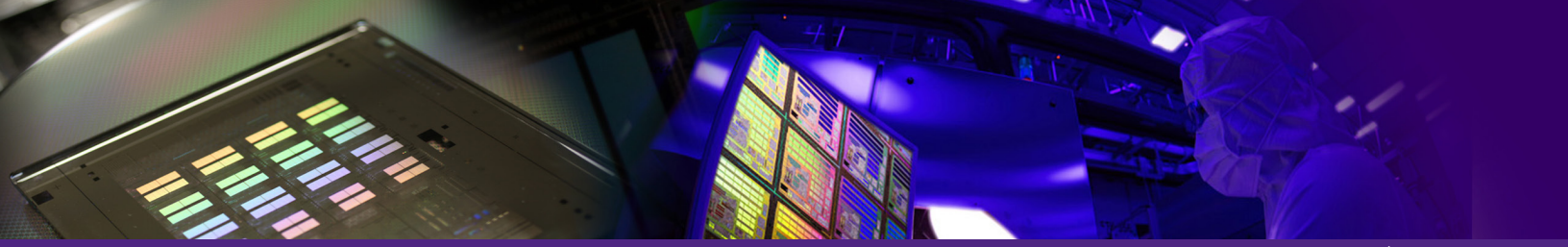


Newsletter for data prep engineers

Issue 5 | Summer 2014

CATS news

In This Issue

Using VSB12 Common Cell for File Size Reduction....1

JD MRC: Simplifying Selection of Jobdeck Chip Names via Regular Expressions.....1

Using Curvilinear Pattern Detection to Select Mask Writer.....3

Auto Testing Script 5

QR Code Generation 9

Metrology: New Ability to Add Marks on Rotated or Staircase Data 11

Dear CATS User:

2014 is shaping up to be a great year for CATS. We have seen wider adoption by key customers including leading semiconductor foundries, merchant mask shops and semiconductor companies. We would like to thank you for making CATS your solution of choice for mask data preparation. In June 2014 we released the latest version of CATS: J-2014.06. Since the last release we delivered higher performance, better quality of results and improved functionality for advanced technology nodes. It is our hope that you will find this newsletter informative and useful and we look forward to your feedback.

Anjaneya Thakar
CATS, Product Marketing Manager

Using VSB12 Common Cell for File Size Reduction

Alvaro Palma, CATS R&D Engineer

CATS now supports the implementation of the Common Cell feature present in NuFlare's VSB12 format spec. It takes advantage of the presence of repetitive data across different VSB12 frames (not within a given frame(*)) to remove such repetitions and replace them with references to a single instance placed in a newly created file, called the "common cell" file.

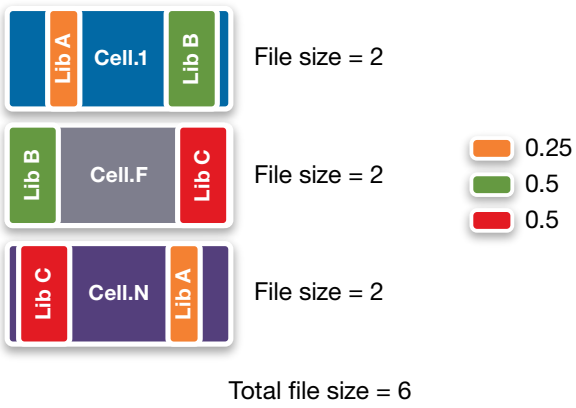


Figure 1: Original VSB12 output containing cells present in more than one frame ("common cells")

Continued on page 2

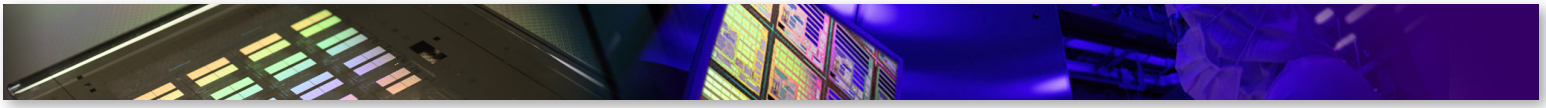
JD MRC: Simplifying Selection of Jobdeck Chip Names via Regular Expressions

John Valadez, CATS CAE

This new capability provides an easy and fast method to aid in deciding the kind of technology to use to write photomask

Because jobdecks in production today can reference hundreds of chips, the need to list each chip by name or number in a script or .cinc file can become too cumbersome. This article describes a TCL procedure which provides CATS JD MRC users a simpler approach to specifying the names of the chips to be processed.

Continued on page 2



Using VSB12 Common Cell for File Size Reduction
continued from page 1

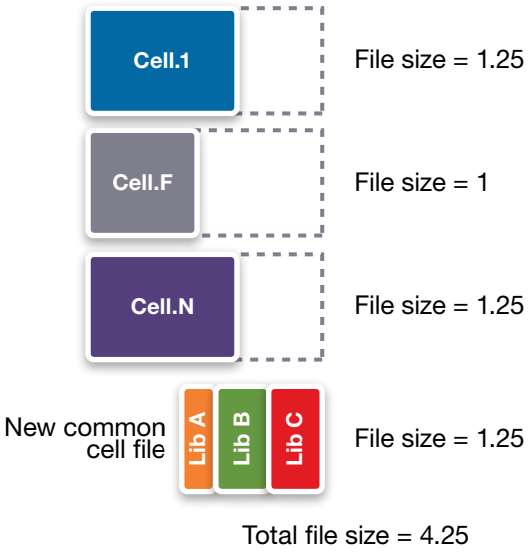


Figure 2: Final VSB12 output, after common cell extraction, showing the file size reduction

(*) Removal of duplicated cells ***within*** a frame is already performed in existing CATS VSB12 flow.

Besides the described feature, CATS also provides a mode called Fast Estimate. This mode provides a quick estimate of the potential file size reduction without modifying the data. If the user decides the estimated reduction is sufficient, they can then run a separate standalone application (also provided) to reduce file size using common cells.

The feature is enabled by using a single CATS command:

```
VSB12 COMMON_CELL [YES/FAST]
```

Where:

- YES extracts common cells and outputs a reduced file size with common cells.
- FAST provides a quick estimate of the potential file size reduction. Please note that FAST mode estimates only the file size reduction, not how long it would take to perform the reduction (the TAT increase).

In summary, VSB12 Common Cell is a new feature introduced to aid the user in reducing file size without compromising data quality. ■

JD MRC: Simplifying Selection of Jobdeck Chip Names via Regular Expressions
continued from page 1

Normally, when executing JD MRC, the user specifies which jobdeck chips are to be processed. By default, a wildcard is used (shown in Example 1). This specifies that all chips will be processed within the given JD MRC ruleset.

Example 1:

```
JD MRC 1 MODE CHIP
JD MRC 1 CHIP_LIST *
```

The user can pass a list of chips, by name or index number, to be used in a given JD MRC ruleset. This can be done using either 1) the CHIP_LIST option when executing in JD MRC run mode CHIP, or 2) CHIP_FILTER option in JD MRC run modes, CHIP_OVERLAP or CHIP_INTERACT. Example 2 below shows all three run modes each with a set of chips to process specified by chip name.

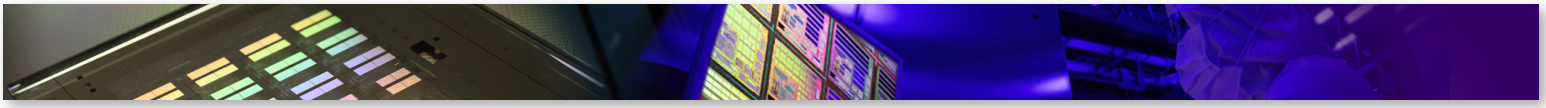
Example 2:

```
JD MRC 1 MODE CHIP
JD MRC 1 CHIP_LIST PRIME1 PRIME2 PRIME3
!
JD MRC 2 MODE CHIP_INTERACT
JD MRC 2 CHIP_FILTER PRIME1 FRAME1
!
JD MRC 3 MODE CHIP_OVERLAP
JD MRC 3 CHIP_FILTER FRAME1 BARCODE1
```

If chips are specified by name, CATS immediately associates the name with the jobdeck chip number, which is then passed to the JD MRC engine. In Example 2, this could also be shown as:

```
JD MRC 1 MODE CHIP
JD MRC 1 CHIP_LIST 1 2 3
!
JD MRC 2 MODE CHIP_INTERACT
JD MRC 2 CHIP_FILTER 1 4
!
JD MRC 3 MODE CHIP_OVERLAP
JD MRC 3 CHIP_FILTER 4 5
```

A regular expression parser is provided for JD MRC via TCL library files. Please refer to the chapter on CATS scripting in the CATS Reference Manual for more information on regular expressions. ▶



The user must first setup the TCL libraries in order to load the necessary functionality into CATS. This is done via script or TCL command pane, in the following manner:

```
package require jdmrclib
package require toolslib
```

Please reference the CATS Command Manual for more information on TCL Interface for MRC Commands.

After the libraries are loaded, the `tools::jd::chiplist` function and its `–name` option are the keys that provide advanced filtering capabilities. Example 3 shows the TCL equivalent to Example 2 above.

Example 3:

```
jdmrc::mode CHIP “CHIPS = [tools::jd::chiplist -name
{PRIME[1|2|3]}]” { }
jdmrc::mode CHIP_INTERACT “CHIPS =
[tools::jd::chiplist -name {.*ME1}]” { }
jdmrc::mode CHIP_OVERLAP “CHIPS =
[tools::jd::chiplist -name \ {^FRAM.*|^BAR.*}]” { }
```

These commands are translated into their normal command line equivalents:

```
JD MRC 1 MODE CHIP
JD MRC 1 CHIP_LIST 1 2 3
!
JD MRC 2 MODE CHIP_INTERACT
JD MRC 2 CHIP_FILTER 1 4
!
JD MRC 3 MODE CHIP_OVERLAP
JD MRC 3 CHIP_FILTER 4 5
```

The regular expression is placed inside the `{}`'s and passed via the `–name` option. Because regular expressions are used, there are other combinations that could work as well. It all depends upon the creativity of the user.

Using the `tools::jd::chiplist` TCL function will save users time in two ways. First, in runset creation, since users do not have to list each chip one at a time. And second, this function ensures the correct set of chips are processed which can save valuable runtime. ■

Using Curvilinear Pattern Detection to Select Mask Writer

Sebastian Munoz, CATS R&D

Introduction

There are two main technologies commercially available to write photomasks: Raster Scanning and Variable Shaped Beam (VSB).

Unlike Raster machines, in which mask writing time primarily depends on the extent of the data being written and its resolution, the time that a VSB machine requires to perform such a task greatly depends on the number of shots needed. Given this, is important to know the location of all curvilinear patterns, as they tend to require a great number of VSB shots be written to a mask. Knowing the location and size of such patterns aids in deciding whether it is convenient to use VSB technology, or whether alternatives should be explored.

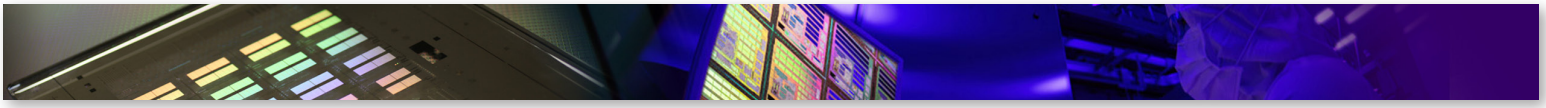
The Shot Count Estimator has been developed to provide the user with a fast method of detecting where circular patterns are likely to be located in their design. Since this tool will be used before the MDP process actually begins, it imposes an extra step to the whole process. Furthermore, as it is intended to save time (when choosing for one technology over the other), the extra time spent using this tool should remain low. Therefore, the focus of this tool is speed over accuracy. The implementation lies in the execution of a simple and fast heuristic rather than a complex and accurate algorithm.

Inner Workings

All polygons in a design are internally interpreted as a collection of primitive shapes. CATS does not know what the original polygons were, so to decide whether a group of figures corresponds to a circular pattern, some kind of approximation must be made.

Although a convex hull algorithm would find circular patterns with a high degree of accuracy, the performance requirements forces a sacrifice of accuracy over speed.

The heuristic chosen to accomplish this task is to find the ratio between rectilinear and non-rectilinear data in each stripe. If a majority of non-rectilinear data is found, a circular pattern is assumed to exist in the stripe, and in such cases, a rectangle of the size and location of the stripe is placed in the output file. This rectangle points to the stripe location in which a circular pattern probably lies. Also, while this command is being executed, CATS gathers and consolidates the information of the amount of primitives present in the entire design, as well as the percentage of area each kind of figure covers. This information is displayed when the command has finished processing. ►



It must be noted that the number of non-rectilinear primitives that are produced to approximate a circular pattern depends on the resolution used. As the resolution decreases, so does the quality of the approximation and the number of non-rectilinear primitives generated, which are replaced by rectangles. A very low resolution may negatively impact the QoR of this heuristic, as it may not detect circular patterns present in a design.

Use Model

This feature was implemented as a function of CATS, which means in order to be able to use it; CATS must be configured to run `FUNCTION SHOT_ESTIMATE`.

This feature works with standard SP, DP and MT modes.

Example 1:

In the following example, a test design has been used that consists of Manhattan data near the top and small circles near the bottom.

This tool should pinpoint the location of the circles at the bottom of the design. Also, because this relies on a heuristic, some false positives are expected, particularly in the cases where the stripes contain a majority of non-rectilinear primitives that are not part of a circular pattern.

```
Format FLAT
Produce CFLT
Res 0.001
Height 10
Width 10
Inp /slowfs/cats9/sebastia/catsTmp/TEST_DESIGNS/
VC316O6A1.00
Out ./Circle_Map.cflt
Function SHOT_ESTIMATE
```

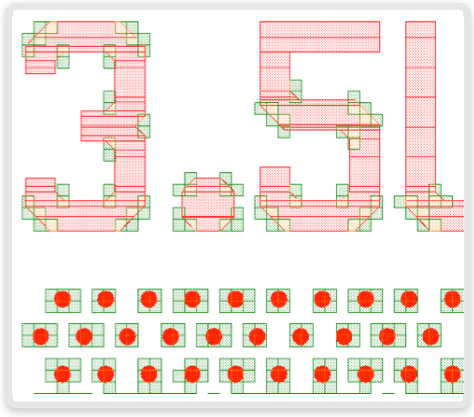


Figure 1: Curvilinear pattern detection output

It can be seen that at this resolution, all circular patterns were correctly detected, but some false positives were detected as well.

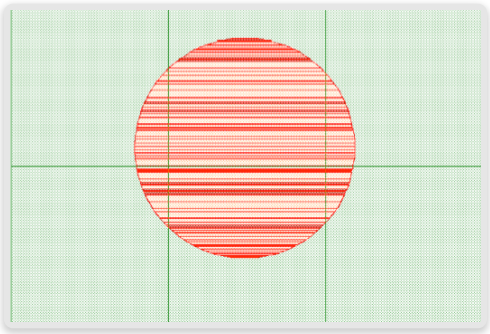


Figure 2: Curvilinear pattern zoom in

When zooming into one of the circular patterns, it can be seen that it is composed of a large number of non-rectangular figures.

Example 2:

Now consider what happens when you decrease the resolution of the operation to 1µm.

```
Format FLAT
Produce CFLT
Res 1
Height 10
Width 10
Inp /slowfs/cats9/sebastia/catsTmp/TEST_DESIGNS/
VC316O6A1.00
Out ./Circle_Map.cflt
Function SHOT_ESTIMATE
```

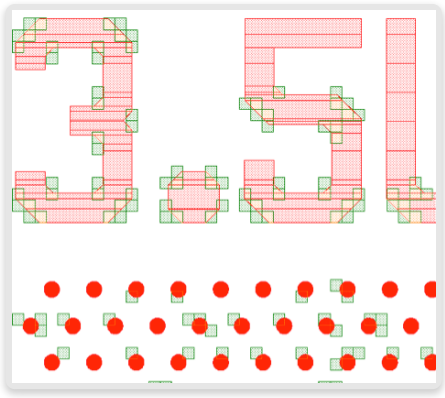
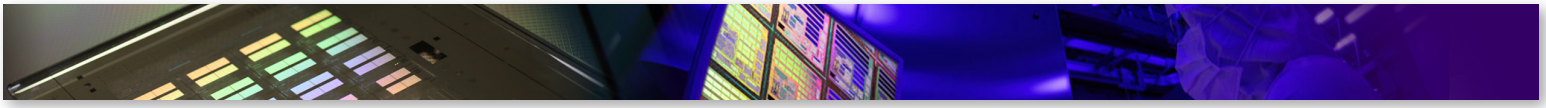


Figure 3: Curvilinear pattern detection at low resolution

It can be noted that not all circular pattern were detected, because internally, many of those circles were fractured to mainly rectangles.



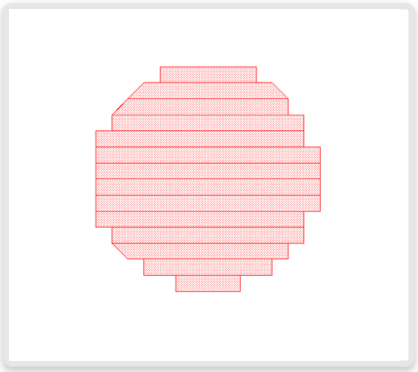


Figure 4: Curvilinear pattern zoom in at low resolution

When zooming into one of the circular patterns that were not detected, it can be noted that there are more rectilinear than non-rectangular figures. In these cases, the heuristic considers that no circular pattern is present in the stripe.

Conclusion

This tool consists of a fast and easy-to-use way of detecting circular patterns in a design, which helps in deciding the kind of technology that should be used to manufacture a mask.

Although its accuracy depends greatly on the resolution used, at fracture resolution, the tool displays a good performance in terms of speed and precision. ■

Auto Testing Script for Customer Qualifications

Tokiharu Sekiya, CATS CAE

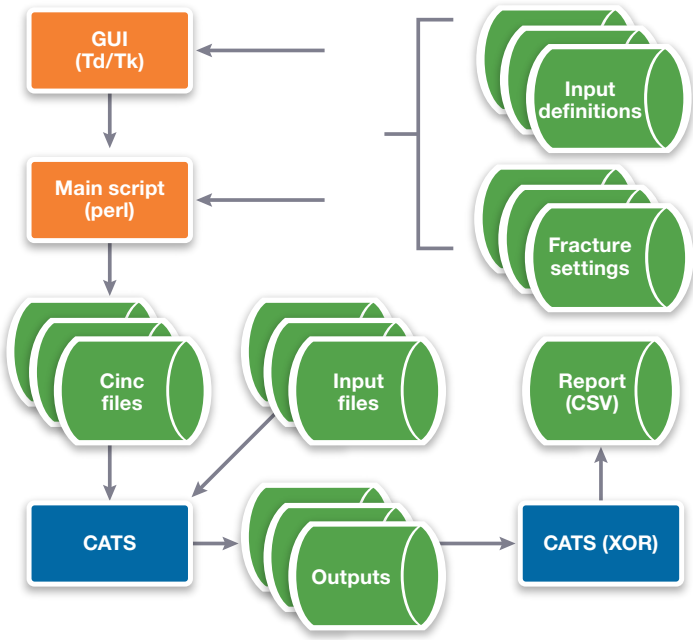
This article describes an automated script that can reduce CATS qualification times, reduce associated errors, and free up user time. The reader should be aware that the auto testing script is not officially supported and is provided only as an example of how a wrapper script can automate a CATS process. Users are free to modify the script to match their individual needs. Depending upon your level of qualification stringency, a fully manual process may be time consuming.

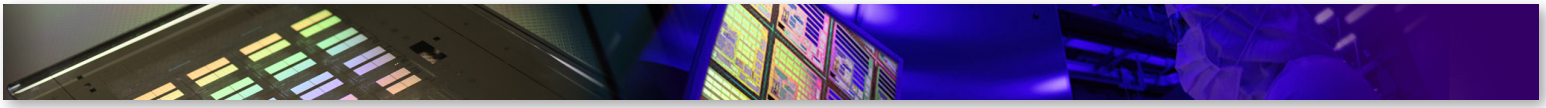
The script provides a user friendly interface to setup various qualification criteria ranging from versions, parameters, and even complete flows. It can help in qualifying two different CATS versions or two different sets of parameters in the same CATS version.

It also simplifies the laborious data collection stage as well as the key pass/fail stage of the qualification process. The script automatically does an XOR against the baseline and measures performance metrics, such as TAT and file size. Automation of these backend steps is key to reducing the CATS qualification time and freeing up user time, since the test cases and permutations can number in the tens and even thousands of actual runs each with their own associated log files.

Flow

The data flow is shown in the diagram below. First the input definitions (filtered subset of data from a larger data file) and fracture settings (intended CATS flow to test) must be defined. These can be defined in a user friendly GUI based on Tcl/Tk. The GUI then automatically communicates this to the main script. The script generates.cinc files that use the definitions specified earlier, which then serve as instructions for an autonomous run of CATS. CATS uses these instructions to convert the input files to the output file in two serial runs each defining a different configuration. Finally, an XOR is run on the two output files and a report file is generated in CSV format. ▶





Capabilities

The following are some of the key capabilities of the auto testing script.

- Compare different CATS versions
- Compare different fracture settings
  Example: Compaction YES vs. Compaction REDO Rule WEXTEND vs. Rule WPARAGON
- Execute Boolean operation
- Compare performance
  Example 1: Threads 1 vs. Threads 4 vs. Threads 8
  Example 2: DP 10 vs. DP 20 vs. DP 40
- Measure elapsed time of command execution
  Example: Input command

Example

Below is an example of a run using the auto testing script.

Input Definition: This is part of the .cinc file. It requires input file name (INPUT command), top structure name (STRUCTURE), layer/datatype number (LAYERS), data window (LIMITS), scale (SCALE) and orientation (ORIENTATION). If STRUCTURE/LAYERS/LIMITS/SCALE/ORIENTATION commands are not specified, the result depends on CATS default behavior. This information is entered in the GUI as shown in the GUI section below this example. This section also shows the final log file CSV that is generated.

```
OASIS file:
Input /home/testcases/OASIS/flyingcats2.oas
Structure BIGARRAY
Layers 1, 3-5
Limits (INPUT -40.999,-28) (INPUT 1636.999,1130)
Scale 4
Orientation 90
CFLT file:
Input /home/testcases/CFLT/BIGARRAY_00.cflt
```

Fracture Setting: This is part of the `.cinc` file.

```
CFLT fracture:
Format FLAT
Resolution 0.001
Height 500
Width 500
VSB_T fracture:
Format FLAT
Produce VSB_T
Resolution 0.001
Height 500
Width 500
VSB_T SLIVER_WIDTH 0.1
VSB_T MAX_SHOT_SIZE 2
VSB_T CD_DIMENSIONS 0,0,10
VSB_T SHOT_OPT BASIC
VSB_T HALO_SIZE 12
VSB_T VERIFY N
```

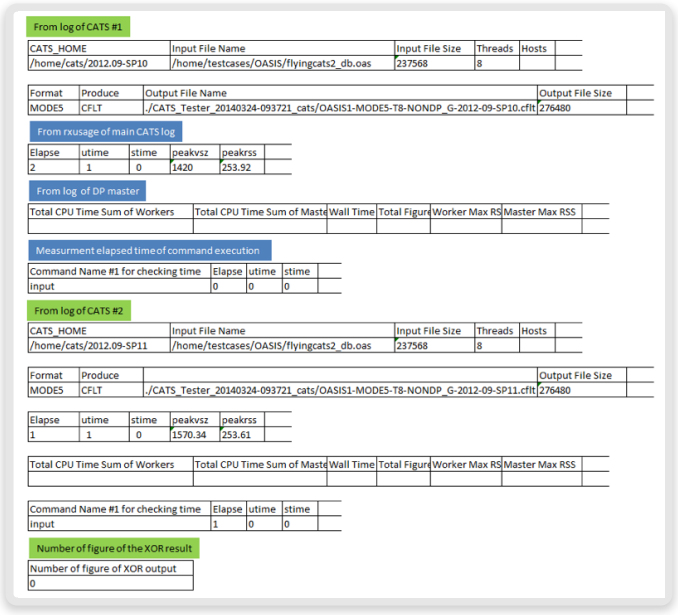
The following is an example .cinc file generated by the script.

Text Legend:

Green: Fracture Setting
Red: Measuring command execution time
Blue: Input Definition
Purple: Threads/DP settings

```
Clear
Pregrid No
Precut NO
Allocate_rects 100000
Allocate_traps 100000
Allocate_space 3000000
Format MODE5
Compact YES
Produce CFLT
Join NO
Twosided YES
Rule WPARAGON
Border OUTSIDE
Resolution 0.025
Height [65536]
Width [32768]
Fieldheight 1024
Fieldoverlap 10
Subfieldheight 64
Subfieldwidth 64
Subfieldoverlap 0.5
say "&gt;&gt;&gt;&gt;&gt; START CHECK TIME : input  &lt;&lt;&lt;&lt;&lt;"
date
peak
say "&gt;&gt;&gt;&gt;&gt; &lt;&lt;&lt;&lt;&lt;"
Input ./CLIB/flyingcats2_db.clib
say "&gt;&gt;&gt;&gt;&gt; &gt;&gt;&gt;&gt;&gt; END CHECK TIME &lt;&lt;&lt;&lt;&lt; &lt;&lt;&lt;&lt;&lt;"
date
peak
say "&gt;&gt;&gt;&gt;&gt; &gt;&gt;&gt;&gt;&gt; &gt;&gt;&gt;&gt;&gt; &lt;&lt;&lt;&lt;&lt; &lt;&lt;&lt;&lt;&lt; &lt;&lt;&lt;&lt;&lt;"
Structure BIGARRAY
Limits (INPUT -40.999,-28) (INPUT 1636.999,1130)
Layers -
Scale 4
Orientation 0
Compact REDO
Threads 8
Output ./CATS_Tester_20140324-092142_cats/OASIS1-
MODE5-T8-NONDP_G-2012-09-SP11
ignore do
peak
quit
```

The following is an example log file (CSV).



From log of CATS #1

CATS_HOME	Input File Name	Input File Size	Threads	Hosts
/home/cats/2012.09-SP10	/home/testcases/OASIS/flyingcats2_db.oas	237568	8	

Format	Produce	Output File Name	Output File Size
MODE5	CFLT	./CATS_Tester_20140324-093721_cats/OASIS1-MODE5-T8-NONDP_G-2012-09-SP10.cflt	276480

From rxusage of main CATS log

Elapse	utime	stime	peakvsz	peakrss
2	1	0	1420	253.92

From log of DP master

Total CPU Time Sum of Workers	Total CPU Time Sum of Maste	Wall Time	Total Figur	Worker Max RS	Master Max RSS

Measurment elapsed time of command execution

Command Name #1 for checking time	Elapse	utime	stime
input	0	0	0

From log of CATS #2

CATS_HOME	Input File Name	Input File Size	Threads	Hosts
/home/cats/2012.09-SP11	/home/testcases/OASIS/flyingcats2_db.oas	237568	8	

Format	Produce	Output File Name	Output File Size
MODE5	CFLT	./CATS_Tester_20140324-093721_cats/OASIS1-MODE5-T8-NONDP_G-2012-09-SP11.cflt	276480

Elapse	utime	stime	peakvsz	peakrss
1	1	0	1570.34	253.61

Total CPU Time Sum of Workers	Total CPU Time Sum of Maste	Wall Time	Total Figur	Worker Max RS	Master Max RSS

Command Name #1 for checking time	Elapse	utime	stime
input	1	0	0

Number of figure of the XOR result

Number of figure of XOR output
0

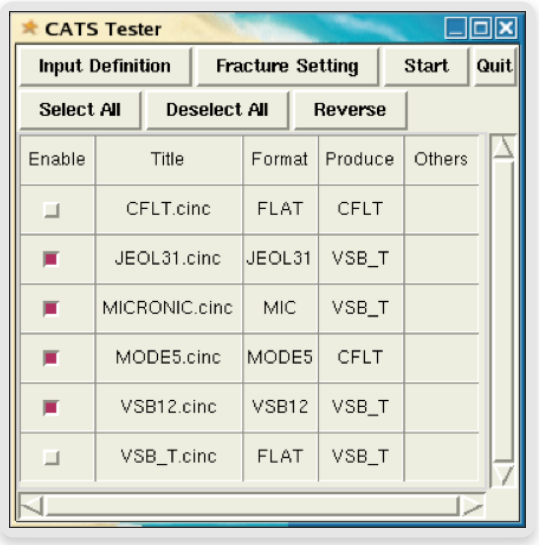


Figure 3: For this example, 4 settings (JEOL31, MICRONIC, MODE5, VSB12) are selected as fracture settings

GUI

Below is an image showing an example of the input definitions.

Input Definitions View: This view lists input definitions, and is used for selecting definitions for the test. Notice it allows multiple Input Definitions to be defined and managed individually. Each definition can be turned on or off. When they are turned off (not enabled) they are prevented from being used later in the Execution View.

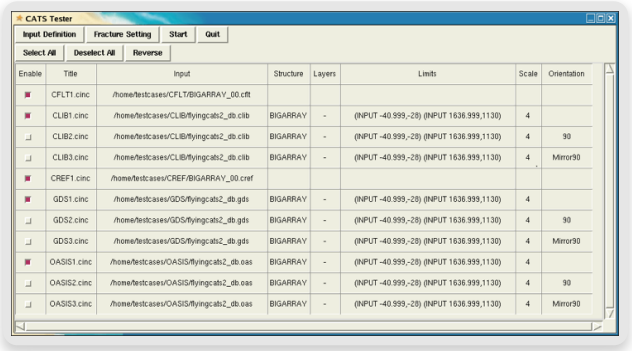


Figure 2: For this example, 5 files (CFLT, CLIB, CREF, GDS, OASIS) are selected as input files

Fracture Settings View: This view lists fracture settings, and is used for selecting settings for the test. Similar to the controls of the input definitions, each Fracture Setting can be toggled on or off to be excluded from the Execution View.

Execution View: This view is used for the following test settings, and starts the test. The Execution View controls how the auto testing script will run the input and fracture settings. A particular set of tests is started by populating the two subsets of the Input Definitions and Fracture Settings from the larger pool defined earlier in their respective GUI sections. Next the Boolean operations consisting of a Boolean and a Primary and optional Secondary space is defined. Each space consists of one of the Input Definitions while each Boolean operations uses one fracture setting to determine the particular output format configuration to test. Also a random test can be defined for each Boolean where each run which consists of a permutation of each input file with itself and all other input files.

1. Listing selected input definitions and fracture settings for double check
2. Defining Boolean operation. There are two type Boolean operation tests; specific and random tests. For specific tests, the script executes the specified Boolean operation for the specified primary and secondary files. For random tests, the script executes all combinations of specified input definitions. Example) there are three input definitions specified. The script will execute input 1 vs. 1, 1 vs. 2, ….., 3 vs. 1, 3 vs. 2 and 3 vs. 3.
3. Defining CATS directory. Two CATS home directories for the test.

4. Defining optional command for both CATS version. If you would like to compare different fracture settings on the same CATS version, you can add different settings (commands) for both CATS version here and you need to set the same CATS version for both CATS home directories in step 3. Example) Add“Compaction REDO” for CATS #1 and “Compaction YES” for CATS #2.
5. Defining optional CATS command for XOR step. If you would like to remove small difference in XOR, you can add the “Grow -….” command here.
6. Defining running condition. MT/DP settings are defined here. ODE_BATCH mode can be enabled here. If “Running twice” is on, the script will execute CATS twice.
7. Defining setting for performance test. If you would like to execute a performance test, you can specify DP hosts files and/or threads.
8. Defining CATS command for measurement elapse time of execution. The script will measure elapsed time by peak command for the specified command.
9. Defining clean up mode for output files of the test. If “Force” is selected, the script will remove all fracture output files. If “XOR is clean” is selected, the script will remove fracture output file when XOR is clean. These settings are provide to reduce disk usage during the test.
10. Defining c-shell script for the test. The GUI script will generate a c-shell script file to run the main perl script. You can keep this c-shell script file for re-running.
11. Start the test.

The example shows the general steps for implementing the auto testing script. The flow chart shows the relationship between the needed information and the resulting intermediate and final outputs. With this information, a user can make a preliminary judgment on whether this system is worth further investigation.

This article described an automated script that can reduce CATS qualification times, reduce associated errors, and free up user time. ■

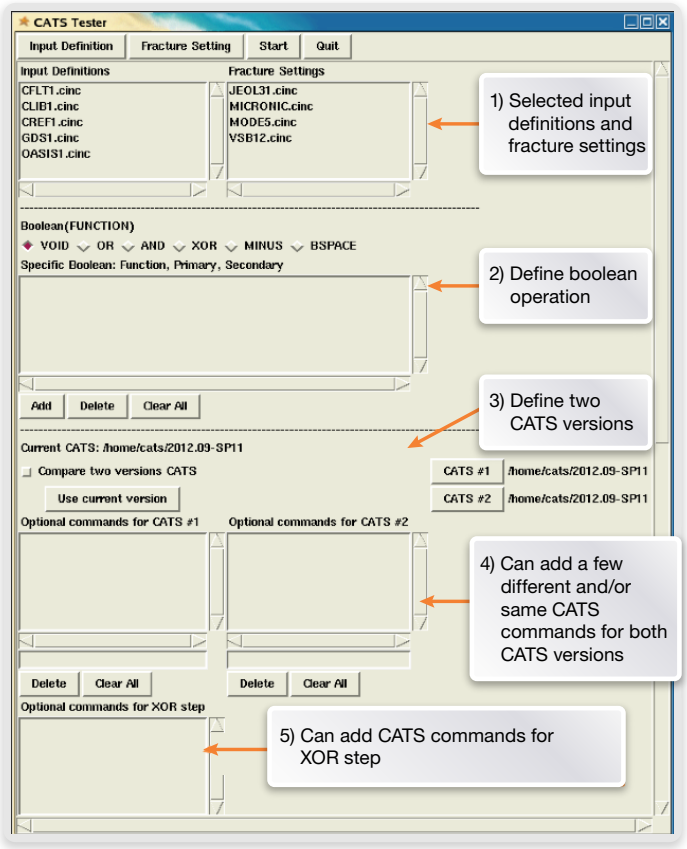


Figure 4

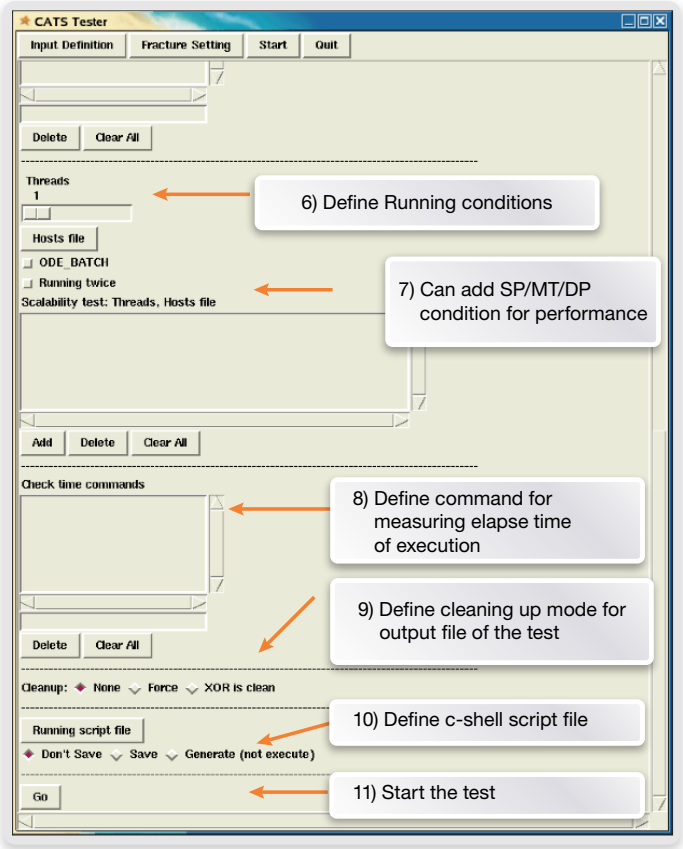
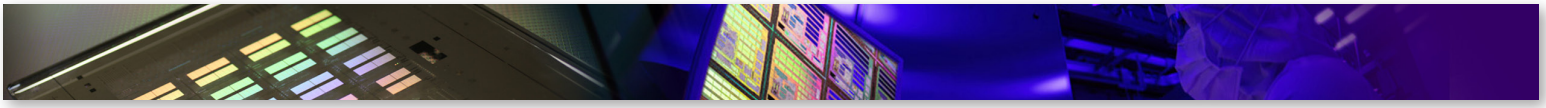


Figure 5



QR Code Generation in CATS using Third Party Code Generator

Hari Konnanur, CATS CAE

Background

Most semiconductor Masks made today have watermarks in the form of a barcode or an alphanumeric text, which are like signatures on the Mask. These watermarks can be in 1D or 2D.

Processes were much simpler when the information to be written on the mask was more 1D in nature, where the logo information (such as "Made in USA") was typographically pre-fractured into a library and assembled. Later barcodes were embedded but required very precise dimensions of bar widths and would contain 12 numerical digits.

As stepper manufacturers such as ASML wanted to describe more information, they needed the ability to put 2D barcodes on the Mask.

QR Code has become the de facto industry standard for generating 2D barcodes.

There are many industry standard QR Code generation tools in the marketplace and this article describes how a third party QR Code generation tool could be used in conjunction with CATS as a flow to achieve fractured native machine formats for those QR Codes.

This article does not imply or endorse the use age of any specific third-party QR Code generation tool. In this article, QRenCode is used for demonstration purposes. For more information on this tool, visit http://fukuchi.org/works/qrencode/

Benefits of generating 2D QR barcode

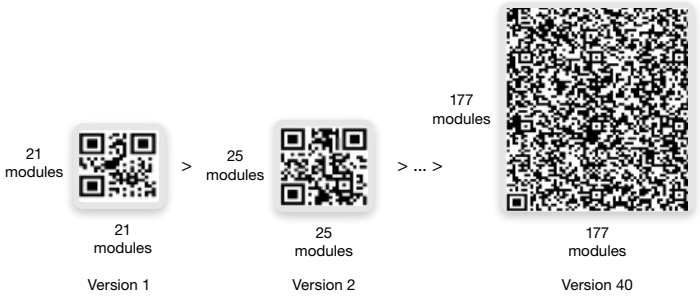
Some of the benefits of using QR Code to watermark semiconductor chips today include:

1. Capacity: 2D QR Code allows users to store more information than 1D with a maximum of 4,296 alphanumeric characters
2. Content: The benefit on content for 2D barcode mostly on the reader side is to largely decode alphanumeric text, URL's etc.
3. Error Correction: Every QR Code has a built-in error correction mechanism. Depending on the optics of the scanner, it can retrieve information from a 7% to 30% damaged QR Code.
4. Readability: QR Codes are readable independent of orientation, which is a significant benefit when on the Mask.
5. Standards: Due to the standardization of QR Code, no matter which tool you use to generate a 2D code the results are always the same. Today there are two models of standardization, with Model 2 being quite popular. These standards are part of ISO (ISO/IEC18004-2006).
6. Licensing QR Code is a patented technology. The patent belongs to a Japanese company named Denso Wave. The "license agreement" is similar to GNU in that it is free for use and given to the community for both personal and commercial use.

QR Code Versions and specifics

There are different version for QR code ranging from Version 1 to Version 40. Each version has a different module configuration. It is important to know the version requirements and other specifications before generating the 2D code.

The figure below illustrates, the level of 2D barcode complexity with various versions



For example, the specification from ASML for 2D bar code reads:

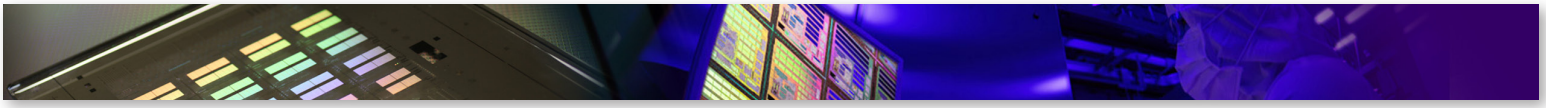
QR Encoding

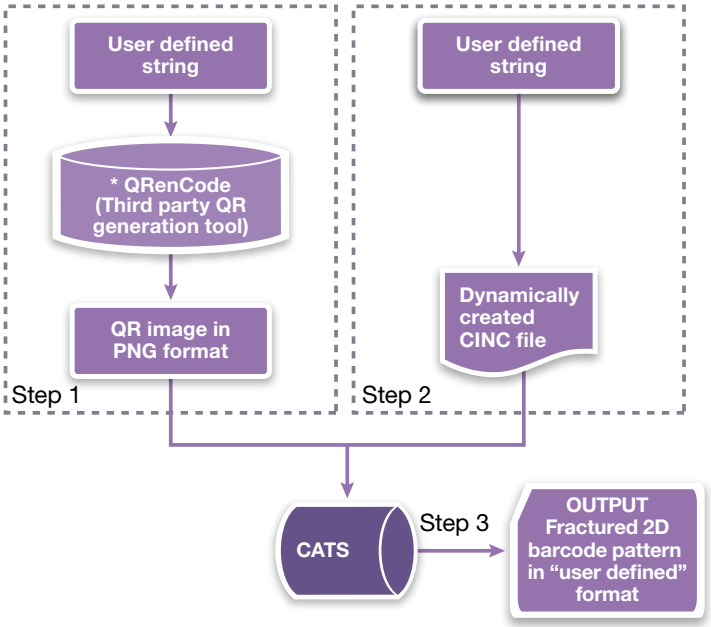
The NXE systems that have the QR format reader option installed adhere to the following specifications:

- Model: 2
- Version: 4
- Error correction level: H

QR Code Generation in CATS using Third Party Code Generator Flow

" Typical request from the customer could be : Generate a QR Code based on a user supplied string and fracture it to user specified machine format with a fixed dimension of 1.6 micron square in reverse tone" ▶





*A third-party application is demonstrated for generating QR Code.

To demonstrate the flow, a flow script called qrGenerator_for_CATS.pl* was created with the following algorithm:

1. Read the user specified QR Code string to be generated
2. Read the user defined final native machine format type: Could be one of Nuflare, JEOL or MEBES
3. Read the desired output filename.

As processing steps:

- **Step 1 (Create the PNG image)**
  QRencode binary is called within the program with the correct set of options to generate a 1.6 micron square QR Code in bitmap format
- **Step 2 (read the user defined output format)**
  Based on the user defined output format, generate a dynamic CINC file with a place holder to read the generated PNG image from Step 1
- **Step 3 (Execute the dynamic CINC file)**
  - The image is stored for further processing
  - CATS executes in the background with the dynamically generated CINC file as the input and generates the final QR Code pattern in the desired native machine format as the output.
  - Based on final output name the user specifies, the following rules are applied format specific:
    – The final pattern generated will be named to the MEBES or JEOL pattern name as specified by the user
    – The final pattern generated in case of VSB will be renamed to the directory name that contains VSB data
- **Step 4 (clean up : Run automatically, not described in the flow chart)**
  All temporary files are cleaned up

*-qrGenerator.pl is a script developed to show the integration. This script is not a part of the standard CATS distribution. But if customers are interested please feel free to contact CATS support.

Demonstration and usage of qrGenerator.pl script step by step

```
qrGenerator_for_CATS.pl <QR_STRING> <fracture_
format> VSB/MODE5/JEOL <Output_FileName>
```

For example, if the string is "JBX_007" is to be converted to 2D barcode in MODE5 format, you would use:

```
qrGenerator_for_CATS.pl JBX_007 MODE5 QR_
JBX_007_00.
```

In this example, the wrapper generates a text string with name "JBX_007", to be in MODE5 format, with the final pattern name as QR_JBX_007_00.

For JEOL and VSB, it should be substituted with CATS supported machine options such as VSB12, JEOL31, or 309000mv.

Please refer to the FORMAT command in the CATS Command Manual for the complete list of supported formats.

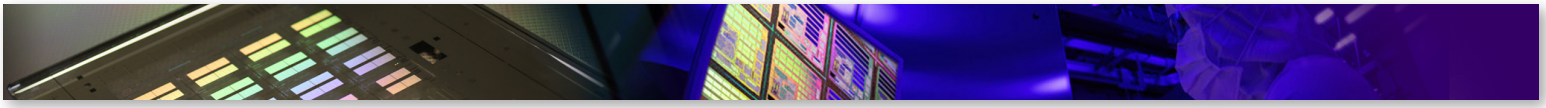
To show the outputs at each Steps

Here is the output after Step 1 [Create the PNG Image]

CATS is run as a part of Step 2 [Read the user defined output format]

Here is the sample of the dynamically created CINC file:

```
CATS G-2012.09-SP12 RHEL64 Linux 2.6.9-89.ELsmp
DEVELOPMENT 2014/03/16 23:59:14 378823 (not for
production use)
!Tue Mar 18 11:21:54 2014
Clear
! ALLOCATE Parameters
Allocate_rects 100000
Allocate_traps 100000
Allocate_space 3000000
Format MODE5
MEBES MACHINE 3
Compact REDO
Reverse YES
Resolution 0.1
```



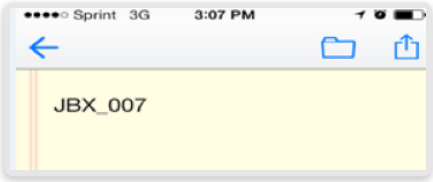
```
Direct_Output YES
Height [65536]
Width [32768]
Input ./image_for_fracture.png
Structure root
Layers -
Limits (INPUT 0,0) (INPUT 1650,1650)
Output ./QR_JBX_007_00
```

Now post-fractured, CATS is run as a part of Step 3 [Execute the dynamic CINC File]

As per the result, the pattern generated is in reverse tone, and measures 1.62 microns in MODE5 format.

Validation Step

There are many ways to validate the generated QR result. A convenient way is to use one of many apps available for smart phones. The following figure validates the initial text string.



Summary

This article describes a method to show the integration of a third-party QR generation tool to be used within CATS using a simple Perl script wrapper. This could be extended to any QR generation tool the customer might be using today.

CATS can seamlessly take bitmap image patterns generated from any third-party QR generation library and generate patterns to any supported native machine formats.

These patterns can be generated with any user specific .cinc file settings (e.g. in reverse tone).

Finally, the generated pattern can be placed in the Jobdeck for exposure. ■

Metrology: New Ability to Add Marks on Rotated or Staircase Data

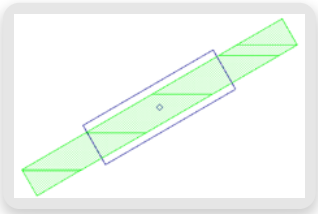
Herman Boerland, CATS R&D

Marktypes for rotated data

Almost all metrology tools (with the exception of KLA8100, KMS, and LMS2000) can measure rotated data. Within CATS, these features can be selected with the CDRot and BarRot marktypes. The WLCD metrology tool also has the ability to measure rotated boxes with the BoxRot marktype.

This article will describe some additional commands which can be used during the setup to measure these features.

```
JD ADD MARK dopick CDROT <coordinate>
```



This example shows a feature in which edges are actually parallel, and therefore easy to select. However, it is also possible to measure:

- Variable Shaped Beam (VSB) data, in which slanted edges are approximated with staircase or 45 degree edges
- Features containing OPC

To resolve these issues, the metrology module in CATS is extended with some new techniques, including:

- JD MARKCHECK SMOOTH_DEVIATION
- JD MARKCHECK JOGSIZE
- Filtering ▶

VSB Data

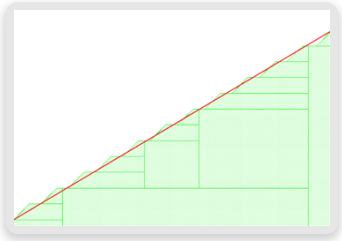
The algorithm to find rotated features is as follows:

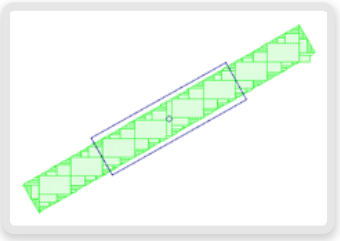
- Find the edge closest to the coordinate
- Search for a parallel edge on the opposite side of the coordinate

For VSB data, the edges are created using a staircase of 0- and 45-degree vertices, and therefore the correct angle cannot be found. When the edges are smoothed before the algorithm starts, the correct angles can be found and the mark will be created:

```
JD MARKCHECK SMOOTH_DEVIATION <value>
```

Here, the value is the size of the perpendicular distance between the corners and the final (smoothened) edge (in µm):



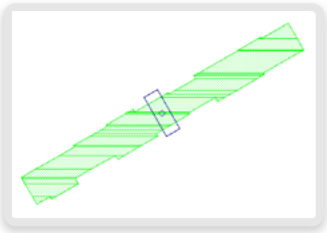


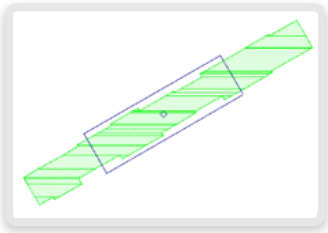
OPC Data

Even when the rotated features do not have straight edges, they still might be measurable, but most likely, the nonmeasure_width of the mark should get extended beyond the jogs.

To remove the jogs, another MARKCHECK parameter is available:

```
JD MARKCHECK JOGSIZE <value>
```





Here, the value is the size of the jogs that need to be removed (in µm). The difference with the SMOOTH_DEVIATION parameter is that the angle of the original two edges is kept the same.

Filters

Another enhancement for the CDRot, BarRot, and BoxRot marktypes are filters. In the past, filters could be used only when your marking session used JD MARKPLACE SEARCH. For the rotated marktypes, you can now add filters when using a 'plain' JD ADD MARK:

```
JD ADD MARK dopick CDRot
[CDRange:<minimum>,<maximum>]
[ANGLE: <minimum>,<maximum>]
[Tone:<OPAQUE | CLEAR>]
<coordinate>
```

The CDRange, Angle, and Tone filters are optional.

When using JD MARKPLACE SEARCH, both filtering methods are available.

Summary

This article describes marks that can be applied to rotated data. Rotated data can be slanted lines or lines approximated by a staircase (with or without 45 degrees edges). It can also handle rotated data that is OPC corrected by specifying a jogsize filter. ■

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

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

©2014 Synopsys, Inc. All rights reserved. Synopsys is a trademark of Synopsys, Inc. in the United States and other countries. A list of Synopsys trademarks is available at http://www.synopsys.com/copyright.html. All other names mentioned herein are trademarks or registered trademarks of their respective owners.
07/14.AP.CS4528/CC/50.