_out.gds is now complete, Wed
Feb 12 11:42:11 2014


Concurrent[3]: All data is present in the output for TB 3.
server 33: ltgpe-master1 exited: MSG_NUM 42 Total Times:User:
64.84 Sys: 0.11 Elapsed: 72.61 Memory: 74.782M

 Correction job complete for TB 3, Wed Feb 12 11:42:11 2014


 Cleaning up intermediate recipe files at project finish
 rm -rf MOF_NONE_0210_TB1.pjx 2> /dev/null

 Cleaning up intermediate recipe files at project finish
 rm -rf MOF_NONE_0210_TB2.pjx 2> /dev/null

 Cleaning up intermediate recipe files at project finish
 rm -rf MOF_NONE_0210_TB3.pjx 2> /dev/null

 Cleaning up fragment directories at project finish
 rm -rf ./gds/TB1_out.gds.dir/ 2> /dev/null

 Correction job complete for all template blocks, Wed Feb 12
11:42:12 2014

 TB3 Times:User: 3.54 Sys: 1.99 Elapsed: 128.45 Memory: 197.351M

Checking in PROTEUS_OPC...
```

```
Checking in PA...

 Server Processing Time Summary
 TB1 Elapsed: 0:00:49 [11:40:10-11:40:59]
 TB2 Elapsed: 0:01:07 [11:41:00-11:42:08]
 TB3 Elapsed: 0:00:00 [11:42:08-11:42:09]

Total Times:User: 11.39 Sys: 2.95 Elapsed: 138.00 Memory: 197.351M
```

Glossary

abut
When a polygon edge (or face) is flush with (abutting) the edge of another polygon, and the interiors of the two polygons on opposite sides of the abutting edge (not overlapping or overlaid). *See also* touch state, normal, correction segment.

accumulated estimate
An intermediate value in an iterative process. Typical correction recipes compare the results of an accumulated correction estimate against error criteria, then readjust the correction and repeat until the results are acceptable.

ahead face
The polygon edge immediately ahead of the current or center face being processed. *See also* back face, right-hand rule.

ambit
A scalar quantity (in nanometers) describing a correction range, as defined in the job control file or model file. This value comes from both the model and the recipe, and should agree in both files.

angle
The angle between a polygon edge and its next neighbor is defined as the equivalent base direction the neighbor polygon edge assumes when the original edge is held as a horizontal reference (base direction 0). In other words, a 90-degree right turn is defined as an angle of 6, regardless of the absolute base direction of either edge. In corrected shapes all edges are normalized to orientations that are integer multiples of 45 degrees.

annular illumination
When the distribution of source illumination, as seen from the object plane, appears as an annulus or ring. Modifications of source illumination shape are intended to improve stepper imaging characteristics. *See also* off-axis illumination.

ASCII file format
An acronym for American Standard Code for Information Interchange, the ASCII format defines a standard, printed-character, alphanumeric interpretation for a byte of data (strictly speaking, only the first 7 bits or 128 characters for standard ASCII, the full 256 characters for extended ASCII). In general, an

ASCII file is considered to contain text, and can be displayed and manipulated directly using a text editor, such as vi or EMACS. In UNIX there is no distinction between an ASCII file and a binary format file. In DOS (PC), however, these are different file formats. When transferring ASCII DOS files to UNIX you must ensure proper conversion with dos2unix or a similar utility.

assist feature
See auxiliary feature.

auxiliary feature
A nonprinting structure added to mask pattern data in order to influence proximity effects or to balance process load. The term is also used to refer to features added for data marking. *See also* outrigger.

back end (BE)
The context analysis and template generation portion of the hierarchy analysis.

back face
The polygon edge immediately behind the current or center face being processed. *See also* ahead face, right hand rule.

back load
A mode of operation in the concurrent pipeline in which some results from the last stage are accessed as soon as possible.

base direction
The absolute angle of a polygon edge or correction segment.

basis file
A convolution kernel stored in a proprietary format, used in behavior models and accessed directly by the correction processor.

behavior model
A two-dimensional model consisting of one or more convolution kernels (in the form of basis files) and a mapping function. A behavior model generates a scalar property value, such as intensity.

binary format
A general file format. Bytes (data) contained in a binary format file are interpreted via an explicit file format specification. *See also* ASCII file format.

boundary
A GDSII structure describing a polygon.

CD
See critical dimensions.

cell
In IC design, a graphical unit of pattern data.

center face
The current face being processed. *See also* ahead face, back face, right-hand rule.

clamping
In correction recipes, clamping confines the correction assigned to a segment to fall between an upper and lower limit. This can only be accomplished with a user-written function.

cluster
One or more pattern cells, grouped together according to the clustering process, for consideration as a correction template.

clustering
The process whereby the Hierarchy Manager applies a user-specified size parameter to the pattern hierarchy at various levels, finding the appropriate level at which to group underlying branches and leaves into a single cluster for consideration as a correction template.

concurrent pipeline
Subsequent template blocks can start before prior blocks are complete. The only change to PJF recipes required to run concurrent pipelining is to add the `PIPELINE_STRATEGY` keyword to the job control file. Concurrent pipelining also refers to the concurrency enabled by running `PROTEUS_EXCHANGE` capabilities, such as allowing the CATS application to process Proteus tool output concurrently.

context-based hierarchy
A hierarchy organized to address conveniently both target geometries to be corrected and the regions that surround them. The Hierarchy Manager produces a context-based hierarchy by arranging the input pattern into templates.

contour
The curve generated by intersecting a 3-D surface with a plane. The intersecting plane is typically perpendicular to the vertical (Z) axis of the surface, and specified with a single scalar threshold corresponding to the Z position.

contour point
A point in a slice. *See also* contour.

convolution kernel
A two-dimensional function (expressed in the form of a basis file) that is combined with the image data in a mathematical operation called a convolution. One or more convolution kernels, when combined with a mapping function, embody the behavior models used in the model-based correction technique.

convolution value
For a given point in a two-dimensional graphic image, a convolution value is computed by multiplying a region of the image centered around the point by a two-dimensional function or convolution kernel, and then integrating the resulting product in both dimensions. In the correction processor, the center point used for this operation is the evaluation point. *See also* mapping function, convolution value, behavior model.

corBASIC
A limited dialect of the BASIC language, used to write correction recipes.

correction
See proximity correction.

correction cell
That portion of the correction template (along with the recursion cell) that contains the target geometries to be corrected. The contents of the correction cell are passed to the output file after correction, whereas the recursion cell is temporary.

correction grid
The grid defined in the job control file to which all output pattern polygon vertices must conform. Used by the Correction Processor.

correction recipe
A class of statements found in the job control file that specifies the behavior models to be used, defines correction options, and provides a detailed, relatively low-level correction program that specifies the correction instructions.

The portion of the correction process that evaluates behavior models, inverts the proximity effect (*see* inverse mask), and applies a displacement to correction segments. It also verifies performance of the corrected shapes, performs logging functions, and optionally generates auxiliary or assist features.

correction segment
The atomic graphical element of correction, as created during dissection by dividing the edges of the original pattern polygons into shorter pieces with stitch points. Each correction segment has an associated target point located somewhere along its length, as well as registers (`CORWT`, `DIR`, `TLEN`, and so forth) associated with it.

correction segment geometries
A local coordinate system based on the current correction segment. *See* the *corBASIC Reference Manual* for an explanation of this local coordinate system.

correction template
The Hierarchy Manager reorganizes the input data into an optimal set of geometric elements, which capture cell proximity relationships while retaining as much hierarchic compression as possible (using provided proximity parameters).

The structures of these elements are contained within the correction template.

correction weight
The offset amount to be applied to a correction segment, as calculated by the correction recipe. A positive correction weight moves the segment outward (away from the interior of the polygon), while a negative weight moves it inward.

crimp
The process of preparing non-45 degree input edges for correction by converting polygon edges into an approximating sequence of segments, which are multiples of 45 degrees (0, 45, 90, …). *See* MIN_N45_L on page 232 for further information on crimping.

critical dimensions (CDs)
The widths of the lines and spaces of critical circuit patterns, as well as the area of contacts. *See also* design rules.

database unit
The physical units associated with database values of length/position, such as nanometers.

design rules
A set of rules that dictate the minimum sizes of specific features (and typically for specific materials, such as polysilicon vs. metal) for a given process. Usually the rules are based on a single minimum feature size, expressed as a fraction of a micron (10-6 meter) or in nanometers.

design rule check (DRC)
The process of inspecting an integrated circuit pattern for violation of the design rules.

dissection
The process of cutting polygon edges into segments by assigning stitch points and target points to the pattern data polygons in preparation for proximity correction. Also called segmentation.

DRC
See design rule check.

edge
The line section defined by two adjacent vertices in polygon data. This is delimited by a change of direction in the polygon.

edge deviation
The difference (distance) between drawn pattern edges and actual or modeled edge position.

empirical behavior model
A behavior model trained, regressed, or parameterized from a set of experimental CD measurements using test patterns.

evaluation point
The point at which a behavior model is evaluated. This might be identical to the target point, or offset from it under correction recipe control.

face
An edge. The line section defined by two adjacent vertices in polygon data.

flat pattern
Pattern data that contains little or no hierarchy. A hierarchic pattern can be flattened by expanding all cell references to instances in absolute space. *See also* hierarchic compression.

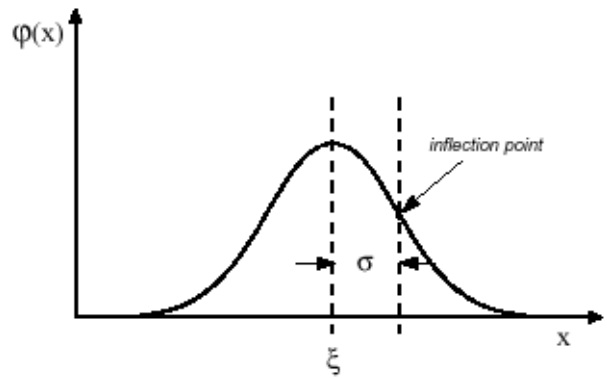
front end (FE)
The hierarchic preprocessing that must occur before the back-end (BE) hierarchic analysis can be performed. Includes scaffolding.

front load
A mode of operation in the concurrent pipeline in which the first stage is performed with all servers; then, as servers are no longer able to contribute to the first stage, they move on to perform work on the second stage, and so forth. This can have efficiency benefits without the extra cost of back load.

Gaussian
Variants of the bell-shaped curve described by:

Equation 1

$$\varphi(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-1}{2}\left(\frac{x-\xi}{\sigma}\right)^2}$$



This is also known as the *normal frequency function* or *probability density function*.

GDSII (.gds)
An integrated circuit editor and database created by the Calma Company. The binary interchange format for this data, called GDSII Stream, has been released to the public domain and has become a popular format for interchanging IC layout data. The Proteus Hierarchy Manager conforms to GDSII Stream release 6.0. A format specification for GDSII is available from http://www.vsi.org.

global iteration
A pass through the entire data structure, in which all template shapes are adjusted as part of a multiple-pass method of applying and assessing proximity correction. Local iterations, by contrast, may be applied to a single correction segment in the process of determining the appropriate correction weight.

graphics scaffold
In the Hierarchy Manager, this is a process that limits the maximum size of any one correction template by dividing large cells into smaller fragments. The Hierarchy Manager accomplishes this by breaking very large polygons into

smaller ones of a maximum permissible size and arranging them in groups—the size, shape and complexity of which are determined by parameters in the job control file.

GUI
Acronym for Graphical User Interface. A software user interface that relies heavily on graphical interactions as opposed to text commands.

hierarchic compression
Reducing the total size of a data file by identifying repeated structures and organizing the data into a hierarchy of patterns and replicated instances of those patterns.

Hierarchy Manager (hierman)
The Proteus Hierarchy Manager reads GDSII or OASIS input files and outputs a table of correction templates. The objective of the Hierarchy Manager is to find an optimum number of unique correction templates to maximize correction processing efficiency and to minimize the size of the output file.

instance
To represent graphic data involving repetitive structures efficiently, it is often organized into a list or hierarchy of prototype pattern structures, accompanied by specific placements of those structures in the image. These placements, called instances, are created by associating the prototype structures with specific spatial transformations (position, scale, rotation, and so forth). The definition of prototype structures and instances can involve a multi-level hierarchy, so prototype patterns can themselves be combinations of patterns and instances.

interference state analysis
The process of reducing overlaps or interference between the template polygons to a single touch state by adding vertices. *See also* edge, face, segment, touch state, and the individual definitions of the touch states normal and abut.

intermediate recipe
A recipe from a single `TEMPLATE_BLOCK` of a `PROTEUS_JOB_FLOW` generated using the `RETAIN_JOB_FLOW_FILES` keyword.

inverse mask
A mask that incorporates geometry changes to counteract, or form an inverse of, known systematic distortions. This is the final product of proximity correction.

iteration
A single pass through a process that requires or embodies repetition. *See also* global iteration, local iteration.

job control file
A text file accessed by virtually every process in the correction processor, the job control file contains global statements, the dissection program, and correction recipes. The job control file conventionally has a .pjx extension.

kernel
See convolution kernel.

lambda
In optics, the wavelength of the light used by an optical projection system.

layer
When two-dimensional image data is organized into potentially overlapping/overlaid but distinct types of data, these different datatypes are often referred to as *layers*. For example, selective correction control layers are used to describe regions of varying correction behavior rather than shapes in the final output mask data. Also, input data might include multiple layers associated with individual process levels in the IC.

leaf
In a hierarchy tree, a node with no associated subnodes; a terminating node.

linear transversal filter
A means of redistributing correction weights within a five-segment region (*see* Chapter 4, Hierarchy Management, for more details). Linear transversal filtering is invoked using the `LTV_LN_FILTER` recipe statement.

local iteration
The correction process typically involves several levels at which iterative adjustments are made. Not only is the entire correction template typically traversed (on a segment-by-segment basis) several times to capture interactive effects, but the correction weight applied to an individual segment may involve an iterative loop as well. The latter is termed a local iteration.

long-range effects (LRE)
Process effects that create proximity-like distortions over relatively long distances (50 to 100 mm, or longer). For example, reactive ion etch loading can create such long-range effects.

mapping function
A function specifying the algebraic combination of convolution values, the final result of which is a signal value, typically meaning an intensity or pseudo-intensity. *See also* convolution.

mark
A selection technique based on cell-name and hierarchic path/node information.

match conditions
A class of statements used to determine which rules apply to which polygon edges in the pattern data via feature characteristics such as edge length, direction, angle, level, or interference state. Match conditions are evaluated by the dissection program.

MEBES
An acronym for Mask Electron Beam Exposure System. Refers both to a line of mask writing systems built by Etec, as well as a file format accepted by such systems.

model-based correction
An algorithm that calculates correction values based on a simulation of the effect to be corrected. *See also* rule-based correction.

NA
See numerical aperture.

node
The smallest elements in a hierarchic structure that reference, or are referenced by, other structures. The stream file nodes may consist of cells containing one or more explicitly defined geometric elements, as well as references to other cells.

non-pipeline application
A non-`PROTEUS_JOB_FLOW` recipe for a single cohesive purpose.

normal (touch state)
The condition in which a polygon edge neither touches nor intersects any other polygon. *See also* touch state, abut, and correction segment.

numerical aperture (NA)
When squared, a measure of the light-gathering properties of a lens. The resolving power of an optical system, defined as the minimum transverse distance between two object points that can be resolved (separated), is

inversely proportional to the numerical aperture (within depth of field/focus constraints). The numerical aperture (NA) is related to the f number of a lens by f/# = 1/(2(NA)).

OASIS
OASIS is a file format created by the Semiconductor Equipment and Materials Institute (SEMI) that has improvements over the GDSII file format, including a reduced file size.

outrigger
A type of auxiliary or assist feature added to the pattern mask data in order to control certain types of proximity effects, usually in the form of elongated rectangles parallel to features in the pattern data.

pattern orientation dependency
Lithography effects that depend on the orientation of pattern edges, such as when using a quadrupole stepper.

PCX
Refers to `PROTEUS_EXCHANGE CATS|CATS2` (also known as PCX1 and PCX2). PCX2 is a potential stage in the pipeline in which incremental data is sent to CATS as templates are processed. There are restrictions on which modes this is compatible with. *See* PROTEUS_EXCHANGE on page 262 for details.

PCX1 is an older generation and less efficient technology for sending incremental data to CATS as templates are processed. There are restrictions on which modes this is compatible with. *See* pipeline and `PIPELINE_STRATEGY`.

phase shift mask (PSM)
A technique for enhancing the imaging qualities of a mask by manipulating the relative phase of light passing through adjacent features. When light of the same phase passes through two adjacent transparent regions of the mask, the diffraction patterns can add together and cause a blending of the two features. If the phase of the light passing through one feature is shifted 180 degrees (by either adding or subtracting the appropriate amount of refractive medium), the diffraction patterns between the structures is subtractive rather than additive.

pipeline
A technology that allows multiple stages at full-chip scope to be concurrently executed during a Proteus tool run. (A stage can be a "template block" or a non-template based operation, such as a global operation like DPT.) It is also

possible to include non-Proteus modules in the pipeline (for example CATS) through the PROTEUS_EXCHANGE interface. Also referred to as Proteus Pipeline Technology (PPT).

PIPELINE_STRATEGY FRONT_LOAD|BACK_LOAD
Recipe keywords that cause full concurrency for all stages. These two settings turn on the concurrent pipeline. *See* PIPELINE_STRATEGY on page 261 for details.

PIPELINE_STRATEGY SINGLETONS
A special mode of operation that identifies some singleton templates quickly, in order to start distributed processing of some work as soon as possible before hierarchy management has fully completed. In this mode, only context analysis and the following template block can be processed concurrently. *See* PIPELINE_STRATEGY on page 261 for details.

placement
See instance.

polyline
A series of connected vertices, usually describing a path or an edge.

Proteus Pipeline Technology (PPT)
A technology that allows multiple stages (also known as template blocks) to be specified and executed during a Proteus tool run. It is also possible to pipeline to a different engine (for example CATS) through the PROTEUS_EXCHANGE interface. Also referred to as pipeline.

proteus
An executable that runs the flow specified in a recipe input. When using PJF-style recipes, proteus allows you to split hierarchy management steps to allow separation of front-end analysis from the back end. *See* proteus on page 387 for details and options.

PROTEUS_JOB_FLOW (PJF)
The high-level recipe syntax that enables specification and execution of multiple stages (also known as template blocks) using the same front end hierarchic analysis. Any Proteus recipe using PJF syntax is also pipelined. *See* PROTEUS_JOB_FLOW and END_PROTEUS_JOB_FLOW on page 75 for syntax details.

protobar
The prototype correction primitive, the protobar is a long bar of programmable width and position. Used to perturb an edge location to determine the simulated effect quickly.

protosegment
A protosegment is a pending dissection segment that exists only as a table of data. When the `UPDATE_GRAPHICS` function is called, these segments become recipe segments.

prototype correction
A trial correction in an iterative process, evaluated to determine whether it meets acceptable error criteria.

proximity correction
The process of countering systematic distortions encountered during IC fabrication by creating an inverse mask.

proximity effect
Effects that are dependent on pattern features surrounding (in the proximity of) the point of distortion.

radial axis
In correction segment geometry, the radial axis is the axis normal to the correction segment, which is the same direction as `RDELTA`. The positive direction of the radial axis points away from the polygon interior. *See also* tangential axis.

recipe
An algorithm as contained in a script file.

recursion cell
This defines the evaluation context of an associated correction cell. The size of the recursion cell is dictated by the proximity ambit.

recursive effect
Local proximity corrections rendered less correct by corrections applied to nearby regions later in the process. Global iterations are used to gradually stabilize recursive effects.

register
In Proteus correction recipes, registers represent predefined locations in memory, in which pertinent values are stored and accessed. (For example, `N_SEG` stores the number of segments in the current polygon, `SEG_NO` stores the segment index number of the current segment.)

right-hand rule
A protocol that associates the order/direction of a series of polygon edges with the interior of the polygon. If the index finger of your right hand (with palm down) is pointing in the edge direction, the extended thumb points toward the

polygon interior. In other words, the outer edge of a right-hand rule polygon runs in a counterclockwise direction, whereas interior or hole edges run clockwise.

rule-based correction
An algorithm that determines corrections via a set of rules, as opposed to one which uses a model of the proximity effect to calculate a correction. *See also* model-based correction.

run (segment)
A term used to describe a chain of equal-length segments, or, alternatively, to describe the length of such segments.

scanning electron microscope (SEM)
A type of microscope used to magnify images as much as 50,000 times, by scanning the surface of the subject with an electron beam. The impacting electrons cause surface electrons to be emitted (in some cases the surface must be prepared with a metallic coating, usually applied via sputtering). The emitted electrons are used to generate an image of the surface.

script file
ASCII files containing declarations and recipes used to control the operation of the correction processor.

segment
See correction segment.

segmentation
See dissection.

SEM
See scanning electron microscope.

serial pipeline
One stage is performed after the other with no repeat of the front-end hierman work (FE) compared to a non-pipeline recipe. Stages are not concurrently processed.

sigma
The partial coherence. This is the conventional symbol for standard deviation, or, alternatively, an optical parameter describing the shape of an illuminator.

stage
Refers to the work done inside a `TEMPLATE_BLOCK` during a PJF recipe in either concurrent or nonconcurrent modes.

stand-alone application
A PROTEUS_JOB_FLOW recipe that could be one or more TEMPLATE_BLOCKs for a single cohesive purpose. This is a pipeline recipe. For example, PJF recipes for OPC or AF.

stitch point
A vertex added to the pattern polygon data during the process of segmentation. The stitch points and original vertices divide the pattern data into atomic correction segments, each of which has an associated target point, and registers.

structure
In the GDSII data format, the building blocks used to describe IC layout data in a hierarchical form. Structure elements accepted by the correction processor include: BOUNDARY, a polygon; PATH, a polyline of finite width; SREF, a single placement reference to another structure; and AREF, a multiple-placement reference to another structure. Structures are identified by an ASCII name. Names must consist only of the characters A-Z, a-z, 0-9, _, $, @, and must be between 1 and 32 characters long, although most applications accept longer names. (The Proteus tool accepts names up to 256 characters in length.)

systematic proximity distortion
Deviations that are repeatable and predictable from pattern geometry. Random proximity deviations are independent of pattern geometry, but may be systematic to other variables.

tangential axis
In correction segment geometry, the tangential axis is the axis parallel to (along) the correction segment, which is in the same direction as TDELTA. The positive direction of the tangential axis points toward the next (ahead) segment in the sequence. *See also* radial axis.

target point
A single point defined along each correction segment as the centering-point for the convolution operations associated with computing the correction weight/value for that segment. *See also* evaluation point.

template
See correction template.

template block (TB)
A step in a flow of many steps during a `PROTEUS_JOB_FLOW` that uses template distribution to divide and process the entire chip. `TEMPLATE_BLOCK`s are defined in a specific recipe section, allowing inclusion of Boolean operations and `corbasic()` calls.

An intermediate `TEMPLATE_BLOCK` is any `TEMPLATE_BLOCK` other than the last one in a recipe.

template call
In a PJF recipe, this is the call to a particular `TEMPLATE_BLOCK` executed as part of the flow, as opposed to the definition specified by the `TEMPLATE_BLOCK` itself. The definition of a template block (like a function) could be called with `TEMPLATE_CALL` multiple times with different parameters. The `TEMPLATE_CALL` itself appears in the PJF section of the recipe.

An intermediate `TEMPLATE_CALL` is any `TEMPLATE_CALL` other than the last one in a recipe.

theoretical behavior model
A behavior model based on a theoretical analysis and characterization of the distortion effects to be corrected for, as opposed to one derived from measured parameters. *See also* empirical behavior model.

threshold
See contour, threshold contour.

threshold contour
Conceptually, the contour representing the edge in the final image if the correction in its present form is applied. An iterative correction recipe compares this contour against the ideal edge position (as defined in the original pattern data) and continue to modify the correction segment displacement until the difference (error) drops below an acceptable tolerance.

touch state
A classification of how two polygons touch or intersect one another, performed prior to segmentation. Touch states include normal and abut.

touch state analysis
See interference state analysis.

unified application recipe
Multiple stand-alone applications put together in a single `PROTEUS_JOB_FLOW` recipe. This flow of applications is in a `PROTEUS_JOB_FLOW` recipe and is also a pipeline recipe. For example, RBAF, OPC, and PLRC combined in a single `PROTEUS_JOB_FLOW` recipe.

vertex segment
In dissection, a segment at one end of the original (presegmentation) polygon edge.

whisker
In a polygonal structure, an isolated edge or segment that protrudes from a drawn figure into the surrounding area and does not bound any interior portion of the figure (such as an edge that doubles back upon itself).

xmscript
A Proteus utility that assembles the components of a correction project (the dissection program, correction recipe, and global statements) into a single job control file. When invoked, the xmscript utility displays a form in the Proteus Script Builder window for selecting runtime parameters. The content of this form is defined using control statements in the global statement file, which conventionally has the .xjc extension.

Feedback

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z