SYNOPSYS®
Predictable Success

Newsletter for data prep engineers

Issue 3 | Spring 2011

CATS news

In This Issue

CATS is 25!

I am pleased to welcome you to the Spring 2011 issue of the CATS newsletter. The year 2011 marks the 25th anniversary of CATS and thanks to our supportive and demanding customers it continues to be the industry's most widely used mask data preparation solution. From its modest beginning as the only

Continued on back cover

MRC in CATS Version E-2011.03

John Valadez, CATS Applications Engineer

Customer-driven, continuous improvement is what Synopsys strives for. A perfect example of this can be found with CATS MRC. In previous CATS versions (those released prior to 2006) users had the ability to perform only a single space and width check per fracture operation. However, mask shops needed something more powerful, and single checks were insufficient as data grew increasingly complex. CATS addressed this need with its Y-2006.09 release, combining its ability to quickly read and process both fractured machine data and design data, with its industry standard counterpart, Hercules' DRC engine. Since then, CATS MRC has continued to make additional enhancements with each release. In addition, features such as the MRC Explorer have been added to improve ease of use.

With the release of CATS Version E-2011.03, not only will you find improvements to the most commonly used fundamental dimensional checks, but CATS has also integrated newer and more powerful functionality. This includes edge selection, filtering, and sizing, for multi-layer data. In addition, the core engine has been overhauled so that the output of one rule can be used as input to a subsequent rule for DRC-like flow.

Continued on page 2

CATS Zeiss PROVE Marking Interface

Stephen Kim, CATS Application Engineer

Through a close collaboration with Zeiss, Synopsys has developed the CATS marking interface for the Zeiss PROVE™ registration metrology tool. This new marking interface is available with the major release of CATS version E-2011.03 (a.k.a. CATS v32). In addition to industry-proven marking interfaces and capabilities, CATS also supports Zeiss' 2D correlation measurement method by providing reference clips in the OASIS.MASK format.

CATS marking interface for Zeiss PROVE includes the following features:

- Support for both standard and advanced mark types such as box-in-box, and mark link capability for target registration sites
- Parsing of Zeiss Template XML files to be used with CATS marking information
- A graphical and command line interface for Zeiss PROVE-specific parameters
- The ability to produce valid Zeiss PROVE recipe files in XML format based on the Zeiss PROVE XML schema
- The ability to produce valid OASIS.MASK design clips according to Zeiss specifications
- 2D alignment mark support

Continued on page 5

MRC in CATS continued from page 1

Here are previews of some specific new capabilities:

Extensive Corner Checks

Users can now isolate corner-to-corner and corner-to-edge violations when performing dimensional checks with either 1 or 2 layer input. A few examples are shown here:

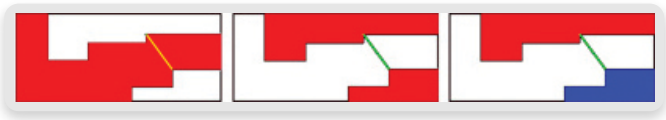


Figure 1: Concave to Convex

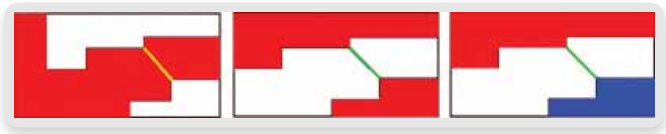


Figure 2: Convex to Convex

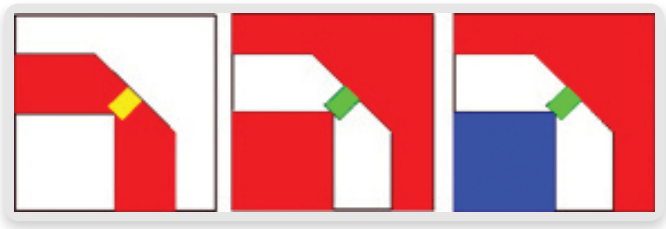


Figure 3: Convex to Edge

Other options have been added to dimensional checking to assist the user in isolating and flagging concave, convex, and 90° angles; however, this is just the tip of the iceberg.

Data Creation

A robust set of MRC SELECT commands make it possible to output new layers, based on results from operations on either a single input layer or two input layers. For example, a user may want to fill in slots or donut holes. The solution is achievable with the MRC SELECT INSIDE_HOLE command. Further tuning is facilitated by activating some of the options available for this command such as INNER_HOLE_ONLY or EMPTY_HOLE_ONLY.

Figure 4: MRC SELECT INSIDE_HOLE EMPTY_HOLE_ONLY TRUE

The new MRC SELECT operators used for data creation and filtering are defined in Table 1 below.

Table 1: SELECT Operators

COMMAND	DESCRIPTION
SELECT INSIDE_HOLE	Selects the hole in any donut-shaped piece of data on single-layer
SELECT INSIDE	Selects data from PRIMARY that is fully inside data of SECONDARY
SELECT OUTSIDE	Selects data from PRIMARY that is fully outside data of SECONDARY
SELECT INTERACT	Selects data from PRIMARY that interacts with data of SECONDARY
SELECT TOUCHING	Selects data from PRIMARY that is fully outside of and touches data of SECONDARY
SELECT EDGE_TOUCH	Selects data from PRIMARY that has a coincident edge with SECONDARY and is fully inside or fully outside SECONDARY, depending on the chosen option
SELECT ENCLOSING	Selects data from PRIMARY that fully encloses and touches data of SECONDARY

Edge Selection

When the need arises to characterize the perimeter of a polygon, specialized MRC SELECT commands are used to sort out the edges. Filtering can be performed by angle, length, or in the case of two-layer inputs, whether they touch or interact with each other.

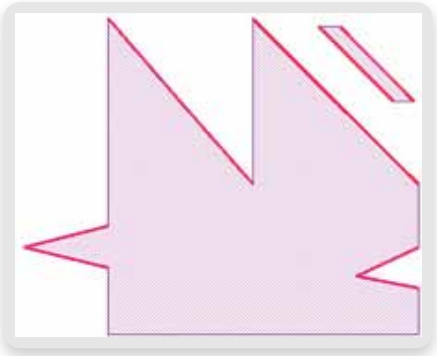


Figure 5: Using MRC SELECT_EDGE1 to Flag Non-Orthogonal Edges (1 Input)

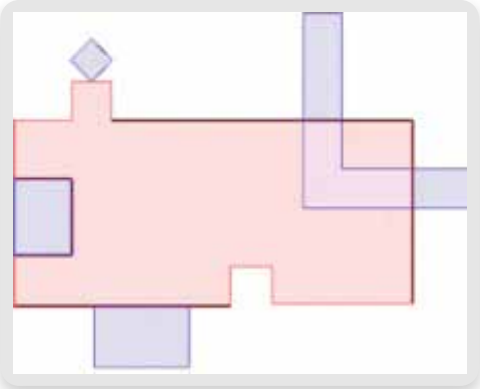


Figure 6: Using MRC SELECT_EDGE2 to Flag Interacting Edges (2 Inputs)

Edge selection is another option available in our dimensional checks. The OUTPUT_EDGES option controls this. With the EXTERNAL2 check, which requires two-layer inputs, the user can output both edges, the edge from input 1, or input 2. For example, Figure 7 shows that only one edge is flagged after a spacing check of <= 4.

Vectors and Vector Sizing

Once edges are filtered and assigned to a layer in the output file, they can then be used as input into other operations in order to be further processed through the MRC engine using other MRC operations. Moreover, these edges can be shortened, extended, or expanded using the MRC VSIZE command.

Using Figure 7 output data, the edges on both sides can be shortened, and then expanded outward by using the following:

```
MRC VSIZE 2 HEAD -1
MRC VSIZE 2 TAIL -1
MRC VSIZE 2 OUTSIDE 1
```

Now the output will appear as shown in Figure 8 below.

It does not take long to see where this is going. MRC VSIZE is at the heart of edge movement. Vector sizing is fundamental to rules-based OPC. Table 2, on the next page, lists the options applicable to MRC VSIZE.

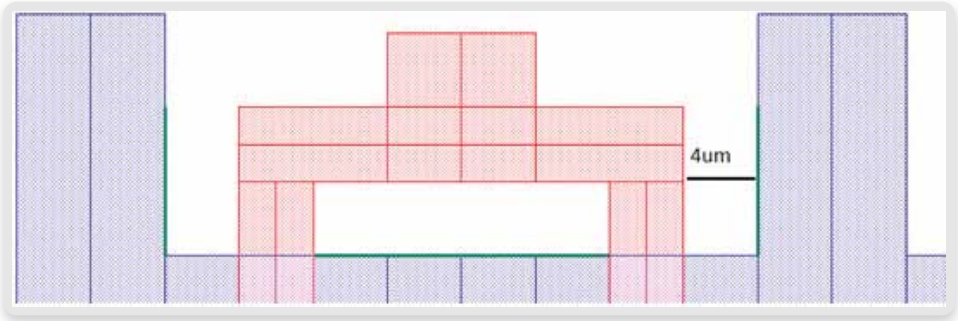


Figure 7: MRC EXTERNAL2 OUTPUT_EDGES EDGEB

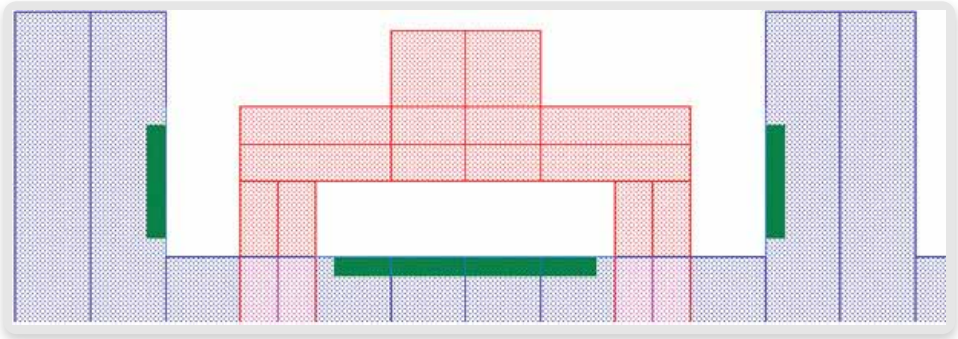


Figure 8: Using MRC VSIZE to Shorten and Expand Edge

Table 2: VSIZE Command Options with Descriptions

VSIZE option	Description
INSIDE	Widens the inside of a directed vector path
OUTSIDE	Widens the outside of a directed vector path
HEAD	Extends or shortens the beginning of a directed vector path
TAIL	Extends or shortens the end of a directed vector path

These are just a few of the new MRC features available in CATS Version E-2011.03. Other features include error compaction and output reports in XML format. For dimensional checks, users can specify a min and max range. CATS has also created and implemented a new spacing check option to capture errors between internal data and its data boundary. This type of check is very useful for Jobdeck MRC.

Mask Correction

The new options for the MRC dimensional commands are very powerful. The new MRC commands, like VSIZE, SELECT and POLYGON_FEATURES, offer very interesting possibilities. But when these features are coupled with the new MRC engine architecture, the possibilities are endless.

The new MRC engine allows not only the application of rules to the input data, but also allows for data feedback. This means that data can be processed by a set of rules, one after another, allowing OPC-like functionality.

Consider Figures 7 and 8. The first step is an EXTERNAL2 rule, which results in edges. The second step is a VSIZE rule, which modifies the edges and explodes them, resulting in the green rectangles in Figure 8. These two steps can be performed by the following rule set:

```
MRC EXTERNAL2 1 TEMPORARY 1
MRC EXTERNAL2 1 SPACING <= 4
MRC EXTERNAL2 1 OUTPUT_EDGES EDGEB
MRC VSIZE 2 INPUT T:1
MRC VSIZE 2 LAYER 2
MRC VSIZE 2 HEAD -1
MRC VSIZE 2 TAIL -1
MRC VSIZE 2 OUTSIDE 1
```

The output of the EXTERNAL2 command is left on TEMPORARY buffer 1. VSIZE then takes this intermediate result and processes it, leaving the final result on layer 2.

Since the temporary buffer was used, the intermediate result will not be output to a file, which is more efficient compared to using the output buffer.

This same data feedback technique can be expanded for applications in Mask Corrections. Consider the following scenario.

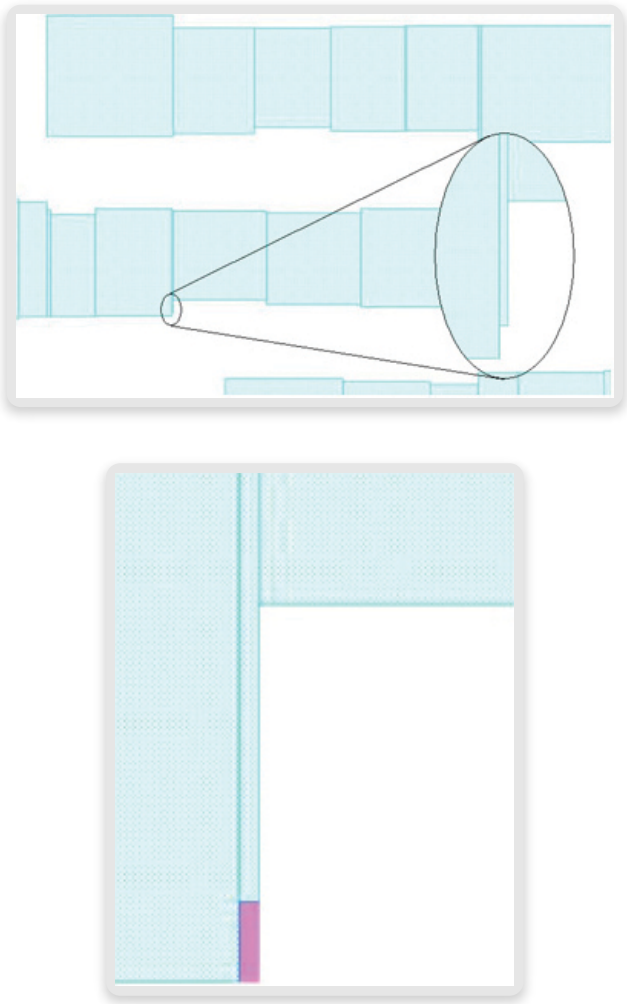


Figure 9: Embedded sliver and resulting notch

After a fracture, data may contain embedded slivers due to misaligned jogs on opposite sides of a polygon, which can be caused by OPC. If for process reasons, the data must be biased a certain value, eliminating the corner notch can eliminate these slivers.

Such slivers can number in the millions, so if they were removed, the write time (and maybe inspection time) could be reduced noticeably.

This can be accomplished with the new MRC engine by four simple rules:

```
MRC SELECT VECTOR_LENGTH 1 ON
MRC SELECT VECTOR_LENGTH 1 TEMPORARY 1
MRC SELECT VECTOR_LENGTH 1 LENGTH_RANGE 0 0.004
MRC INTERNAL 2 INPUT T:1
MRC INTERNAL 2 TEMPORARY 2
MRC INTERNAL 2 FLAG_CONVEX_ANGLE TRUE
MRC INTERNAL 2 FLAG_CONVEX_90 TRUE
MRC INTERNAL 2 POINT_EXPLODE 0.0005
MRC SELECT EDGE2 3 ON
MRC SELECT EDGE2 3 INTERACT
MRC SELECT EDGE2 3 TEMPORARY 3
MRC SELECT EDGE2 3 INPUT P T:2
```

```
MRC POLYGON_FEATURES BOUNDING_BOX 4 ON
MRC POLYGON_FEATURES BOUNDING_BOX 4 INPUT T:3
MRC POLYGON_FEATURES BOUNDING_BOX 4 LAYER 0
```

For maximum efficiency, the TEMPORARY buffers were used to pass the intermediate results between rules. The result will be rectangles that fill out the notches (Figure 9 bottom image), placed on layer 0 of the output. When the input data is ORed with these rectangles, the result is data that no longer contains the embedded slivers, i.e. a corrected mask.

From basic dimensional checks to data filtering, and from edge processing to mask correction, CATS MRC in release 2011.03 provides the user a full suite of tools that will assist them in developing customized applications unique to their situations.

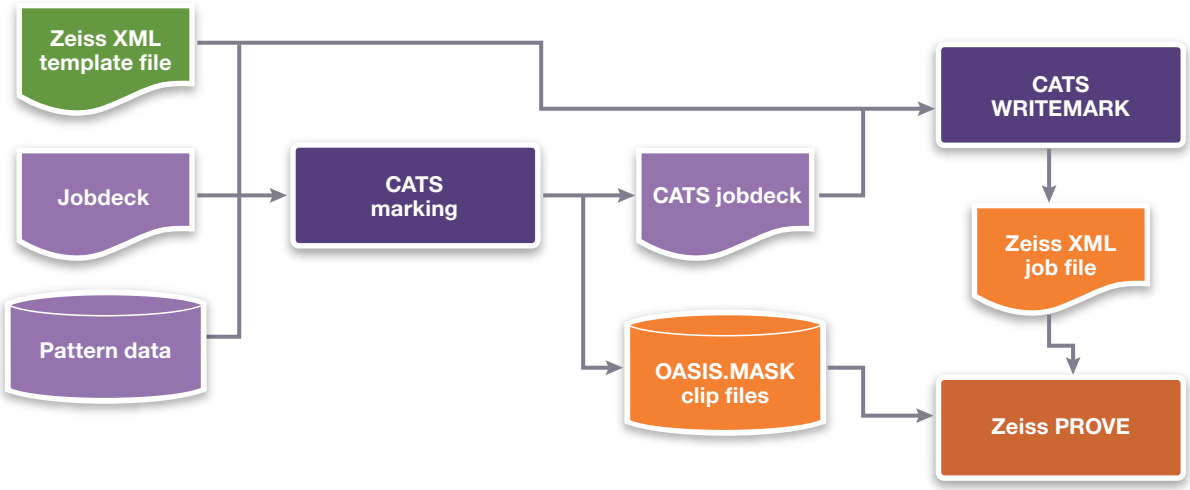
CATS Zeiss continued from page 1

The following diagram shows the CATS marking flow and how it generates PROVE job files (shown in orange).

To generate a PROVE job file, a Zeiss PROVE Template file (also in XML format) is combined with marks created during a CATS marking session.

A Zeiss PROVE Template file can be considered to be a static XML file that contains specific information and instructions created or maintained by a PROVE tool engineer. This file provides additional information needed by the PROVE tool to properly execute, and as such, the information in the job template must be merged with additional information generated from CATS during a marking session to create a final recipe file. Because of this, such a template file should be provided to the CATS interface during a marking session. CATS will then parse the job template file and merge its contents with the marking information it generates from the marking session to create the final PROVE job file.

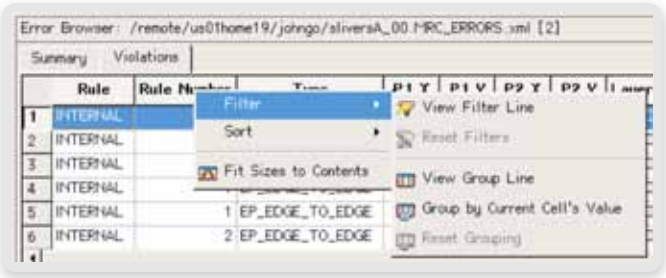
The final PROVE recipe files include an XML job file and image clips (per mark) in OASIS.MASK format, ready to be validated and executed by a Zeiss PROVE tool.



Advanced Filtering in MRC Browser

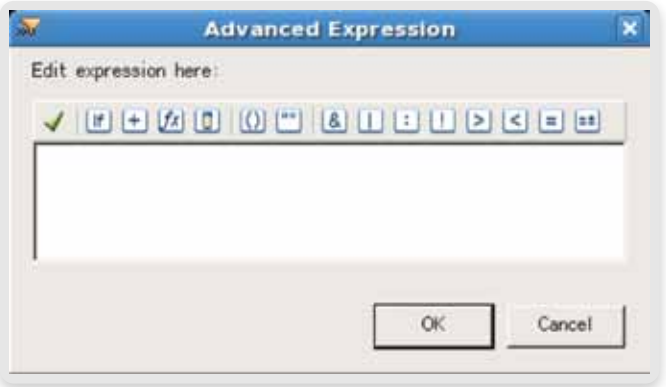
John Gookassian, CATS R&D Engineer

Most MRC users are aware of the filtering feature in the MRC Browser, where right clicking a cell, and choosing **Filter➔View Filter Line** adds the filtering line to the top of the browser, similar to that of Microsoft Excel®.



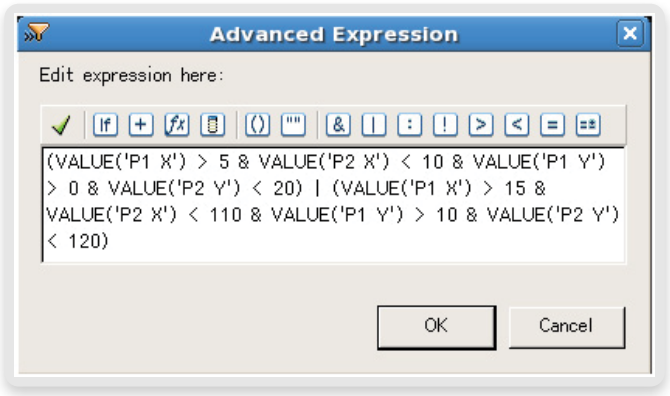
However, few are aware of the button in the filter line:

This "Advanced Expression" button opens a dialog that allows you to define complex filtering expressions that support Booleans, math and statistical functions, conditional expressions, and more:



While the full description of usage is documented in the CATS User Guide, the following example illustrates what can be done using this dialog:

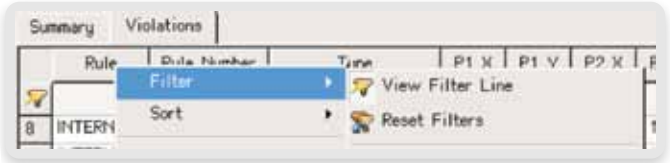
Example: Filtering out violations in two regions of the layout



After you have entered an expression, it is always a good idea to click the Verify ( ) button to check the syntax.

Note also that using the toolbar helps to prevent errors. For example, to enter a column name, click the button, then double-click the column name in the field that appears. Using the previous figure, if you double-clicked the "P1 X", "VALUE('P1 X')" would be inserted at the current caret location.

To cancel filtering and return to the entire error list, choose **Filter➔Reset Filters** from the context menu of MRC Browser:



Automatic CINC Generation Using CSVCATS

Kazunori Takano, CATS Applications Engineer

When you execute some fracture jobs in CATS, you must specify copious amounts of information. For example, various environment variables, I/O utilities such as READFILE and WRITEFILE to convert external and internal format, sizing and Boolean operations, and many .cinc steps for complicated polynomial expression.

Because the information should be specified in shell and .cinc files separately, and both are poor for documentation, it is difficult to unify and standardize the management of information input. Additionally, other specification documents may need to be written.

CSVCATS is a synthetic automation tool that interprets CSV (Comma Separated Values) format files with a series of CATS fracturing job descriptions created by Microsoft Excel or OpenOffice. Please note the following:

- There is no official support for this tool. The following is provided as an example of CATS automation.
- This tool should be used responsibly with sufficient testing.

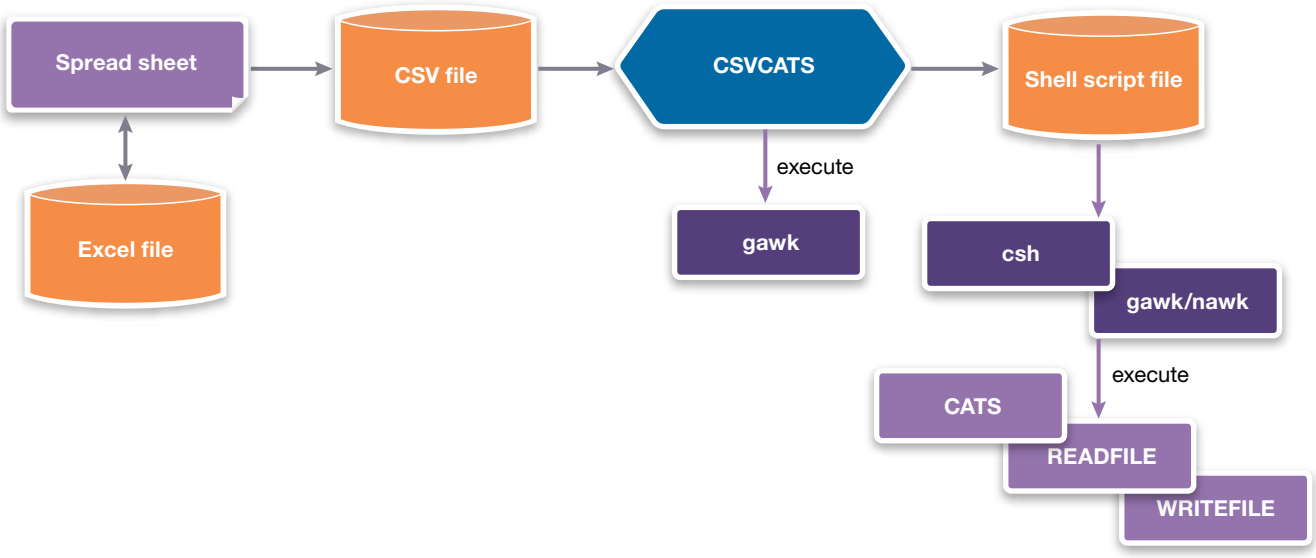
Flow

The data flow for CSVCATS is as follows:

Capabilities

CSVCATS allows you to do the following:

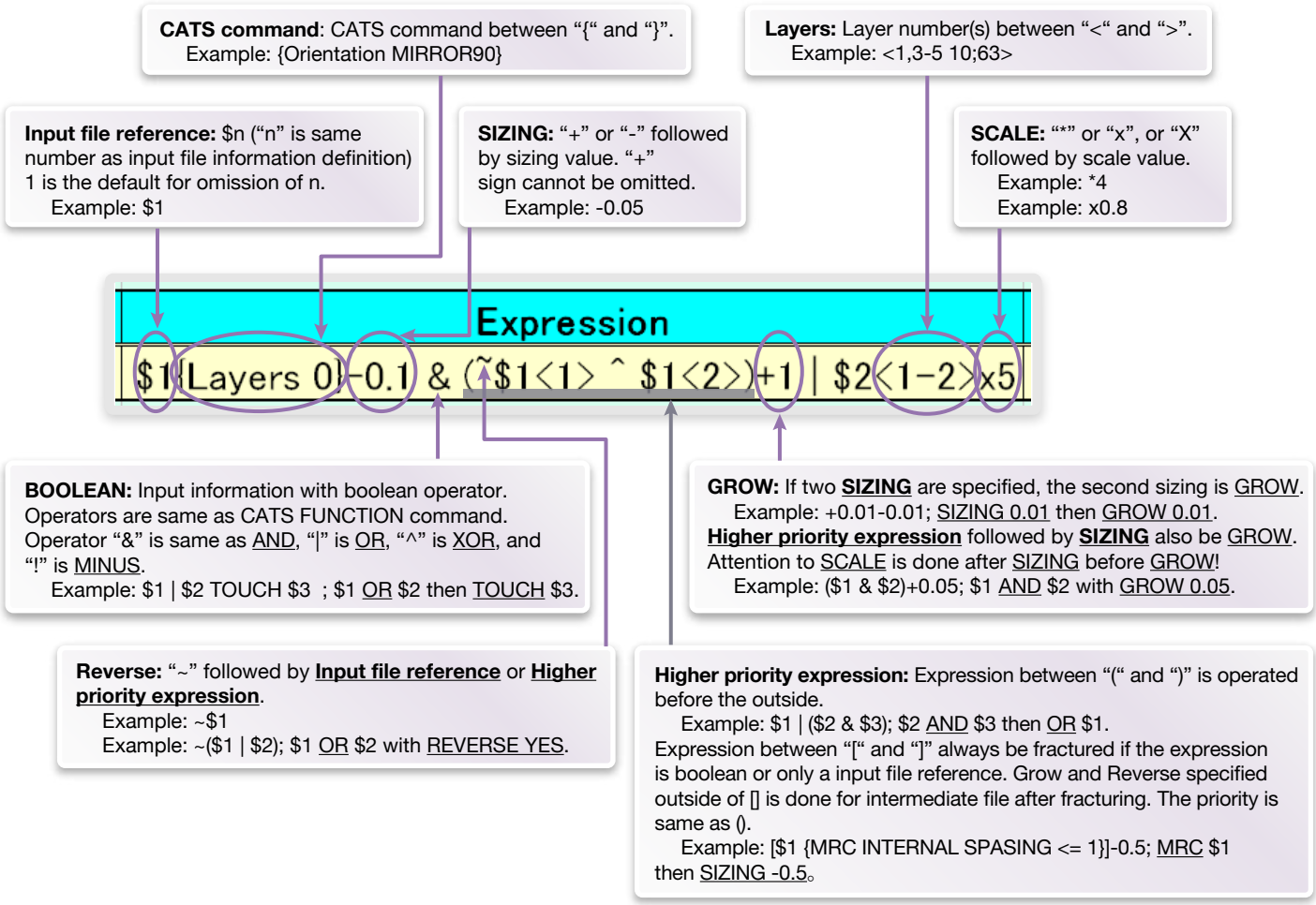
- Manage all information in a single spreadsheet by describing environment variables, fracturing commands, I/O utilities, and general UNIX commands.
- Reduce your work time and mistakes, and increase efficiency using its strong expression capability. A maximum of 10 input files can be referred, while there are only two in CATS. There is no need to write multiple .cinc files for complicated fracturing steps, thus reducing confusing intermediate filenames and time checking.
- Specify multiple operations such as sizing, scaling, and Booleans by a formula.
- Accelerate automation, simplification, and documentation by describing these in the Excel sheet for the CSV file. Excel has many functions such as borders, macros, and a VB application. You may create and manage those spreadsheets in a large database using Access.
- Easily read and write the CSV file using a text editor—the CSV file is text.
- Output a shell script file for execution. You can execute it in the background using a batch tool (such as LSF or SGE).
- Modify and customize CSVCATS as needed. CSVCATS is a script program of gawk.



Command Summary

The following commands and terminology are used in CSVCATS:

- **ENVIRONMENT:** Environment variables
- **DEFAULT:** Commands and their arguments as default
- **FRACTURE:** Fracturing commands and their arguments
- **EXPRESSION:** Defining multiple-step operations in a single polynomial expression. For example:
- **DISTRIBUTE or DP:** DISTRIBUTE commands and their arguments
- **CATS:** CATS commands and their arguments without the fracture
- **NORMALIZE:** JD NORMALIZE commands and their arguments
- **BATCH:** Batch commands (such as at, qsub, bsub) and their arguments
- **Other commands:** I/O utilities (such as READFILE, WRITEFILE) and their arguments
- **Direct Execution:** "!" followed by a UNIX command.



Example CSV Files

Example 1: Direct output fracture to JEOL31 and VSB12 format files

# This is an example spreadsheet for CSVCATS.											
# Environment variables for CATS											
ENVIRONMENT	TED	TE_HISTORY	TE_CHECKSUM_IN	TE_CHECKSUM_OUT	TE_LICENSE_QUEUE						
	.	$TED/transcriptchst	FALSE	TRUE	0 10 0						
# Fracture defaults											
DEFAULT	Allocate_rects	Allocate_traps	Allocate_space	Compact	Overlaps	Twosided	Rule	Border	Produce	VSB_THALO_SIZE	Threads
	500000	500000	1000	YES	NO	YES	WPARAGON	OUTSIDE	VSB_T	12	2
DEFAULT	Input	Structure	Limits	Scale	Input2	Structure2	Limits2				
	$TED/flyingcats	BIGARRAY	(-50,-30) (1640,113	4	$TED/flyingcats	TOPSTRU	(-200,0) (200,220)				
# Fracture by expression											
FRACTURE	Format	Resolution	Height	Width	Fieldheight	Fieldwidth	Fieldoverlap	Direct_output	Expression	Output	
	JEOL31	0.001	500	500	500	500	4	YES	$1<0>-0.1 & ("$1<1> " $1<2>)+1 |$2<1-2>x5	$TED/BIGARRAY_J31DO.JEOL3	
	"	"	"	"	"	"	"	OASIS	"	$TED/BIGARRAY_J130ASIS.mas	
	VSB12	0.0005	128	128	512	128	10	YES	"	$TED/BIGARRAY_VSB12DO/chi	
!rm -fr *.dir											

Example 2: MEBES Jobdeck editing in progress

# This is an example spreadsheet for CSVCATS.									
#									
# Environment variables									
ENVIRONMENT	TED	TE_HISTORY	TE_CHECKSUM_IN	TE_CHECKSUM_OUT	TE_LICENSE_QUEUE				
	.	$TED/transcriptchst	TRUE	TRUE	0 10 0				
# Defaults									
DEFAULT	Format	Resolution							
	MEBES	0.25							
!rm -f	EXAMPLE6.JB								
# Create a jobdeck									
CATS	JD ADD JOB	JD ADD CHIP ""	JD ADD PATTERN ""	JD ADD INSTANCE ""	JD ADD ARRAY ""	JD MODIFY JOBDECK	JD MODIFY PATTERN	JD WRITE	
	EXAMPLE6.JB 5" 0.25								¥
# Chip PERI									
		PERI	PERIMTRA0 50 1		3500,41000 1 4 0.15000		PERIMTRA0 50 ADDRESS=0.5		¥
					41000,3500 4 1 15000,0				¥
					41000,123500 4 1 15000,0				¥
					123500,41000 1 4 0.15000				¥
# Chip PRIME									
		PRIME	PRIMARY00 01 1		18500,18500 9 9 10000,10000				¥
					18500,108500 9 1 10000,0				¥
					108500,18500 1 9 0.10000				¥
# Chip INSERT									
		INSERT	TUNINGFRK_01 1	108,500,108,500			TUNINGFRK_01 REVERSE=YES		¥
# Chip FRAME									
		FRAME	FRAMEDEMO_01 A	"					¥
						MIRROR=YES		EXAMPLE6.JB	

Execution

To run CSVCATS, use the following syntax:

```
csvcats.csh CSV_file [CSV_file …]
```

Note that the standard input will be redirected to /dev/null, and the display results will be saved to `csv_file_name.log`.

The exit status will be zero upon success, and non-zero if an error is encountered.

CATS Version E-2011.03 Introduces a New Version of Proxecco

Alex Zepka, CATS Applications Engineer

CATS Version E-2011.03 includes new versions of Paraprox and Proxecco to reflect changes implemented since the transfer of the technology from PDF Solutions to Vistec Lithography Inc. If you work with e-beam lithography tools and have the need to apply correction due to the intrinsic proximity effects, this enhancement allows you to perform your tasks more easily than in previous releases.

Paraprox and Proxecco are synonyms of proximity correction in CATS via dose modulations; the main difference being that Paraprox creates the necessary ".pec" (Proximity Error Correction) file, while Proxecco actually applies this file to the layout to assign doses to each of the fractured shapes. These doses are then taken into account by the tool writer in an attempt to undo the effects of the convolution of the point spread function with the target layout.

Paraprox 2.3

The new version of the Paraprox tool (2.3) has expanded the number of options that are available to you, and provides greater flexibility when generating the .pec file. It has also improved the usability via the new graphical interface, while improving on quality.

Both Paraprox and Proxecco are fully integrated into the CATS graphical interface with better organization of the parameters used in the creation of the .pec file, or its application during the correction step.

New parameters have been introduced as well, giving you better control over, for instance, how the dose table is set and used, or while using, applying short-range proximity correction.

Other highlights include:

- Improved integration with Sceleton by allowing the smoothing of a noisy .psf file
- Improved short-range correction
- Improved simplified error report
- A new graphical interface

Below are some specific changes from previous versions of Paraprox:

1. **Defaults file**
   Before Paraprox is run, a set of default values is defined in a text file determined by the PARAPROX_DEFAULT_TXT_FILE environment variable. This file is a new requirement for running Paraprox, but may not contain any defaults at all. Here is an example:

```
########################################
####################
# Paraprox Parameters:
#
# basic quality parameters
NUMBER_DOSES 256
QUALITY STANDARD

# use alpha correction inter if you have a
rather large beam_width
ALPHA_CORRECTION OFF

# beta correction parameters
MIN_DISTANCE 1.0            # microns
GRID 0
# microns
OUTPUT REDUCED_PHYSICAL
########################################
####################
```

   This file allows you to set up the most common settings once, and use arguments in the command line only for those settings that require a change.

2. **Protocol file**
   The protocol file is a new output file in Paraprox. It is an ASCII file containing details about a given .pec file. This is similar in format as the defaults file described above, but contains the actual settings used in the creation of the .pec file. Use the –o_prt (see below) to create a protocol file during any Paraprox run.

3. **File control**
   In previous versions of Paraprox, the –f argument flag was used to define the input point spread function, while the –o would specify the .pec file name and location. Instead, there are new options to control the input and output files used by Paraprox:

-i_txt	input point spread function (e.g., Sceleton .xrz output file)
-i_pec	input a pec file (to apply modifications)
-o_pec	output a pec file
-o_prt	output a protocol file containing ascii info about the pec file just created

Consider the following possible scenarios for using these flags:

- Input a .psf and output a .pec file:
  ```
  % paraprox –i_txt <psf file> -o_pec <pec file>
  [optional args]
  ```
- Input a .pec file, modify it and output to a new .pec file:
  ```
  % paraprox –i_pec <old pec file> -o_pec
  <new pec file> [optional args]
  ```
- Input a .pec file and output a protocol file:
  ```
  % paraprox –i_pec <pec file> -o_prt
  <protocol file>
  ```

It is also possible to output both a .pec and a protocol file during the same Paraprox run:

```
% paraprox –a 0.05 –b 1.0 –e 2.0 –o_pec <pec
file> -o_prt <protocol file> [optional args]
```

4. **New optional settings**

   Paraprox 2.3 introduces a number of new optional arguments that allow greater flexibility when creating .pec files. These new settings are:

   - Dose table parameters:
     - DOSE_SCALING. Scales all doses by this value.
     - LOWERDOSE. Minimum dose.
     - UPPERDOSE. Maximum dose.
   - Transformation parameter:
     - PIXELLENGTH_FACTOR. Modifies the beta correction grid to provide another way to improve the quality of results.
   - Short-range correction parameters:
     - MAX_ALPHA_DOSE. Extends the max value of dose table to allow higher doses to be applied during short-range corrections.
     - ALPHA_FRACTURING_SIZE. Short-range corrections are applied to shapes that are 2 µm (default). Use this setting to modify the threshold.
   - Frame parameters:
     - FRAMING. Similar to FRAME_WIDTH in previous version of Proxecco.
     - FRAMING_DOSE_RATIO. Similar as DOSE_RATIO in previous versions of Proxecco.
   - Gaussian PSF definition:
     - MULTIGAUSS. This parameter allows you to set a multi-Gaussian model for the PSF, as opposed to the double-Gaussian model that can be generated with the -a, -b and –e options. Consider the following an example using three Gaussians with sigmas 0.035, 0.8, and 10 (microns) and weights of 1, 0.25, and 0.5:

     ```
     % paraprox - MULTIGAUSS 3,0.035,1,0.8,
     0.25,10,0.5 –o_pec my_corr.pec
     ```

5. Obsolete and discarded parameters

   The following parameters have been deprecated in CATS Version E-2011.03:

   - DOSE_RATIO. Use FRAME_DOSE_RATIO instead.
   - FRAME_WIDTH. Use FRAMING instead.
   - DENSITYMODE
   - DOSE_EVALUATION

Proxecco 7.3

While Paraprox described above is a standalone executable used for the creation of .pec files, Proxecco is its CATS-based equivalent where the .pec files can be both created and applied to correct a given layout.

The new version of Proxecco is 7.3 and it provides the new functionality introduced by Paraprox 7.3 while simplifying the correction process by taking into account interdependences between settings.

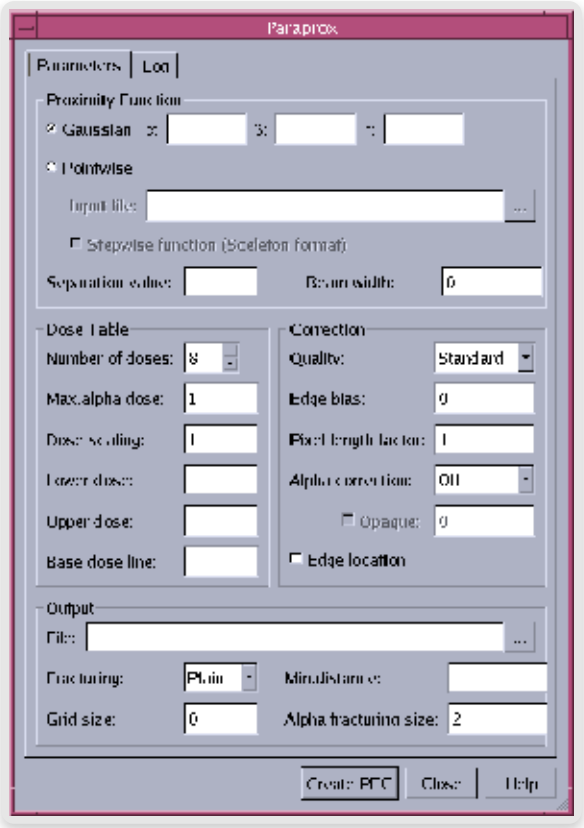


Figure 1a: Shows how the Paraprox parameters are organized in Proxecco 7.3

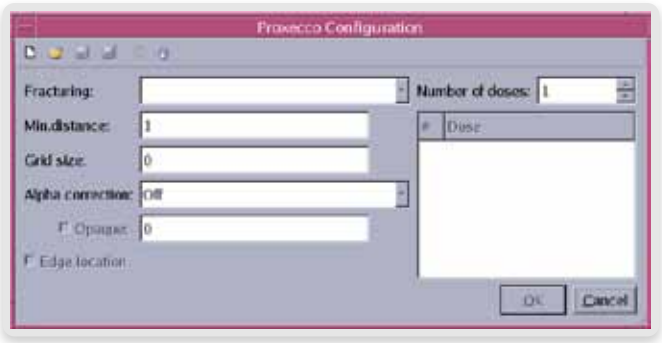


Figure 1b: Shows how the Paraprox parameters are organized in Proxecco 7.3

Once the .pec file has been created, it can be loaded via the Proxecco Settings dialog in CATS.

One main distinction between Paraprox and Proxecco is the idea of "correction modes". This provides continuity from the way corrections have been done in previous versions of Proxecco, but it has been modified to take into account the changes brought up by Paraprox 2.3. Although it does not appear in the dialog above, the PROX CORRECTION_MODE command is valid in a .cinc file and at the CATS command line.

Table 1: Specifying the setting of CORRECTION_MODE corresponds to applying the following settings

Correction Mode	Opaque	Edge Location	Framing
STANDARD	OFF	OFF	OFF
EDGE_LOCATION	OFF	ON	OFF
FRAME	OFF	OFF	ON
EDGE_FRAME	OFF	ON	ON
OPAQUE	ON	ON	OFF

Table 2: This mapping can be seen in reverse by looking at which choices selected in the Proxecco Settings dialog will correspond to the specific CORRECTION_MODE settings

Opaque	Edge Location	Correction Mode
OFF	OFF	STANDARD
OFF	ON	EDGE_LOCATION
ON	OFF	Give a warning and turn "Edge Location" on
ON	ON	OPAQUE

CATS QA Random Testing

Andy Frazer, CATS QA Engineer

In order to ensure the highest quality of the CATS product, the CATS QA group maintains several types of regression tests, including unit tests, flow tests, STAR tests, stress tests, performance tests, fixed (non-random) tests and random tests. This article discusses the importance and trade-offs of random regression tests to the CATS QA test environment.

Before discussing the role of random tests within a comprehensive test strategy, it is important to understand both the value and the limitations of fixed (non-random) test cases. CATS supports many different combinations of input formats, output formats, production (PRODUCE) formats, as well as fracture settings such as HEIGHT, WIDTH, SCALE, SIZING, and GROW. Functionality such as VSB12 DIRECT_OUTPUT requires additional commands such as XBLOCK, YBLOCK, XCELL, and YCELL. Non-random regression testing focuses on the test cases that repeatedly run a carefully selected set of these values. The tests are executed with the same commands and values every time a new build is created by the CATS R&D team.

However, even if the above test strategy was limited to the unrealistic scenario of testing only a single input file, the test time required to test all meaningful permutations of all fracture settings would be prohibitive. The solution is to employ a hybrid test strategy that combines a set of fixed and random regression tests. For example, the fixed regression tests focus on testing only carefully selected input files and parameter values. On the other hand, random tests focus on catching a wide range of unexpected combinations of parameters.

Table 1: Comparison between fixed and random test suites

	Fixed Test Suite	Random Test Suite
Functionality Coverage	Lower	Higher
QA Development Time	Higher	Lower
Time-to-test-failure	Shorter	Longer
Focus	Expected cases	Less-expected cases

For example, suppose that one function in the product supports ten different user-controllable commands: $P_1..P_{10}$. For CATS, those commands could be the examples above (such as HEIGHT, WIDTH, SIZE, and so forth). If the consensus was that most customers will use one of only five common values ($V_1...V_5$) for each of those commands, the set of command-value pairs (P_n,V_m) to be tested would fill a 10x5 matrix. To test each of those matrix points independent of all other command-value pairs could be accomplished with only five test cases. However, many software problems are a function of combinations of parameters, not individual parameters.

It has been well-established in the testing literature ("Practical Combinatorial Testing", *LogiGear Magazine*, Nov 2010) that the majority of combinational failures can be reproduced by varying only two parameters (see Example 1), as opposed to combinations of three or four, or more. In the previous example, complete pair-wise testing might still require an unrealistic number of test cases. However, random pair-wise testing would focus on quickly testing as many pairs of command-value matrix points as possible, instead of testing every permutation of all approximately 9.7 million (5^{10}) command-value combinations.

Example 1: Pair-wise test configurations

Test case	COMPACT	Direct_output	Do/Distribute
1	Yes	Yes	Do
2	Yes	No	Distribute
3	No	Yes	Distribute
4	No	No	Do

In a real random testing application, CATS QA first identifies which commands will be assigned random values. Next, some rules are determined for randomly assigning values to those commands. For example, the upper and lower bounds might be set on the range of random values. Or, conditions might be set such that random values are not generated for one command that is incompatible with a randomly selected value for another command. Once all of the random values have been selected, the test is executed. If the randomly selected values trigger a failure (possibly a crash or an invalid result), a record of these randomly selected input values must be saved so that a QA engineer can investigate the problem. Once the problem has been investigated and corrected by R&D, these values should be used to create a fixed test case that will be executed on a nightly basis.

In addition to testing randomly-selected CATS command values such as HEIGHT, WIDTH, SIZE, GROW, and so forth, the CATS QA team has also found other uses for random testing. One example is to test the situation where an error may occur along a specific edge of fracture region. Another example is the situation where the width of the fracture region is not an even multiple of the stripe WIDTH. Both of these situations can be tested by setting the parameters to select many small regions of the die each with a randomly selected origin and randomly selected X and Y dimensions. By changing the location and size of the fracture window, the boundary conditions that can impact the results can be modified. A small window may uncover an incorrect behavior that will not show up when fracturing the entire chip. Each test case only fractures a small region, resulting in a very fast turnaround time, allowing many iterations of the randomly-selected region.

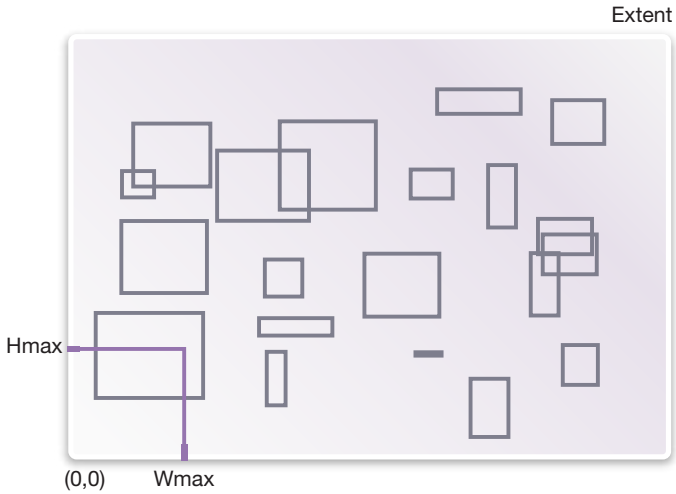


Figure 1: A sample of input regions selected from the same input chip. Each input region has a randomly-selected origin, and a randomly-selected width and height of $W<W_{max}$ and $H<H_{max}$

Another application of random testing is the case where QA wants to test an interesting, but very large, input file (either GDS or OASIS) using limited hardware resources in a server farm. Assume the server farm has a limitation of CPU cores per test case `(C)` and a turnaround time limit of `T`. Also suppose that a DP fracture of this entire input file using `C` cores would require `100*T` hours. One solution is to subdivide the input die into 100 individual test cases. However, another solution would be to randomly select small regions of the die that can be fractured within the given hardware `(C)` and time `(T)` constraints. With the second approach, a small number of test cases could achieve high test coverage of the die when distributed across multiple regression test cycles.

There is no realistic approach to guarantee full test coverage of a complex system. A well-designed test strategy includes the benefits of many different test strategies. Random testing is an important strategy that complements unit testing and directed functional testing.

XML Jobdeck Support

Mike Behnam, CATS Application Engineer

CATS Version E-2011.03 supports Extensible Markup Language (XML)-based jobdecks, which simplify the MDP dataflow and optimize connectivity. This new support continues CATS' adaptation of the P10 syntax into XML format defined by the SEMI P10 "Mask Order" specification.

CATS treats XML Jobdecks (XJBs) as just another jobdeck format with the same creation, viewing, editing, and saving capabilities provided in other jobdeck file formats, such as CJB and JB.

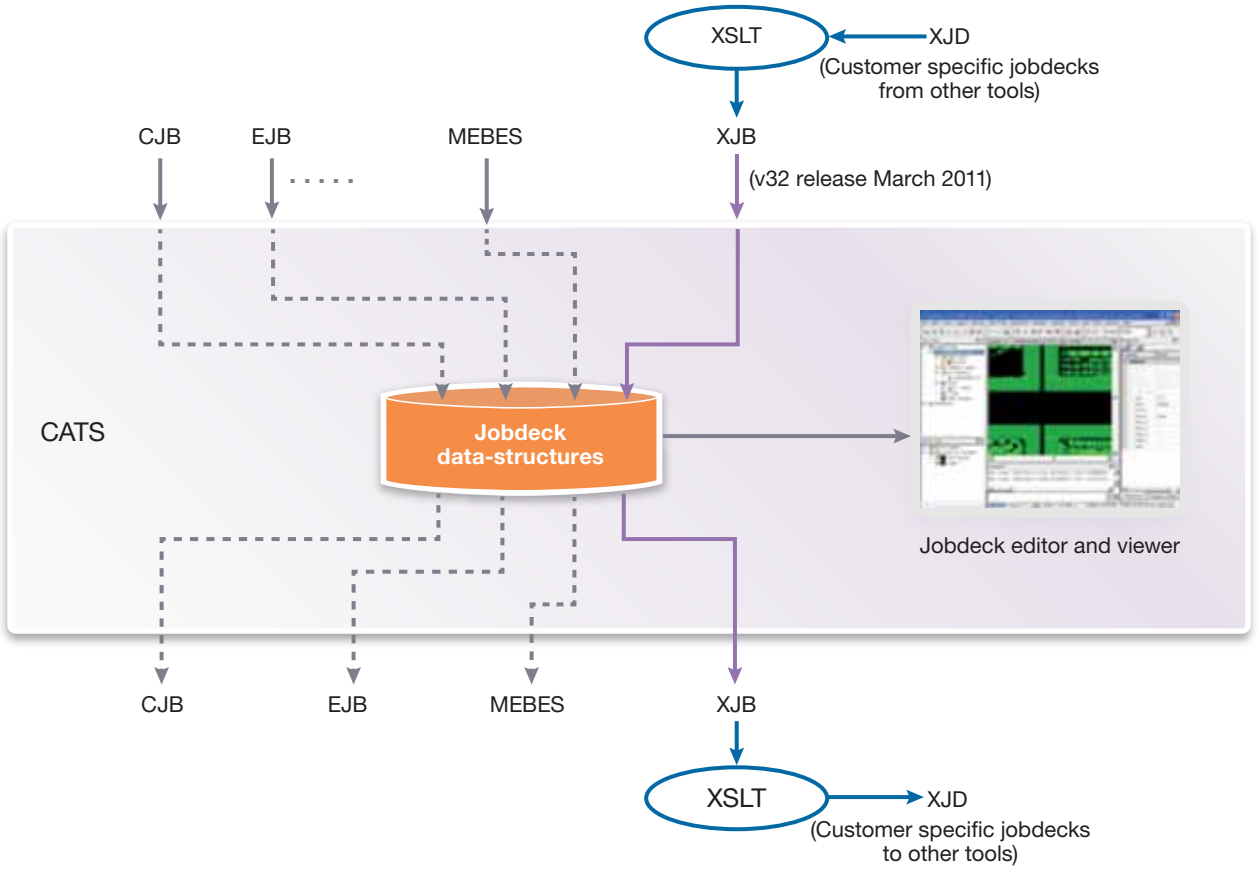


Figure 1: XML jobdeck flow

XML jobdeck support offers a number of benefits, including:

- Generic, standard and open formats instead of proprietary formats, eliminating the need for custom reader/writers
- Flexibility and ease of maintaining backwards compatibility; the addition of new nodes/attributes to the file does not affect existing versions
- Simplified integration into customer flows
- Easier conversion to other jobdeck formats, which can be performed using XML Style Sheets - XSLT. XSLT can also be used to create rich HTML documents for jobdeck, metrology, and other purposes

The development of XJB jobdecks was based on the data CATS uses for CJB jobdeck format, such as titles, chip patterns, placements, and Marking data.

Functionality:

Format	Input	View	Edit	Write	Smash	Normalize	Orient	Marking	Inspection
CATS jobdeck	✓	✓	✓	✓	✓	✓	✓	✓	✓
Extended jobdeck	✓	✓	✓	✓	✓	✓	✓	✓	✓
XML jobdeck	✓	✓	✓	✓	✓	✓	✓	✓	✓

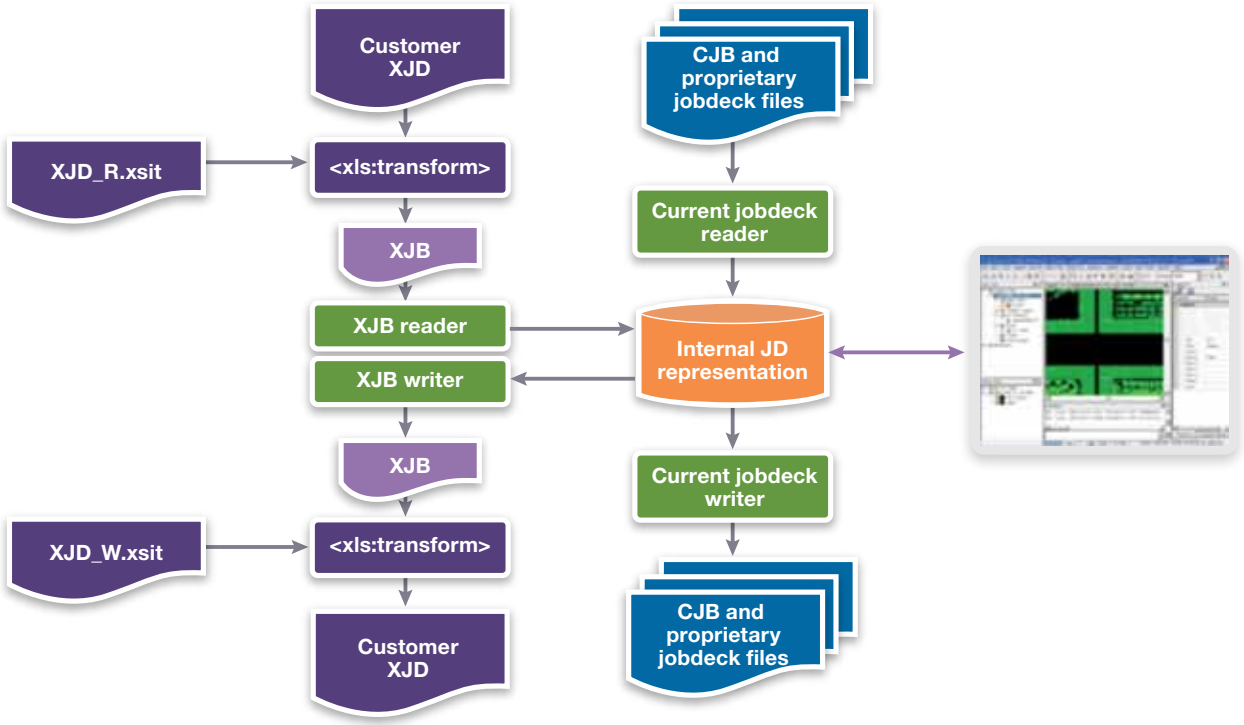


Figure 2: XML jobdeck implementation

CATS customers can open or input any CATS-supported format jobdeck, such as a CATS .cjb or an extended MEBES jobdeck, perform the viewing, and edit as illustrated in Figure 1. Once edits are complete, you can save the jobdeck in this new CATS-supported XML/XJB format. Since .xjb files are now considered to be another supported jobdeck format to CATS, you can also input a .xjb (CATS XML format) file to view, edit, and save it back in the same XJB format.

CATS is 25 continued from page 1

tool to transcribe design data into a format that early mask writers could read, CATS now has the widest support of all MDP tools for mask writing, inspection and metrology.

For 25 years, CATS has maintained its leadership position by keeping pace with the relentless march of semiconductor technology from 2um to 20nm. Using technological innovations, such as multi-variate optimized slicing, orientation independent data analysis and shot count aware fracture, CATS has risen to meet every mask data preparation challenge.

Technology scaling to smaller geometries requires very tight control on variability of critical dimensions (CD) on masks. As designs get more complex, the mask data size increases exponentially, directly impacting data prep, data transfer and write times for masks, all of which contribute to mask turnaround time and cost. CATS meets these aggressive requirements and delivers fractured design data with rigorous symmetry and uniformity in fractured shapes and is optimized for fewest slivers. This superior quality-of-fracture enables tight control over CDs on masks and reduces the overall mask write times on advanced VSB mask writers.

CATS is highly scalable on large computer clusters in a distributed processing configuration. This offers users the flexibility of optimizing for data preparation turnaround time versus cost of computing infrastructure. To achieve this scalability, CATS has implemented sophisticated data handling and transfer techniques which eliminate bottlenecks in the overall data flow.

In short, CATS has maintained its leadership position through continuous innovations which address critical customer needs as technology scales to smaller and smaller geometries.

This newsletter offers details on new capabilities for verification and metrology in line with our customers' expectations to maintain CATS as the most comprehensive solution for mask data preparation. The CATS team appreciates your support of the product and if you have any comments, concerns or questions please email us at the address below.

Anjaneya Thakar
CATS, Product Marketing Manager

For questions about CATS please send email to cats@synopsys.com.

SYNOPSYS®
Predictable Success

Synopsys, Inc. • 700 East Middlefield Road • Mountain View, CA 94043 • www.synopsys.com