

## **Release Notes**

*December 2011*

### **GDSReticle 6.4.0**



39, avenue du Granier

B.P. 65 Inovallée

38242 Meylan - FRANCE

**tel:** +33 (0)4 76 41 10 96

**fax:** +33 (0)4 76 90 29 65

**web:** [www.dolphin-integration.com](http://www.dolphin-integration.com)

**e-mail:** [medal@dolphin-integration.com](mailto:medal@dolphin-integration.com)

COPYRIGHT © 1992 - 2011 Dolphin Integration. All rights reserved. No part of this document may be transmitted, reproduced, or transcribed in a retrieval system without prior written consent of Dolphin Integration.

The information in this manual is subject to change without notice. Dolphin Integration assumes no responsibility for any errors that might be contained in this document. Dolphin Integration reserves the right to revise this document without any obligation to notify any person of such revision or change.

The software described in this manual is supplied under a license agreement between you and Dolphin Integration. The license agreement authorizes the number of copies that may be made and the computer systems on which they may be used. Any unauthorized duplication or use in whole or part is forbidden.

**Mac OS** is a registered trademark of Apple Inc.

**Microsoft Windows** is a registered trademark of Microsoft Corporation.

**SMASH** is a registered trademark of Dolphin Integration.

**Verilog** is a registered trademark of Cadence Design Systems Inc.

# Contents

<b>Contents</b>	<b>2</b>
<b>Preamble</b>	<b>9</b>
Web Site . . . . .	9
<b>GDSReticle 3.4.2</b>	<b>10</b>
New Features . . . . .	10
SPACING CONSTRAINT . . . . .	10
<b>GDSReticle V3.4.3</b>	<b>10</b>
Bug Fixing . . . . .	10
Improvement . . . . .	10
<b>GDSReticle V3.4.4</b>	<b>11</b>
New features . . . . .	11
DUMMY (DDIsa01059) . . . . .	11
Improvements . . . . .	11
DDIsa01243: . . . . .	11
DDIsa01298: . . . . .	11
Bug Fixing . . . . .	11
DDIsa01297: . . . . .	11
DDIsa01346: . . . . .	11
<b>GDSReticle V3.4.5</b>	<b>12</b>
Improvements . . . . .	12
<b>GDSReticle V3.4.6</b>	<b>13</b>
New feature . . . . .	13
Support of SPM patterns (DDIsa01716) . . . . .	13
Improvement . . . . .	13
DDIsa01713: . . . . .	13
Bug Fixing . . . . .	13
DDIsa01710: . . . . .	13
DDIsa01265: . . . . .	13
DDIsa01533: . . . . .	13
<b>GDSReticle V3.4.7</b>	<b>14</b>
Bug Fixing . . . . .	14
DDIsa01740: . . . . .	14

<b>GDSReticle V3.4.8</b>	<b>14</b>
Bug Fixing . . . . .	14
DDIsa01753: . . . . .	14
<b>GDSReticle V3.4.9</b>	<b>14</b>
Improvement . . . . .	14
<b>GDSReticle V3.4.10</b>	<b>14</b>
Bug Fixing . . . . .	14
DDIsa01949: . . . . .	14
<b>GDSReticle V3.4.11</b>	<b>15</b>
Improvement . . . . .	15
DDIsa01848: . . . . .	15
DDIsa01905: . . . . .	15
Bug Fixing . . . . .	15
DDIsa01849: . . . . .	15
<b>GDSReticle V3.5.0</b>	<b>16</b>
Improvement . . . . .	16
DDIsa02188 and DDIsa02179: . . . . .	16
Bug Fixing . . . . .	16
DDIsa02245: . . . . .	16
<b>GDSReticle V3.5.1</b>	<b>17</b>
Bug Fixing . . . . .	17
DDIsa02362: . . . . .	17
<b>GDSReticle V4.0.0</b>	<b>18</b>
Improvement . . . . .	18
<b>GDSReticle V4.0.1</b>	<b>18</b>
Bug Fixing . . . . .	18
DDIsa02462: . . . . .	18
DDIsa02463: . . . . .	18
<b>GDSReticle V4.1.0</b>	<b>19</b>
Improvement . . . . .	19
DDIsa02569 . . . . .	19

<b>GDSReticle V4.2.0</b>	<b>19</b>
New feature . . . . .	19
DDIsa02576 . . . . .	19
Improvement . . . . .	19
DDIsa02258 . . . . .	19
<b>GDSReticle V4.2.1</b>	<b>20</b>
New features . . . . .	20
DUMMY_FILL (DDIsa02715): . . . . .	20
Circuit related patterns (DDIsa02577): . . . . .	20
Scribe crossing areas (DDIsa02578): . . . . .	20
Improvement . . . . .	21
DDIsa022716 . . . . .	21
<b>GDSReticle V4.2.2</b>	<b>22</b>
New features . . . . .	22
SEQUENCE (DDIsa02579): . . . . .	22
Vertical and horizontal dummies (DDIsa02580): . . . . .	23
Bug Fixing . . . . .	23
DDIsa02672: . . . . .	23
<b>GDSReticle V4.3.0</b>	<b>24</b>
New features . . . . .	24
PRECISION (DDIsa01261): . . . . .	24
<b>GDSReticle V4.3.1</b>	<b>25</b>
Bug Fixing . . . . .	25
DDIsa02802: . . . . .	25
<b>GDSReticle V4.3.2</b>	<b>25</b>
Bug Fixing . . . . .	25
DDIsa02862: . . . . .	25
<b>GDSReticle V4.4.0</b>	<b>26</b>
New features . . . . .	26
STEPPINGX and STEPPINGY (DDIsa02889): . . . . .	26
Layer filling (DDIsa02844): . . . . .	26
OVERLAP_ALL (DDIsa02627): . . . . .	27
Improvements . . . . .	28
License management (DDIsa01261): . . . . .	28

<b>GDSReticle V4.5.0</b>	<b>29</b>
New features	29
SPM not centered (DDIsa03158):	29
Avoid placement of dummies in scribe crossing area (DDIsa03157):	29
Improvements	30
Pattern placement optimizations (DDIsa03157):	30
OVERLAP_ALL improvement (DDIsa02627):	31
Bug Fixing	31
Problem in circuit related patterns (DDIsa03163):	31
Problem of placement of wide SPM with guards and TRYNOCROSSING option (DDIsa03162):	32
<b>GDSReticle V4.6.0</b>	<b>33</b>
New features	33
New #include instruction in the fdf language (DDIsa02727):	33
New “check bboxes” parameter in fmpplace, fgdsout (DDIsa02431):	33
Improvements	33
Difference between database unit in fdf file and pattern gds file (DDIsa03277):	33
Patterns having OVERLAP_ALL=TRUE are now defaulted to DUMMY_FILL=TRUE (DDIsa03278):	33
<b>GDSReticle V4.7.0</b>	<b>34</b>
New features	34
Merge the chip layout to the final frame layout (DDIsa01711):	34
A “pattern list” file to inhibit the placement of some patterns (DDIsa02728):	34
Bug Fixing	34
Overlap when the length of a pattern is smaller than the width of a scribe line (DDIsa03659):	34
ftclout program try to read the gds file of a pattern which already has a MASTERBBOX (DDIsa03660):	35
<b>GDSReticle V4.8.0</b>	<b>36</b>
New features	36
A new program fexteye (DDIsa03165)	36
Port to Linux (DDIsa03552)	36
Improvements	36
Upgraded TCL/TK (DDIsa04243)	36
<b>GDSReticle V5.0.0</b>	<b>37</b>
New features	37
Port to Linux 64 bits OS (DDIsa03549)	37
Parameterize the behavior of the external scribe lines at the crossing area for the filling layers (DDIsa02258)	37
Bug Fixing	37
Fatal error when merging with 53 files containing 1100 cells (DDIsa04513)	37

<b>GDSReticle V5.0.1</b>	<b>38</b>
Bug Fixing	38
A license problem with the feature fexteye (DDIsa04724)	38
<b>GDSReticle V5.0.2</b>	<b>38</b>
Bug Fixing	38
The executable fgdsout with option rename may generate wrong gds files (DDIsa05035)	38
<b>GDSReticle V5.1.0</b>	<b>39</b>
New features	39
A new keyword PROTECT_LAYER (DDIsa04732)	39
Improvements	39
Improved default placement algorithm (DDIsa02780)	39
Error message modification for the report of not placed patterns (DDIsa04889)	40
Bug Fixing	40
Bug in one chip optimization algorithm (DDIsa04712)	40
fmplace place patterns over objects already placed, making overlaps (DDIsa04897)	40
<b>GDSReticle V5.1.1</b>	<b>41</b>
Bug Fixing	41
Error messages while launching fexteye 5.1 (DDIsa05304)	41
<b>GDSReticle V5.1.2</b>	<b>41</b>
Bug Fixing	41
For the Virtual objects, the definition FILL=FALSE does not work correctly (DDIsa05360)	41
<b>GDSReticle V5.2.0</b>	<b>42</b>
New features	42
Removing some dice to make room for test patterns (DDIsa04900)	42
Improvements	43
Allow any position of origin in Circuit (DDIsa05088)	43
GDSReticle should allow to choose the tmp directory location (DDIsa05297)	43
Improvement of the zoom-out function (DDIsa05452)	43
Bug Fixing	44
A problem with the zoom function in xftclout window (DDIsa05356)	44
<b>GDSReticle V6.0.0</b>	<b>45</b>
New features	45
Bug Fixing	47
fcplace does not propose all the solutions (DDIsa05677)	47

<b>GDSReticle V6.1.0</b>	<b>48</b>
New features . . . . .	48
Homogeneous placement of some patterns (DDIsa05458) . . . . .	48
Fill-in empty spaces after dummy placement with a smaller dummy pattern (DDIsa05158) . . . . .	49
fexteye should be able to extract the rotation of the cells containing the eyepoints(DDIsa05970) . . . . .	49
Improvements . . . . .	50
Improve spm placement (DDIsa05461) . . . . .	50
Placement improvement (DDIsa5459) . . . . .	50
The LIBNAME in the gds file generated by fgdsout is too long (DDIsa05994) . . . . .	50
Add a button browse for the input template file(DDIsa06024) . . . . .	50
Report improvements for fmpplace and fmpwplace (DDIsa05449) . . . . .	50
Bug Fixing . . . . .	51
fgdsout is 4~5 times slower than V4.5(DDIsa05802) . . . . .	51
 <b>GDSReticle V6.2.0</b>	 <b>52</b>
New features . . . . .	52
Support of the OASIS format in GDSReticle (DDIsa05454) . . . . .	52
Allow to place patterns having an orientation in the scribes of the other orientation (DDIsa05480) . . . . .	52
SEQUENCE in TOP and LEFT external scribes (DDIsa05453) . . . . .	52
Display the merge progress during the merge process (DDIsa05456) . . . . .	52
Improvements . . . . .	53
One color by pattern type (with a legend) (DDIsa05450) . . . . .	53
Bug Fixing . . . . .	53
Graphical interface: the abort buttons must work (DDIsa05457) . . . . .	53
 <b>GDSReticle V6.2.1</b>	 <b>54</b>
Bug Fixing . . . . .	54
The key word GDSDIR is not taken into account by fgdsout (DDIsa06902) . . . . .	54
 <b>GDSReticle V6.2.2</b>	 <b>54</b>
Bug Fixing . . . . .	54
MASTERBBOX defined for the DUMMY pattern is ignored by fgdsout(DDIsa06948) . . . . .	54
 <b>GDSReticle V6.2.3</b>	 <b>54</b>
Bug Fixing . . . . .	54
The LIBNAME in the output gds file contains a path (DDIsa07017) . . . . .	54
 <b>GDSReticle V6.3.0</b>	 <b>55</b>
New features . . . . .	55
Avoid identical top cell names on patterns (DDIsa02433) . . . . .	55

ftclout viewer : Display the name of the pattern when mouseover (DDIsa05451) . . . . .	55
Improvements . . . . .	55
Handling of different DB units in gds files (DDIsa01135) . . . . .	55
<b>GDSReticle V6.3.1</b>	<b>55</b>
Bug Fixing . . . . .	55
The output of fgdsout is erroneous when the total size of the gds files is huge(DDIsa07418) . . . . .	55
<b>GDSReticle V6.4.0</b>	<b>56</b>
New features . . . . .	56
Index the homogeneously placed patterns(DDIsa07069) . . . . .	56
New keyword CENTERED (DDIsa07072) . . . . .	56
New keywords GUARDMIN and GUARDMAX (DDIsa05460) . . . . .	56
GUI: Possibility to highlight a selected type (DDIsa07141) . . . . .	56
GUI: Have a matrix view of 2x2 in xftclout (DDIsa07142) . . . . .	57
Improvements . . . . .	57
The keyword DISTRIBUTE changed from a special type to a property (DDIsa07070) . . . . .	57
Handling of homogeneously placed patterns with respect to the crossing area (DDIsa07073, DDIsa07074) . . . . .	57
Homogeneous placement should be done only once (DDIsa07317) . . . . .	58
Handling of optional patterns when corner exclusion is set(DDIsa07140) . . . . .	58
Provide the possibility to align the SPM patterns as sequences(DDIsa07075) . . . . .	58
Bug corrections . . . . .	58
Partially anchored patterns are not placed correctly (DDIsa07208) . . . . .	58

## Preamble

As always for new releases, we would like to thank those customers who take the time to report problems and/or to suggest improvements (please remember that the best way to do so is by sending an email to [medal@dolphin-integration.com](mailto:medal@dolphin-integration.com) or [support@dolphin-integration.com](mailto:support@dolphin-integration.com) with an accurate description of your problem or suggestion, together with the relevant files if any). As you will see in the new features, we do our best to take remarks into account. And even if your suggestion does not appear this time, don't think it was lost or disregarded. Simply, it means that its implementation could not fit into the development plan for this particular release, but be assured that we will try to take it into account in a future release.

## Web Site

Our web site <http://www.dolphin-integration.com> is a source of information on our EDA solutions. Aside from evaluation kits for our products, a number of application notes, courses or upgrades are available for download.

## GDSReticle 3.4.2

### New Features

#### SPACING CONSTRAINT

A “SPACES” section has been added to the fdf file format. It is used to define X/Y minimum spacing between the mobile objects. The spacing constraint applies to a set of objects defined in the “TYPES” section. The section “SPACES” must therefore appear at least after the TYPES section. Types having a spacing constraint must have an “ORIENT” property defined. The syntax is:

```
....  
TYPES { CIRCUIT... TYPE1 ORIENT = HOR; TYPE2 ORIENT = VER; }  
SPACES { TYPE1 MIN = 100; TYPE2 MIN = 200; }  
LAYOUTS  
{ ... }  
LAYERS { }
```

In this example, objects of type TYPE1 will be placed horizontally with a spacing of 100, objects of type TYPE2 will be placed vertically with a spacing of 200.

Important note: The SPACE section is available only with the “default” option of the placement tool (fmplace binary). As the “fast” and “slow” options will become obsolete in the future, new features will not be implemented for these placement algorithms.

## GDSReticle V3.4.3

### Bug Fixing

The “fmplace” program would crash when placing a huge number of “padding” patterns. This has been corrected.

### Improvement

Memory management has been improved in “default” placement algorithm.

## GDSReticle V3.4.4

### New features

#### DUMMY (DDIsa01059)

The keyword DUMMY is added as a predefined type. It is similar to the user-defined types except that each fdf file can have only one pattern of type DUMMY. A pattern of type DUMMY is used to fill in the unoccupied space in the scribe lines. After the placement of mandatory and optional patterns, the dummy pattern is placed as many times as necessary to fill in the space left in the scribe lines. This feature is useful to replace the PADDING type when the pattern is very small. In this case the PADDING feature could become very slow - and greedy in memory. PADDING and DUMMY cannot be used together. ORIENT keyword has no effect on DUMMY pattern.

Example:

```
DUMMY NAME_OF_PATTERN GDSFILE = "filename"
```

### Improvements

#### DDIsa01243:

A warning is now printed when a bit "Absolute angle/magnify" set to 1 is found in a GDSII file in input of GDSReticle.

#### DDIsa01298:

A warning is now emitted when two different GDSII pattern files have the same top cell name. This could lead to wrong GDS databases.

### Bug Fixing

#### DDIsa01297:

Multi top cells GDSII databases are generated when a pattern in input has a top cell not in first position.

This has been corrected.

#### DDIsa01346:

Cell names in input GSDII files having more than 32 characters are not correctly managed. In some case, this could lead to a segmentation fault. This has been corrected.

## GDSReticle V3.4.5

### Improvements

FlexIm protection server has been updated from version v7.0 to v7.2f. This has been done in order to keep compatibility between GDSReticle and other Dolphin products (SoCGDS, GDSCompiler, SMASH. . .)

## GDSReticle V3.4.6

### New feature

#### Support of SPM patterns (DDIsa01716)

SPM are patterns with specific placement constraints. A SPM pattern is to be placed as close as possible to the center of the frame.

**SPM\_X** keyword is dedicated to horizontally placed SPM marks and must have the property **ORIENT = HOR** defined.

**SPM\_Y** keyword is dedicated to vertically placed SPM marks and must have the property **ORIENT = VER** defined.

A **GUARD** property can be defined if a space is needed between the SPM and the circuit or the neighboring dummies. In this case, two virtual patterns from the specified width are added along each side of the SPM object.

### Improvement

#### DDIsa01713:

When a GDSII cell name is larger than 32 characters, GDSReticle stops with no explicit error message. Now the name of the cell is issued before the program exits.

### Bug Fixing

#### DDIsa01710:

Aref of dummies are not correctly placed in the final GDSII file.

This has been corrected.

#### DDIsa01265:

Default placement algorithm: Sometimes patterns are not placed even though there is enough remaining space.

This has been corrected.

#### DDIsa01533:

When several optional patterns have the same name, only one is placed.

This has been corrected.

## GDSReticle V3.4.7

### Bug Fixing

#### DDIsa01740:

On SPM patterns, dummies objects should not fill «guards» protections.

This has been corrected.

## GDSReticle V3.4.8

### Bug Fixing

#### DDIsa01753:

Bug in fmplace binary, default placement algorithm (all versions of GDSReticle (and old frog) software): Sometimes, in rare cases, free (not oriented) patterns could be placed with a wrong orientation parameter. This could lead to overlapped patterns. In that case, overlaps are correctly detected by fgdsout, ftclout, and fdrc binaries.

This fmplace issue has been corrected.

## GDSReticle V3.4.9

### Improvement

The number of GDSII object that can be load by the fgdsout program has been increased from 3,000,000 to 20,000,000.

## GDSReticle V3.4.10

### Bug Fixing

#### DDIsa01949:

Bug in fmplace binary, unplaced SPM marks were not correctly added in the final summary of not placed objects.

This could lead to a faulty “All mandatory objects placed” message.

This fmplace issue has been corrected.

## GDSReticle V3.4.11

### Improvement

#### DDIsa01848:

A warning message is now issued when a pattern is defined with an orientation and is typed with an other orientation. In the following example, the type TEST is HOR oriented and the pattern named thePattern is of type TEST and VER oriented:

```
TYPE {  
TEST ORIENT=HOR; }  
  
LAYOUTS {  
TEST thePattern ORIENT=VER; }
```

#### DDIsa01905:

A new **-force** command line option has been implemented in the fmplace program. This option is dedicated to the patterns having a spacing constraint. The **-force** option tell the fmplace program to place spaced patterns even if they cannot be placed according to the required spacing constraint. This is useful if the spacing constraint is not a strong constraint. In that case the patterns are placed normally, without any spacing.

### Bug Fixing

#### DDIsa01849:

Bug in fmplace binary, when an undefined type was used in the "SPACES" section, the binary abort on segmentation fault. Now the program will issue an explicit error message and exit.

## GDSReticle V3.5.0

### Improvement

#### DDIsa02188 and DDIsa02179:

GDSReticle can now manage GDSII databases having a database unit different from 10-9. Two new keywords have been added in the FDF language: **DATABASEUNIT** and **USERUNIT**. If these keywords are not defined, they are supposed to have the following default values (to be backward compatible): **DATABASEUNIT** = 1e-9 **USERUNIT** = 1e-3

It is mandatory to check that all input GDSII files (test patterns...) contain the same database unit and matches with the **DATABASEUNIT** parameter defined in the corresponding fdf file. The final GDSII produced by GDSReticle will contain the correct database unit. Notice that there is no direct link between **DATABASEUNIT** (and **USERUNIT**) and **SCALE** and **UNIT** parameters. **SCALE** converts GDSII units in FDF internal one, and **UNIT** increases the FDF coordinates accuracy.

Example of use:

```
TECHNO YOUR_TECHNO_NAME
GDSCELLNAME="$CELL"
SCALE=1000.0
USERUNIT=1e-3
DATABASEUNIT=0.92e-9
GDSDIR = "$env(PATTERNS)/GDSFILES_TEST_3_092u"
{ ...
```

### Bug Fixing

#### DDIsa02245:

Bug in fgdsout binary, when a top cell name has exactly 32 characters, the binary abort on segmentation fault.

This has been corrected.

## GDSReticle V3.5.1

### Bug Fixing

#### **DDIsa02362:**

A lack of accuracy in the storage of the new **DATABASEUNIT** variable could lead (in rare cases) to bad coordinates in the final GDSII.

This has been corrected.

## GDSReticle V4.0.0

### Improvement

In the frame of the new requirements linked to sub-micron technological processes, and in order to answer recent improvement suggestions from our customers, a new gdsout module has been linked to GDSReticle. This gdsout module is intended to read, process and write all the GDSII files requested and produced by GDSReticle.

This shared library is originally part of Dolphin's SoCGDS product.

The old version of the GDSII library module was becoming obsolete for the following reasons:

- Memory management not designed for large GDSII files (speed and memory footprint)
- Cell names limited to 32 characters
- Layers limited to the range of 0-255
- Datatypes limited to the range of 0-63

The SocGDS version of the GDSII unit library encompasses the following possibilities:

- Large GDSII file management (file larger than 2Gb for 64 bits version)
- Reading gzipped GDSII files
- Unlimited size for cell names
- Layers and datatypes in the range of 0-32767

For the user, there is no modification in the software interface. Some information messages have been modified in the "fgsout" program printout.

## GDSReticle V4.0.1

### Bug Fixing

#### DDIsa02462:

On the Solaris 64 bit version, the optional patterns (having a "PRIORITY" property defined) were sometimes not placed.

This defect has been corrected

#### DDIsa02463:

An error occurs in the placement of the dummies arrays when the origin of the dummy pattern does not correspond to the lower left corner of the layout. In this case, the array of patterns is slightly shifted.

This has been corrected.

## GDSReticle V4.1.0

### Improvement

#### DDIsa02569

In the frame of the new requirements linked to sub-micron technological processes, and in order to answer recent improvement suggestions from our customers, GDSReticle now provides support for reading circuits and patterns from OpenAccess libraries. The extensions to the FDF and command line parameters are 100% upwards compatible, so existing configurations are unaffected by this improvement

## GDSReticle V4.2.0

### New feature

#### DDIsa02576

This new feature defines a protection against placement of objects around patterns. The width and the height of the protection are parameters. Two new keywords are available: **EXTENT\_DX** and **EXTENT\_DY**. **EXTENT\_DX** defines the width of the protection along the shorter side of the pattern, and **EXTENT\_DY** is the width along the longer one. Dummies fill the empty area reserved by the protection if a **DUMMY** pattern is defined.

### Improvement

#### DDIsa02258

The bottom and top scribe lines are now filled to the end of the left and right side of the frame by the filling layers, instead of stopping at the inner border of the left and right scribe lines.

## GDSReticle V4.2.1

### New features

#### DUMMY\_FILL (DDIsa02715):

A new property **DUMMY\_FILL** has been added for virtual patterns. This property is a Boolean, if it is true (**DUMMY\_FILL=TRUE**), the corresponding pattern will be filled by the dummy pattern if it exists. **DUMMY\_FILL** is useful to create empty areas where only dummies can be placed. This feature cannot be used with patterns having a layout file attached. **DUMMY\_FILL** will often be used in association with the following **DDIsa02577** to create protections around the die circuit area.

#### Circuit related patterns (DDIsa02577):

This new feature adds the possibility to place patterns (either virtual or true layout patterns) relatively to the circuit area. As many circuits can be present in a frame, this kind of pattern will be duplicated as many time as necessary to be placed relatively to each circuit of the frame. If a fixed object is in the way and prevents the placement, a warning is emitted and the current instance of the pattern is not placed. A “circuit related” pattern is defined by a “pattern vertex” = **CIRCUIT** . ”vertex” assignment.

Example of use:

```
TYPES{
...
PROTECTCIRCUIT VIRTUAL=TRUE;
}
LAYOUTS{
LET WIDTH=10;
...
PROTECTCIRCUIT CIRCUIT_L MASTERBBOX=[0,0,WIDTH,FRAME.CY] RB=CIRCUIT.LB;
...
}
```

In this example the protection box named **CIRCUIT\_L** will be placed along the left side of each circuit of the frame. (The right bottom corner of **CIRCUIT\_L** is attached to the left bottom corner of the circuit by the **RB=CIRCUIT.LB** assignment).

#### Scribe crossing areas (DDIsa02578):

New properties have been added in order to avoid placement in the crossing areas of the scribe lines. These properties should be declared in the header of the fdf file.

**NOCROSSING=TRUE** can be used to avoid placement in the crossing areas. No pattern will be placed in these areas (not even the mandatory patterns).

**TRYNOCROSSING=TRUE** can be used instead of **NOCROSSING**, in this case, if some mandatory patterns cannot be placed, the crossing areas will be released one after another until all the mandatory patterns have been placed.

Crossing areas can be extended horizontally and vertically beyond the intersections of the scribes: **XCROSSGUARD**=(integer) defines an horizontal extension of the crossing area **YCROSSGUARD**=(integer) defines the vertical extension.

Example of use:

```
TECHNO YOUR_TECHNO_NAME
GDSCELLNAME="TheCellName"
SCALE=1000.0
USERUNIT=1e-3
DATABASEUNIT=0.92e-9
TRYNOCROSSING=TRUE
XCROSSGUARD=100
YCROSSGUARD=200
GDSDIR = "/HOME/GDSDIR/..."
{
...
```

## Improvement

### DDIsa022716

The ftclout program has been improved by adding some colors in the visualization. Now the dummy pattern is gray, the SPM light blue, the virtual pattern maroon, the “virtual filled by dummies” orange. This makes the pre-visualization much easier to read.

## GDSReticle V4.2.2

### New features

#### SEQUENCE (DDIIsa02579):

Two new keywords **SEQUENCE** and **ORIGIN** are provided to define a list of patterns, which should be placed in specific scribe lines and in a specific order. The keyword **ORIGIN** is used to define the starting scribe line for the placement of the sequence. Only the internal scribe lines are counted when using the keyword **ORIGIN**.

The placement of sequences occurs after the placement of mandatory patterns and before the placement of optional patterns. The keyword **PRIORITY** is ignored for sequences.

Syntax:

Pattern\_name **SEQUENCE** = "sequence\_name"

**FILLX** = R2L|L2R

**FILLY** = T2B|B2T

**ORIENT** = HOR|VER

**ORIGIN** = scribe number

**MASTERBBOX** = [bbox];

Where:

**ORIENT** is the orientation of the scribe lines in which the sequence will be placed.

**ORIGIN** is the scribe number, used (together with **ORIENT**, **FILLX** and **FILLY**) to define the scribe line from which the placement should start. If the **ORIENT** is HOR and the **FILLY** is B2T then **ORIGIN**=2 means the second internal horizontal scribe counting from the bottom. **FILLX**, **FILLY** define the filling direction of the sequences. For **ORIENT**=HOR, **FILLX** indicates the patterns in the sequence will be placed from left to right or from right to left; **FILLY** indicates the scribe number is counting from top to bottom or vice versa.

Example:

```
...
LAYOUTS {
...
pattern_name1 SEQUENCE="sequence1"
ORIENT=HOR FILLX="R2L"
FILLY="T2B"
ORIGIN=2
MASTERBBOX=[bbox1];
pattern_name2
SEQUENCE="sequence1"
ORIENT=HOR
FILLX="R2L"
FILLY="T2B"
```

```

ORIGIN=2
MASTERBBOX=[bbox2];
...
}
...

```

In the above example, "sequence1" contains 2 patterns, it should be placed in the horizontal scribe lines starting from the second internal scribe counting from the top, in each scribe the patterns should be placed from right to left.

Note:

There are some constraints to the declaration of the sequences. Please refer to the User Manual for the details and further examples.

### Vertical and horizontal dummies (DDIsa02580):

Two new keywords **DUMMY\_H** and **DUMMY\_V** are now available for the definition of different dummy patterns to fill the vertical and the horizontal scribe lines. They are both predefined types and are used exactly the same way as the keyword **DUMMY**. **DUMMY\_H** defines a dummy pattern to fill the horizontal scribe lines, whereas **DUMMY\_V** defines a dummy pattern to fill the vertical scribe lines.

Example:

```

DUMMY_H Name_Of_HOR_Pattern GDSFILE = "hor_pattern_file_name"
DUMMY_V Name_Of_VER_Pattern GDSFILE = "ver_pattern_file_name"

```

Note:

- **DUMMY\_V** and **DUMMY\_H** should be used together. If only **DUMMY\_H** (or **DUMMY\_V**) is defined in the fdf file, the vertical (horizontal) scribe lines will not be filled with dummies.
- If both **DUMMY\_H/DUMMY\_V** and **DUMMY** are defined, the keyword **DUMMY** is ignored.

## Bug Fixing

### DDIsa02672:

If intermediate file .sol is modified manually and passed again to fmplice, the fmplice will attempt to place the groups already fixed by the previous iterations and the result is unpredictable.

This has been corrected.

## GDSReticle V4.3.0

### New features

#### PRECISION (DDIIsa01261):

From now on, both real numbers and integers are accepted as coordinates. A new keyword **PRECISION** is now available. To use real numbers as coordinates, **PRECISION** has to be specified in the header “**TECHNO**” section of the fdf file. A **PRECISION=0** corresponds to the actual situation (only integers are used). A **PRECISION=1** allows real numbers with one digit after the decimal point, and **PRECISION=2** allows two digits after the decimal point, etc. . .

Example:

```
TECHNO TEST_1
GDSCELLNAME="TEST_1_frame"
SCALE=1000.0
PRECISION = 2
GDSDIR = "... "
TYPES {
CIRCUIT DX=5000 DY=4000 SPACEX=160 SPACEY=160;
}
LAYOUTS {
RETICLE R1 DX=21000 DY=22800 DIAM=29700;
RETICLE R2 DX=15800 DY=26280 DIAM=29700;
/* TITLE */
MASKID TITLE GDSFILE="TEST_1_TITLE.gds" TC=FRAME.TC-[0.52,10];
...

```

In this example the **PRECISION** keyword is set to 2 and a decimal coordinate can also be used in the MASKID definition (0.52)

Note:

- For integers, that are not coordinates, such as priorities and scribe numbers, a letter L must be added as suffix (PRIORITY=1L, ORIGIN=2L, etc).
  - In an expression, all integers that are not coordinates should be suffixed by L, for example the following expression: (FRAME.DY / FRAME.NY) / 2 should be written (FRAME.DY / FRAME.NY) / 2L.
- Very important:** this is true even if **PRECISION** is not defined in the fdf file (by default, PRECISION=0), and it will cause GDSReticle to issue an error or warning (depending on the expression) on the fdf files which work fine with previous versions.
- The keyword **UNIT** is ignored if both **UNIT** and **PRECISION** are defined in the same fdf file.

## GDSReticle V4.3.1

### Bug Fixing

#### DDIsa02802:

The fcplace binary takes four coordinates in parameter to specify the width of the external scribe lines: -ellx, -elly, -eurx, -eury. In 4.3 version these parameters were rounded to integer values instead of taking correctly into account the digits after the decimal point.

This has been corrected.

## GDSReticle V4.3.2

### Bug Fixing

#### DDIsa02862:

The fgdsout binary crash on segmentation fault when linking more than 32 AREFs of dummies. This could be the case when a lot of empty space is to be filled by dummies in the frame. In that case the fgdsout binary stop before producing any .gds file.

This has been corrected.

## GDSReticle V4.4.0

### New features

#### STEPPINGX and STEPPINGY (DDIsa02889):

Two new keywords have been provided in order to calculate automatically the scribe line width according to a given stepping dimension. **STEPPINGX** is the stepping dimension for the vertical scribe line (corresponding to **SPACEX**); **STEPPINGY** is the stepping dimension for the horizontal scribe line (corresponding to **SPACEY**). The **STEPPING** keywords must be added in the **TYPE** section, on the **CIRCUIT** definition line:

Example:

```
TYPES {
CIRCUIT DX=5000 DY=4000 SPACEX=160 SPACEY=160 STEPPINGX=25.4;
...
}
```

In this example, **SPACEX**=160 means the minimum value for the vertical scribe lines is 160. GDSReticle will calculate (at the circuit placement step -fcplace binary-) a new **SPACEX** so that the value of **DX + SPACEX** is a multiple of **STEPPINGX**. After fcplace processing, the exact value for **SPACEX** will be 181.6.

```
TYPES {
CIRCUIT DX=5000 DY=4000 SPACEX=160 SPACEY=181.6 STEPPINGX=25.4 ORIENT=HOR;
...
}
...
LAYOUTS{
LET WIDTH=10;
RETICLE R1 DX=21000 DY=22800 DIAM=29700;
ARRAY CIRCUIT FRAME NX=4L NY=5L LB=[-10408.6,-10400] RETICLE="R1" ELLX=181.6 ELLY=80 EURX=90.8
EURY=80 SPACEX=181.6 SPACEY=160 CX=5000 CY=4000;
...
}
```

#### Layer filling (DDIsa02844):

A new keyword **LAYER\_DATA** has been defined. It allows the user to fill a virtual pattern by a set of layer and datatype pairs. This new feature is convenient to place protection objects filled by one or several layers.

Example:

```
THEBOX VIRTUAL=TRUE MASTERBBOX=[0,0,1200,80] LB=FRAME.LB LAYER_DATA=[54L,0L] LAYER_DATA=[62L,34L]
ORIENT=VER;
```

In this example the virtual pattern named “THEBOX” will appear on the final GDSII file as a rectangle (under the top cell), filled by the layers 54, datatype 0 and layer 62, datatype 34.

#### **OVERLAP\_ALL (DDIsa02627):**

If the **OVERLAP\_ALL** keyword is defined on the declaration line of a pattern, the overlap with other objects (circuits or patterns) are NOT checked, no error or warning message is issued. This feature is useful to put protection rectangles over the whole reticle.

Warning:

As overlaps are not checked, this feature must be used carefully.

Example:

```
THEBOX VIRTUAL=TRUE MASTERBBOX=[0,0,1200,80] LB=FRAME.LB LAYER_DATA=[54L,0L] LAYER_DATA=[62L,34L]
OVERLAP_ALL=TRUE ORIENT=VER;
```

#### **COMMAND keyword (DDIsa02929):**

The new keyword **COMMAND** allows GDSReticle to launch an external layout generator to create the GDSII file of the pattern “on-the-fly”.

Example:

```
THEBOX GDSFILE="theGDSFile.gds" MASTERBBOX=[0,0,1200,80]
COMMAND="theCommand.csh" ORIENT=VER;
```

In this example, if the GDSII file theGDSFile.gds does not exist or cannot be found, GDSReticle will launch the external command script theCommand.csh with the following parameters:

```
theCommand.csh --left=0 --bottom=0 --right=1200 --top=80 --orient=VER
```

This command can be a shell script or a binary. It is up to the user to write it correctly in order to generate the GDSII file theGDSFile.gds at the right place. The user can use the parameters to calculate the bounding box and the geometry accordingly to the **MASTERBBOX** and the **ORIENT** given in the .fdf template file. theCommand.csh must be placed in the directory from where GDSReticle is launched.

Note:

The **COMMAND** instruction cannot be used to generate virtual and dummy patterns.

## Improvements

### License management (DDIsa01261):

A better license management has been implemented in GDSReticle v4.4:

- If all the licenses owned by the user are already in use, GDSReticle will wait until the requested token is released (instead of exiting).
- A warning message will be issued if the license will expire in less than 15 days.

## GDSReticle V4.5.0

### New features

#### SPM not centered (DDIsa03158):

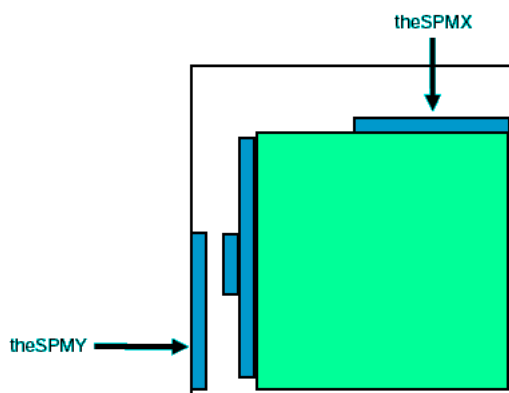
A new keyword is provided in order to avoid the center constraint on SPM patterns. A SPM having the **NOT\_CENTERED=TRUE** feature will not be centered. **FILLX** and **FILLY** can be used to control the placement of the SPM, (guards included). The new property **NOT\_CENTERED** can be used either on the type of the SPM or directly on the SPM itself.

Example:

```
theSPMXType theSPMX NOT_CENTERED=TRUE FILLX="R2L" FILLY="B2T";
```

```
theSPMYType theSPMY NOT_CENTERED=TRUE FILLX="L2R" FILLY="B2T";
```

The resulting layout of not centered SPM's:

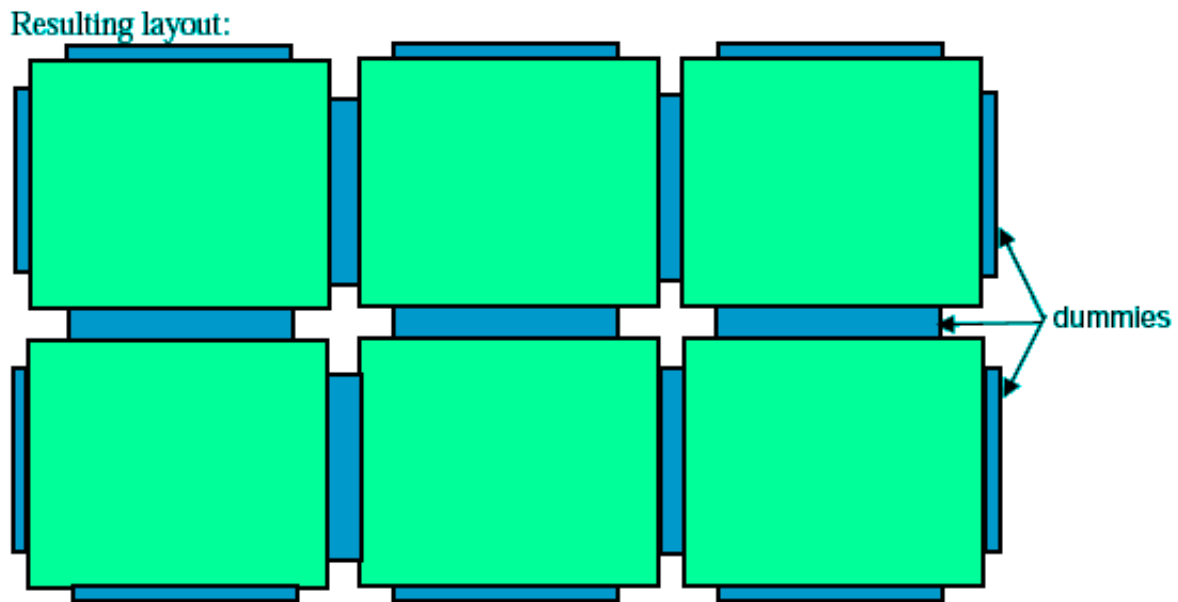


#### Avoid placement of dummies in scribe crossing area (DDIsa03157):

A new keyword **DUMMIESNOCROSSING** is provided in order to avoid the placement of dummies in the scribe crossing areas. **DUMMIESNOCROSSING** is to be used in the header part of the fdf file in association with **NOCROSSING** or **TRYNOCROSSING**. The X and Y **CROSSGUARD** properties are also taken into account.

Example of use:

```
TECHNO TEST_1 GDSCELLNAME="TEST_1_frame" SCALE=1.00e+03 TRYNOCROSSING=TRUE XCROSS-  
GUARD=100 YCROSSGUARD=100 DUMMIESNOCROSSING=TRUE
```

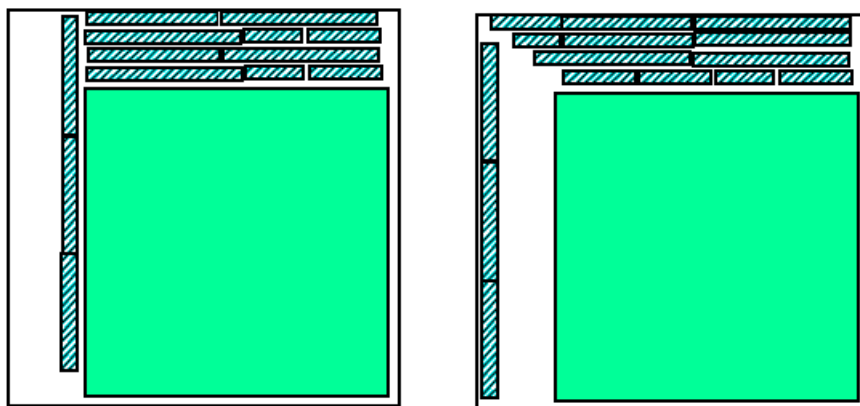


## Improvements

### Pattern placement optimizations (DDIsa03157):

Major improvements in patterns placement have been made for frames having only one chip. In this particular case (which is becoming the most common case on 90, 65 and 45nm frames), the placement constraints are very specific. In GDSReticle 4.5, the one-chip-frame case is automatically detected and the following optimizations are activated:

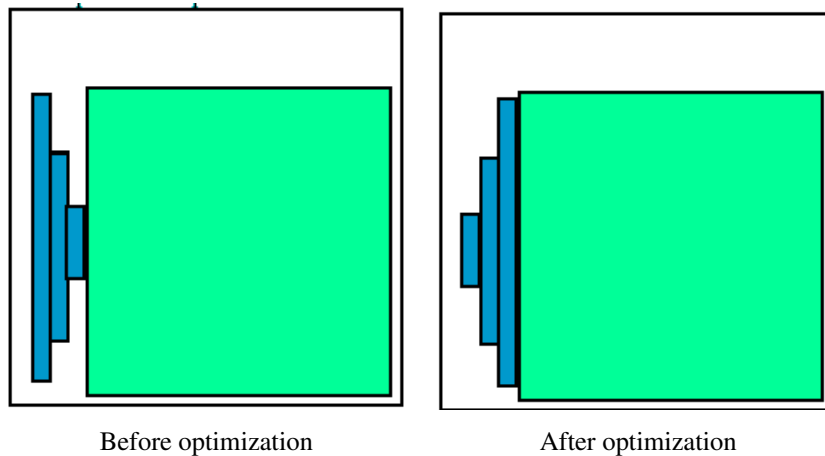
- Detection of the critical scribe in term of space and placement in this scribe first.
- Ol- Placement from the external side of the scribe to the internal side (in order to save empty space in the crossing scribe area)
- Placement from the smallest scribe to the largest one.
- Sort the patterns by decreasing width before placement (so that the largest are placed first), SPM included. Examples of placement:



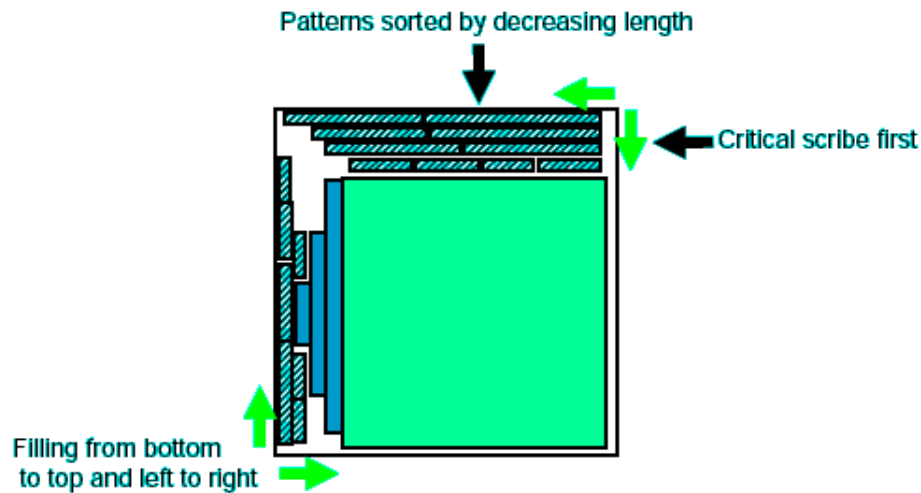
Old Placement

New Placement

## SPM placement optimization:



Summary of optimization:

**OVERLAP\_ALL improvement (DDIsa02627):**

The **OVERLAP\_ALL** keyword was only compatible with the fgdsout program (.solve files) in GDSReticle 4.4. Now the **OVERLAP\_ALL** feature can be used through the whole flow of GDSReticle (template .fdf files, .array files, .solve files).

**Bug Fixing****Problem in circuit related patterns (DDIsa03163):**

The fmplice program crash on segmentation fault when using an expression in a “circuit related” declaration: For example, the expression  $LB = \text{CIRCUIT.RT} + [10L, 10L]$  would crash fmplice.

This has been corrected.

**Problem of placement of wide SPM with guards and TRYNOCROSSING option (DDIsa03162):**

When the length of a SPM is larger than the chip and the **TRYNOCROSSING** option is used, and SPM guards are requested, the fmplace program creates wrong overlapping objects.

This has been corrected.

## GDSReticle V4.6.0

### New features

#### New #include instruction in the fdf language (DDIsa02727):

A #include instruction can now be used in the fdf language. This feature is very convenient to manage libraries of patterns.

Example:

```
#include "list_of_patterns.inc"
```

In this example, the content of the file named list\_of\_patterns.inc will replace the #include line after the parsing of the fdf file.

- #include instructions can also be put in included files.
- Notice that in template files, it is possible to use environment variables

Example:

```
#include "$env(PATTERNS)/list_of_pattern.inc"
```

In this case, the fdfcreate program will expand the variable \$PATTERNS

#### New “check bboxes” parameter in fmplice, fgdsout (DDIsa02431):

A new parameter “-**checkbboxes**” has been added to the binaries fmplice and fgdsout. When using this parameter, the program will read all the gds files of the patterns defined in the fdf file, even if the **MASTERBBOX** is defined, and the real masterbbox calculated in the GDSII will be compared to the one defined in the fdf file. If the masterbbox defined in the fdf file is smaller than the one defined in the GDSII a non-blocking error message is issued. It is not blocking because sometimes smaller bounding box are used to define non-rectangular patterns (like corners and crosses). The fdrc program now always checks the bounding boxes.

### Improvements

#### Difference between database unit in fdf file and pattern gds file (DDIsa03277):

If the GDSII file of a pattern has a different database unit from the database unit set in the fdf file, a fatal error is emitted and the program halts.

In earlier versions only a warning was emitted.

#### Patterns having OVERLAP\_ALL=TRUE are now defaulted to DUMMY\_FILL=TRUE (DDIsa03278):

When a pattern has the **OVERLAP\_ALL** property, it means the pattern overlaps a large area of the frame. In this case, it is more convenient to allow the filling of empty spaces by dummies in the area covered by this pattern. The default value for the **DUMMY\_FILL** property of this pattern is now set to TRUE instead of FALSE in V4.5.

## GDSReticle V4.7.0

### New features

#### Merge the chip layout to the final frame layout (DDIsa01711):

GDSReticle V4.7 introduces an important new feature: the merge of the circuit layout to the frame. In previous versions the fgdsout program only covered the chip area with rectangles of layers. The final merge is done by the fracturing program. In V4.7 it is possible to merge the layout of the chip directly by using the fgdsout program. The user has only to add into the fdf file, on the CIRCUIT type declaration line, a GDSDIR and a GDSFILE properties pointing on the layout file of the chip.

Example:

```
TYPE {
/* Circuit definition */
CIRCUIT DX=$DX DY=$DY SPACEX=$SPACEX SPACEY=$SPACEY ORIENT=$ORIENT GDSDIR="/circuit/path/"
      GDSFILE="circuit_file.gds";
...
}
```

The chip will be merged as a GDSII AREF of structures.

Notice that the orientation of the layout must be consistent with the DX and DY parameter.

#### A “pattern list” file to inhibit the placement of some patterns (DDIsa02728):

A new parameter has been added to the fmplace program:

**-patternfile** <pattern file name>

The “pattern file” is a text file containing a list of pattern names (corresponding to a subset of the patterns found in the .fdf file). The patterns listed in the “pattern file” will not be placed during the fmplace step.

A “pattern file” is a text file with a very simple format: one pattern name per line.

Example:

```
PATTERN_NAME_1
PATTERN_NAME_2
PATTERN_NAME_3
PATTERN_NAME_4
```

### Bug Fixing

#### Overlap when the length of a pattern is smaller than the width of a scribe line (DDIsa03659):

Sometimes (in rare cases) an overlap can happen if a small pattern is placed in a wide scribe line. This overlap was correctly detected as an error by GDSReticle.

This bug has been corrected.

**ftclout program try to read the gds file of a pattern which already has a MASTERBBOX (DDIsa03660):**

Sometimes (in rare cases) in the V4.6 version only, the ftclout program attempted to check the bounding box of some patterns even if the MASTERBBOX property is defined.

This bug has been corrected.

## GDSReticle V4.8.0

### New features

#### A new program **fexteye** (DDIsa03165)

A new executable **fexteye** has been introduced. It can be used to extract eyepoints from GDSII databases.

Eyepoints are special text objects in the GDS2 frame file. The tool **fexteye** will read the GDSII file and extract the names and the coordinates of such objects and put the result in an output file.

#### **usage:**

This program uses a different set of libraries from the rest of programs, before the first use, you have to set the environment:

```
source $GDSReticle_DIR/fexteye.csh and then
```

```
fexteye <input-gds-file-name> <output-text-file-name>
```

Example:

```
fexteye example.gds example_output.txt
```

**fexteye** will read the file `example.gds` and write the eyepoints found in it in `output.txt`.

Options: the option **-prefix** allows the user to specify a prefix, in this case **fexteye** will extract only eyepoints having the given prefix.

Example:

```
fexteye -prefix "KEY" example.gds example_output.txt
```

#### Port to Linux (DDIsa03552)

From now on, GDSReticle is available on linux platform.

### Improvements

#### Upgraded TCL/TK (DDIsa04243)

In GDSReticle 4.8, TCL/TK libraries have been upgraded to V8.4. The radio buttons in the interface which were not correctly initialized before are now initialized with the default values.

## GDSReticle V5.0.0

### New features

#### Port to Linux 64 bits OS (DDIsa03549)

From now on, GDSReticle is available on linux 64 bit platforms.

#### Parameterize the behavior of the external scribe lines at the crossing area for the filling layers (DDIsa02258)

In the previous releases of GDSReticle, the vertical external scribe lines are longer than the horizontal ones. This caused a problem when filling only some of the external scribe lines (top and right, for example), since the horizontal ones were short and would leave a hole at the intersection with the vertical external scribe (the left) which was not to be filled.

In GDSReticle 5.0, extremity descriptors are added in the **LAYERS** section to allow the users to control the behavior of a external scribe line at each of its end points for the filling layers.

The syntax of the extremity descriptor:

```
extremity_name = "SHORT" | "LONG"
```

The possible extremity names for horizontal (**TOP** and **BOTTOM**) scribes are **LEFT** and **RIGHT**, and for vertical (**LEFT** and **RIGHT**) scribes, **TOP** and **BOTTOM**.

When an endpoint of a scribe line is defined to be "LONG", the filling of the layer will extend to the outer limit of the frame, and when it is defined as "SHORT", the filling will stop before the intersection with the perpendicular external scribe. By default, the extremities are LONG.

Example:

...

```
LAYERS {
11 NUM = 100 DATATYPE = 31 SCRIBE = "RIGHT" TOP = "LONG" BOTTOM = "SHORT"
11 NUM = 100 DATATYPE = 31 SCRIBE = "TOP" LEFT = "SHORT" RIGHT = "SHORT"
}
```

The first statement says that the **RIGHT** scribe will be filled all the way to the top but stops before the intersection with the **BOTTOM** scribe. The second statement says the filling of the TOP scribe stops at both intersections with **LEFT** and **RIGHT** scribes.

### Bug Fixing

#### Fatal error when merging with 53 files containing 1100 cells (DDIsa04513)

In previous versions, GDSReticle was not able to perform a fgdsout with option rename when the total number of cells to be merged exceeds a certain limit. In GDSReticle V5.0, the tools for the merge have been upgraded and as a consequence, the limit has been lifted.

## GDSReticle V5.0.1

### Bug Fixing

#### **A license problem with the feature fexteye (DDIsa04724)**

There was a problem with the license when using the feature fexteye.

This has been fixed in V5.0.1.

## GDSReticle V5.0.2

### Bug Fixing

#### **The executable fgdsout with option rename may generate wrong gds files (DDIsa05035)**

There was a bug in the dealing of the name conflicts during the gdsout procedure which may cause the output gds file to be erroneous.

This defect appears only in release V5.0.1 and is fixed in GDSReticle V5.0.2.

## GDSReticle V5.1.0

### New features

#### A new keyword **PROTECT\_LAYER** (DDIsa04732)

This keyword allows the users to declare patterns of type **PROTECT\_LAYER**, which is a predefined type.

A pattern of type **PROTECT\_LAYER** must have a **LAYER\_DATA** property. Its position will be specified with respect to the **FRAME** or the **CIRCUIT**. The type **PROTECT\_LAYER** could be used to fill certain scribe line areas left by the patterns with the desired layer.

The syntax for the declaration:

**PROTECT\_LAYER** pattern\_name positions

**LAYER\_DATA**=[layer\_num, datatype\_num]

**MASTRBBOX**=[x1, y1, x2, y2]

optional: **ORIENT**=HOR/VER, **OVERLAP**=TRUE/FALSE (allowing the pattern to overlap with the circuit array, by default it is true)

Example:

**LAYOUTS** {

...

**PROTECT\_LAYER** THE\_FILL BC=FRAME.BC **LAYER\_DATA**=[1L, 51L]

**MASTERBBOX**=[0, 300, 1560, 0]; /\* pattern position related to frame \*/

**PROTECT\_LAYER** THE\_FILL\_CIRCUIT C=CIRCUIT.C **LAYER\_DATA**=[1L, 51L]

**OVERLAP**=FALSE **MASTERBBOX**=[0,0, FRAME.CX+160, FRAME.CY+160]; /\*circuit related \*/

...

}

### Improvements

#### Improved default placement algorithm (DDIsa02780)

In some cases the default placement algorithm was not able to place mandatory patterns even though there is visibly enough room to place them. In release 5.1 the algorithm for the default placement has been improved so that more patterns can be placed.

#### New interface for **xftclout** (DDIsa04735)

The interface has been improved with a menu bar containing "file", "close", "quit" etc. as well as a file selection box. The command **xftclout** can take an argument now so that a file can be displayed directly.

**Error message modification for the report of not placed patterns (DDISa04889)**

A circuit related pattern is duplicated to be placed relatively to each circuit of the frame. In case of space shortage some of these circuits are sometimes not placed. In the previous release the behavior is to issue a warning during the placement if a pattern can not be placed, and in the final summary the unplaced circuit related objects are not reported. Starting from release V5.1, in addition to the warning, the number of not placed patterns will be shown in the final summary in the form of a fraction.

For example:

```
### Warning: 1/9 of circuit related object 'Pattern_Name_1' NOT placed...
```

```
### Warning: 4/9 of circuit related object 'Pattern_Name_2' NOT placed...
```

**Bug Fixing****Bug in one chip optimization algorithm (DDISa04712)**

When there is only one chip in the frame and there are only external scribes, in some configurations the patterns are placed with wrong orientations.

This has been corrected in release V5.1.

**fmplace place patterns over objects already placed, making overlaps (DDISa04897)**

When using fmplace on a solution file (output of fmplace), it tempts to place the spaced objects (objects of type SPACE) even if they are already placed by the previous iterations.

This defect has been corrected in release V5.1.

## GDSReticle V5.1.1

### Bug Fixing

#### Error messages while launching fexteye 5.1 (DDISa05304)

When the executable fexteye is launched, an error message “cannot create library” appears and the program stops.

This has been corrected in release V5.1.1.

## GDSReticle V5.1.2

### Bug Fixing

#### For the Virtual objects, the definition FILL=FALSE does not work correctly (DDISa05360)

A virtual object can be filled or not by the filling layers. If FILL=FALSE, or the keyword FILL is absent, it should not be filled. In the previous releases, the objects were not filled when the keyword FILL was absent (correct behavior), however they were filled if FILL=FALSE (incorrect behavior).

In release 5.1.2, this has been corrected. Furthermore, a warning is issued when FILL=FALSE is present in the fdf file.

## GDSReticle V5.2.0

### New features

#### Removing some dice to make room for test patterns (DDIsa04900)

When the dice are very small, it may be interesting to remove some dice of the array in order to insert the test structures. In this case the scribe lanes remain empty and the width of the scribes may be reduced to the minimum dimension authorized by sawing tools.

To achieve this, a new keyword **REMOVE** is introduced. This keyword can be used in the circuit definition statement in the **TYPES** section of the fdf file. The dice to be removed are indicated by their index (row, col) in the array. The row indexes start from 1, counting from bottom to top. The column indexes start from 1, counting from left to right. For example (1,1) indicates the die at the lower left corner.

The syntax:

```
CIRCUIT ... REMOVE=[row1, col1, row2, col2]
```

where row1, col1, row2, col2 are integers. row1, col1 indicate the starting die and row2, col2 the ending one. If only one die is to be removed then row1 = row2 and col1=col2.

**Important:** The output file of fofout (Order form): If a die is removed from the circuit array and the test structures have been placed in its place, the coordinates of the die are **not** written into the order form file, they will be simply skipped.

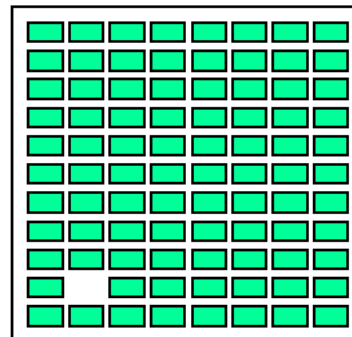
#### Remarks:

1. If the keyword REMOVE is present in the circuit definition, GDSReticle will place the structures in the removed die areas only and leave the scribes empty.
2. If an index goes beyond the limit of the array, a warning message is issued and nothing is removed.

Examples:

```
Example 1
TYPES {
/* circuit definition */
CIRCUIT
...
REMOVE=[2L, 2L, 2L, 2L]
...
}
```

The die removed is situated at second row second column.



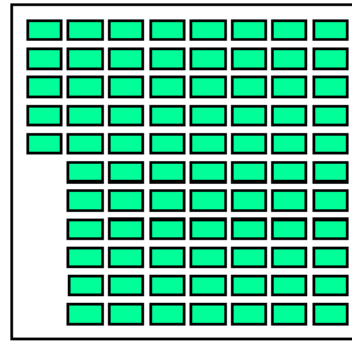
## Example 2

```

TYPES {
/* circuit definition */
CIRCUIT
...
REMOVE=[1L, 1L, 6L, 1L]
...
}

```

The first six dice of the first column are removed.



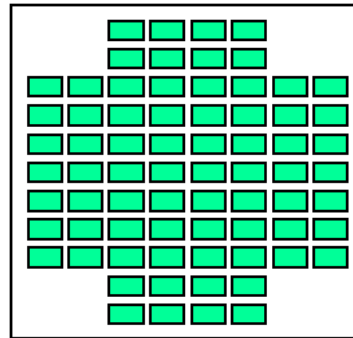
The keyword REMOVE may only be used in the line of the circuit description, however it can appear more than once, as in the following example:

## Example 3

```

TYPES {
/* circuit definition */
CIRCUIT
...
REMOVE=[1L, 1L, 2L, 2L]
REMOVE=[10L, 1L, 11L, 2L]
REMOVE=[10L, 7L, 11L, 8L]
REMOVE=[1L, 7L, 2L, 8L]
...
}

```



## Improvements

### Allow any position of origin in Circuit (DDIsa05088)

In previous releases, when merging the circuit to the frame, the origin of the layout of the circuit must be at lower left corner, otherwise the circuits were not correctly positioned in the final gds file.

This limitation has been withdrawn in release 5.2.

### GDSReticle should allow to choose the tmp directory location (DDIsa05297)

In previous releases of GDSReticle, the directory */tmp* is used to store the temporary files during the merge of gds files (in *fgdsout* program). From now on, the user may specify a directory with the environment variable **TMPDIR** for the storage of temporary gds files. If **TMPDIR** is not specified, the directory */tmp* is used as before.

### Improvement of the zoom-out function (DDIsa05452)

In previous releases of GDSReticle, only zoom-in can be performed by clicking and dragging a rectangle in the *xftclout* window. From now on, if one clicks with the left-mouse and drags to the left, zoom-out is performed, and if one drags to the right zoom-in is performed as before. The key binding *shift-z* for the zoom-out remains available.

## Bug Fixing

### A problem with the zoom function in xftcloud window (DDIsa05356)

In previous releases of GDSReticle, an accidental single click empties the drawing area.

This has been corrected in release V5.2

## GDSReticle V6.0.0

### New features

Support multi-chip placement (MPW) (DDIsa00973) Multi-chip reticles are used to minimize mask fabrication cost for circuit prototypes. We consider only the case where the circuit chips have already been assembled and come with their position well defined in a certain coordinate system. GDSReticle will then perform the placement of the mask patterns between the different circuits and on the rectangular borders.

#### The description of the circuit chips

The format for the description of circuit chips will be very much like the format used to describe other structures. The predefined type **CHIP** is used to distinguish the circuits from other test patterns. A new keyword **TOPCELL** will be introduced to indicate the cell to be used from the specified gds file.

The description of the circuit chips can be put in a separate file from the fdf file and then be included with a #include clause in the **LAYOUT** section of the FDF, or it can be directly written in the FDF in the **LAYOUT** section.

#### The format

```
CHIP chip_name GDSFILE="file_name" TOPCELL="top_cell_name" anchor=[x, y]
MASTERBBOX=[x1, y1, X2, y2] ROT=rotation
```

*where*

**chip\_name:** is the name for identifying the chip.

**file\_name:** is the path to the gds file of the chip.

**top\_cell\_name:** is the name of the cell in the gds file to be used, by default the first top cell found in the gds file will be used.

**anchor:** is one of the existing keywords: LB, RB, TR, TL, CB, ...etc.

**ROT** is optional. The value of rotation may be 0, 90, 180, and 270.

**GDSFILE** is optional, if it is present for a chip, this chip will be merged into the final gds file. If it is absent, **MASTERBBOX** must be defined.

**TOPCELL** is optional, it can only be used together with the **GDSFILE** keyword.

#### Remarks:

1. **MASTERBBOX** may be absent if **GDSFILE** keyword is used to indicate the gds file.
2. The coordinate system used for the position of the chips will be the coordinate system for GDSReticle for further placements.

#### Changes in the FDF for MPW

Two new keywords **MPWARRAY** and **MPWFRAME** have been introduced for the multi-chip FDF's only.

**MPWFRAME** This keyword is used to define the external scribe dimensions for the frame. It will be mandatory for all the MPW projects and should be present in the **TYPES** section of FDF. It replaces the line starting with **CIRCUIT** in mono-chip FDF's

Example:

[TYPES] MPWFRAME ELLX=100 ELLY=100 EURX=100 EURY=100;

**MPWARRAY** This keyword starts the line of the mpw array description. This line is generated by the program *fmpwarray* and is placed in the **LAYOUTS** section of the generated .array file. It starts always with **MPWARRAY CHIP FRAME**, and will contain the name of the reticle associated with this array, the dimensions of the external scribes, the LB position of the FRAME, and the dimensions (DX and DY) of the FRAME. This is the equivalent line of the array description in the mono-chip projects (ARRAY CIRCUIT FRAME. . .)

Example:

[LAYOUTS]MPWARRAY CHIP FRAME ELLX=160 ELLY=260 EURX=100 EURY=100  
LB=[7639.8, 12645.2] DX=14959.6 DY=24770.4 RETICLE="R2";

### The executables

Two new executables *fmpwarray* and *fmpwplace* will be introduced for the handling of MPW. The flow *fcplace->fmpplace* for mono-chip projects will then become *fmpwarray->fmpwplace* for multi-chip projects.

The other existing executables such as *fgdsout*, *ftclout*, *fofout*, etc. have undergone modifications to handle the mono/multi-chip issues, their usages remain the same as before.

### *fmpwarray*

This new executable is equivalent to the existing *fcplace*. It takes an FDF and generated a fram with the circuit chips.

It will compare the size of the fame against the reticles defined in the input FDF and propose a solution for each reticle that may fit. The output file of this step is the .array files as for the mono-chip projects.

In the input file, a new keyword **MPWFRAME** must be present, otherwise the FDF is not considered a multi-chip project. It is also mandatory to have the circuit chips descriptions in the multi-chip FDF, either directly or included. When the input file is not a multi-chip project, *fmpwarray* will issue an error and stop. **Usage:**

**fmpwarray [-t] [-u] [-v] [-V]**

**[-all:give all solutions]**

**<input file> <output file>**

### *fmpwplace*

This new executable is equivalent to the existing *fmplace* for mono-chip projects. It will take a .array file generated by *fmpwarray* and produce a solution file by placing the pattern structures in the free places left by the chips.

When the input file is not a multichip project it will issue an error and stop.

**Usage:**

**fmpwplace [-t] [-u] [-v] [-V]**

**[-checkboxes:** check masterboxes: check for each pattern consistency between fdf hard codedmasterboxes and real GDSII file ones]

**[-patternfilter <pattern filter file name>:** the text file pointed by <path name> must contain a list of pattern names corresponding to the pattern names found in the input file. If a pattern is present in both file, it will NOT be placed in the result file. This is a way to parameterize the patterns to be placed]

**<input file> <output file>**

## Bug Fixing

### **fcplace does not propose all the solutions (DDIsa05677)**

In previous releases of GDSReticle, when (for instance)  $NX=2$ ,  $NY=3$  and  $NX=3$ ,  $NY=2$  are both possible, the program fcplace proposes only one of them. Even though the two solutions have the same amount of circuits (6 in this example), they do not provide the same area in horizontal and vertical scribe lines. That's why the two solutions are now proposed in V6.0.0.

## GDSReticle V6.1.0

### New features

#### Homogeneous placement of some patterns (DDIsa05458)

This feature allows the user to place a special pattern repeatedly in such a way that they are evenly distributed in the scribes.

New keywords:

**DISTRIBUTE**: A predefined type.

**REPEATMIN**, **REPEATMAX**: properties attached to the **DISTRIBUTE** type, to define the minimum and maximum number of times a **DISTRIBUTE** type pattern should be placed.

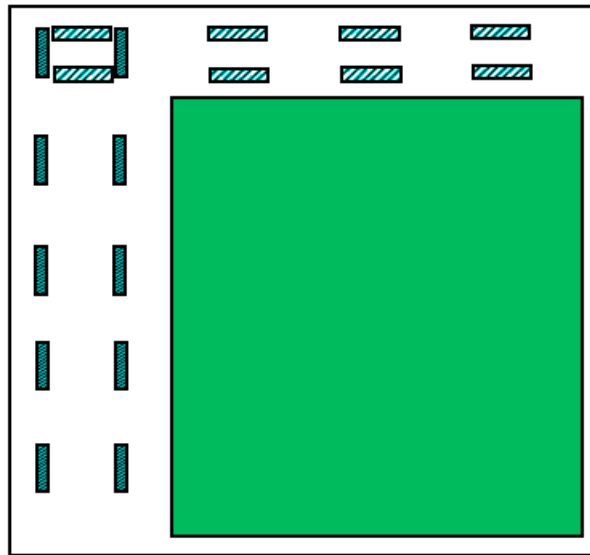
The definition of **ORIENT** is required for **DISTRIBUTE** type patterns, **ORIENT** = HOR / VER indicates the pattern should be placed in horizontal or vertical scribe lines.

If **ORIENT** is not specified, *fmplace* will issue an error and stop.

There may be at most two patterns of type **DISTRIBUTE**, one for the horizontal scribes and one for the vertical scribes.

Examples:

```
TYPES{
...
DISTRIBUTE GDSDIR = "path_to_gds_files" REPEATMIN = 8L REPEATMAX = 12L
}
LAYOUTS {
...
DISTRIBUTE DISTRIBUTE_HOR GDSFILE = "FILE_HOR.gds" REPEATMIN = 5 ORIENT = HOR MASTERB-
BOX =[ ... ] DISTRIBUTE DISTRIBUTE_VER GDSFILE = "FILE_VER.gds" ORIENT = VER ...
}
```



Distributed Placement

### Fill-in empty spaces after dummy placement with a smaller dummy pattern (DDIsa05158)

In previous releases, only one dummy pattern was allowed. The gaps left, although too small for dummies to be placed, were not always negligible. Two modifications have been made in this release. First, merge the spaces as much as possible before filling them with dummies. Second, add the possibility for users to specify in the fdf file up to (but no more than) three dummy patterns. The dummies should be presented in decreasing order of sizes (will be placed in this order).

Examples:

```
...
LAYOUTS{
...
DUMMY DUMMY_L GDSFILE = "large.gds";
DUMMY DUMMY_M GDSFILE = "medium.gds" ;
DUMMY DUMMY_S GDSFILE = "small.gds" ;
... }
```

#### Warning:

Any non-regression tests based on the text comparison of the report will fail due to the change in the merge procedure (less dummy structures with larger dimensions).

### fexteye should be able to extract the rotation of the cells containing the eyepoints(DDIsa05970)

A new option `-rotation` is added to `fexteye`. When `fexteye` is run with this option it reports the rotation of the cell in addition to the usual information.. The rotation is shown at the end of the line with `R=rot`. If this option is not specified, there will be no report of the rotations, as before (the default behavior)

Example of some lines with rotation report:

```
Text=B cell=HF5_9_A0_7 Hier=MV912A_frame/HF5_9_A0/ X=15650 Y=6300 XF=9623900 YF=-2052750 L=62 D=0 R=180
```

Text=IO cell=HF5\_D\_A0\_6 Hier=MV912A\_frame/HF5\_D\_A0/ X=45000 Y=47750 XF=9630000 YF=2258000 L=41 D=0 R=270

Text=B cell=HF5\_E\_A0\_10 Hier=MV912A\_frame/HF5\_E\_A0/ X=38400 Y=-2800 XF=9669800 YF=4617450 L=41 D=0 R=90

Text=EW cell=HF5\_E\_A0\_3 Hier=MV912A\_frame/HF5\_E\_A0/ X=0 Y=10400 XF=9586600 YF=4349950 L=25 D=0 R=0

## Improvements

### Improve spm placement (DDIsa05461)

Normally the spm's should be placed as close to the center of the frame as possible, this is actually the case. However, if the top and left external scribes are very wide (as in the case of one-chip reticle), if a spm is centered to the frame then it would cut into the wide scribe and some long patterns may not be placed as a result. In this new release, GDSReticle will try to avoid cutting into the wide external scribes, if this is possible, by shifting the long spm patterns. The shorter spm's will be placed centered as before, as close to the center as possible.

### Placement improvement (DDIsa5459)

In previous releases fmpplace (and fmpwplace) places vertical mandatory objects first, horizontal ones after (the same order for spm placement). Two new options -vfirst, -hfirst are added in this release to fmpplace and fmpwplace. These options will indicate that spm's and test patterns will be placed in the preferred order of orientation. If neither vfirst nor hfirst is specified, by default, horizontal patterns are placed before the vertical ones (old behavior). It is not allowed to have both vfirst and hfirst options(GDSReticle will issue an error and stop). For fmpplace, in the case of one big chip reticle, these options are ignored. GDSReticle will find a optimal order of placement.

### The LIBNAME in the gds file generated by fgdsout is too long (DDIsa05994)

For releases between GDSReticle 5.0 and GDSReticle 6.0, the LIBNAME contained the full path of the output file which was unnecessarily long. In GDSReticle 6.1, the top cell name is used instead of the full path.

### Add a button browse for the input template file(DDIsa06024)

In previous releases, the field input file in GUI had to be typed manually. In this new release, a button "browse" is added which will open a file selection dialog and allow the user to choose the input file.

### Report improvements for fmpplace and fmpwplace (DDIsa05449)

Add report of the orientations of not placed objects: in addition to the name, the orientation is reported for each pattern which is not placed. Repeat the overlap warnings at the end: In addition to the usual overlap report, these warnings are repeated at the end of the report.

#### Warning:

Any non-regression tests based on the text comparison of the report will fail due to this change.

## Bug Fixing

### **fgdsout is 4-5 times slower than V4.5(DDIsa05802)**

It is noticed that starting from GDSReticle 5.0, fgdsout became much slower. This was due to the upgrade of the underlying software used by GDSReticle to handle the gds format. Modifications have been done in this release and the execution time of fgdsout has been significantly reduced.

## GDSReticle V6.2.0

### New features

#### Support of the OASIS format in GDSReticle (DDIsa05454)

Starting from release 6.2, GDSReticle supports **OASIS** format for both input and output gds files. A new option **-oasisin** is added for *fcplace*, *fmplace* and *fgdsout* to indicate that the patterns used are in oasis format. If this option is not specified, it is supposed that the format of the input patterns is gdsII. A new option **-oasisout** is added to *fgdsout*, if it specified the output of *fgdsout* will be in oasis format. If the option **-oasisout** is not specified, the output format will be gdsII. In the graphic interface, buttons are added to choose the oasis format for input or output.

#### Note:

When **-oasisin** is specified, all the patterns are supposed to be in oasis format.

#### Allow to place patterns having an orientation in the scribes of the other orientation (DDIsa05480)

Some small patterns having an orientation (horizontal for instance) can fit in scribe lines of the other orientation (vertical in this case) if the width of the scribe is larger than the pattern length. However this kind of placement was not possible since the horizontal (respectively vertical) patterns were always placed in horizontal (respectively vertical) scribes. Starting from GDSReticle 6.2.0, this restriction has been relaxed so that a small pattern will be allowed to be placed in the scribes that have room for it, regardless of the orientation.

#### SEQUENCE in TOP and LEFT external scribes (DDIsa05453)

Up to now the patterns of a sequence can only be placed in the internal scribes. Starting from release 6.2.0, it is possible to place sequences in the external scribes. This is achieved by allowing ORIGIN to be 0.

Examples:

```
...
TYPES{
...
SEQUENCE_VER SEQUENCE="SEQUENCE_1" FILLX="L2R" FILLY="B2T" ORIGIN=0L ORIENT=VER; SEQUENCE_HOR
    SEQUENCE="SEQUENCE_2" FILLX="L2R" FILLY="T2B" ORIGIN=0L ORIENT=HOR;
...
}
```

The combination of FILLX="L2R" and ORIGIN=0 for SEQUENCE\_VER indicate that the placement of sequences of this type will start from the LEFT external scribe. Similarly, the combination of FILLY="T2B" and ORIGIN=0 for SEQUENCE\_HOR indicate that the placement of sequences of this type will start from the TOP external scribe.

#### Display the merge progress during the merge process (DDIsa05456)

A progress bar has been added at the bottom of the dialogue window of fgdsout.

## Improvements

### One color by pattern type (with a legend) (DDIsa05450)

Starting from this release, in the tcl interface, the patterns with different types are displayed in different colors with a legend to the right of the main frame.

## Bug Fixing

### Graphical interface: the abort buttons must work (DDIsa05457)

In previous releases of GDSReticle, the button “abort” of the graphic interface for fmplace and fgdsout kills only the dialogue window and leaves the process running. This has been corrected in GDSReticle 6.2.0.

## GDSReticle V6.2.1

### Bug Fixing

#### The key word **GSDIR** is not taken into account by fgdsout (DDIsa06902)

When **GSDIR** is defined for the type **DUMMY** in the **TYPES** section, it is not taken into account by fgdsout and it searches for the pattern file in a wrong directory.

This has been corrected in GDSReticle 6.2.1.

## GDSReticle V6.2.2

### Bug Fixing

#### **MASTERBBOX** defined for the **DUMMY** pattern is ignored by fgdsout(DDIsa06948)

When **MASTERBBOX** is defined in the **LAYOUTS** section of fdf for a dummy pattern, it should be used for the calculation of the dimensions of aref in the output gds file. However, starting from GDSReticle 6.2.0, fgdsout ignores the **MASTERBBOX** defined in the fdf and always fetches the bbox from the gds file of the dummy pattern. This has been corrected in GDSReticle 6.2.2.

## GDSReticle V6.2.3

### Bug Fixing

#### The **LIBNAME** in the output gds file contains a path (DDIsa07017)

In the gdsII file generated by fgdsout, the **LIBNAME** contains a path to a temporary file, this causes the crash of the fracturing tools . Starting from GDSReticle 6.2.3, the **LIBNAME** in the output gds file will be *Top\_Cell\_Name.DB*.

## GDSReticle V6.3.0

### New features

#### Avoid identical top cell names on patterns (DDIsa02433)

In previous releases, when two patterns have the same top cell name, *fgdsout* would issue an error message and stop. Starting from GDSReticle 6.3.0, when the option *-rename* is used, *fgdsout* will rename the top cells having the same name and continue with the merge, a message is issued to inform the user of the renaming. If the option *-rename* is not used, however, *fgdsout* will issue an error message and stop as before.

#### ftclout viewer : Display the name of the pattern when mouseover (DDIsa05451)

Starting from this release, in *xftclout* GUI, the pattern is highlighted when the mouse passes over it, the name of the pattern is displayed in a text field on the bottom-right corner of the frame. A pattern name can be typed in this text field, if it exists and it is in the current view, it will be highlighted (a return key is expected to mark the end of the name).

### Improvements

#### Handling of different DB units in gds files (DDIsa01135)

In previous releases of GDSReticle, a verification of database unit for each gdsII file is performed and if different database units are found an error message is issued. Starting from GDSReticle 6.3.0, different database units are allowed as long as they are compatible. The final gds file will be in the smallest database unit. If incompatible database units are found, an error message will be issued.

## GDSReticle V6.3.1

### Bug Fixing

#### The output of fgdsout is erroneous when the total size of the gds files is huge(DDIsa07418)

In GDSReticle V6xx releases, the output of *fgdsout* became erroneous (and unusually huge) when the total size of the gds files being merged reach a certain limit. This has been corrected.

## GDSReticle V6.4.0

### New features

#### Index the homogeneously placed patterns(DDIsa07069)

The homogeneously placed patterns will have the names with a index: pattern\_name\_1, pattern\_name\_2 etc.

#### New keyword CENTERED (DDIsa07072)

A new keyword CENTERED is introduced, it is a property which can be defined for a single pattern or for a type of patterns.

If for a pattern CENTERED = TRUE is defined, and if the width of this pattern is smaller than the width of the scribe, then the pattern is placed in the center of the scribe with respect to the width, detached from either side of the scribe.

Example:

```
TYPES {
...
TYPE_A GDSDIR="gdsdir_name" ORIENT=VER CENTERED=TRUE
... }
LAYOUTS {
...
TYPE_A PATTERN_A1 ...
TYPE_A PATTERN_A2 ...
TYPE_B PATTERN_B CENTERED=TRUE GDSFILE="gdsfile_name" ....
...}
```

#### New keywords GUARDMIN and GUARDMAX (DDIsa05460)

Two new keywords GUARDMIN, GUARDMAX have been added. They are used to specify the properties for SPM objects only. They have to be both present and together they define a range for the size of the guard space between the SMP object and the neighboring patterns.

The existing keyword GUARD continue to function as before.

Example:

```
SPM_X SPMX_H GDSFILE="filename.gds" MASTERBBOX=[-440,-100,10000,100] GUARDMIN=30 GUARD-
MAX=60 ORIENT=HOR;
```

#### GUI: Possibility to highlight a selected type (DDIsa07141)

Two buttons are added at the lower part of the small screen on the right, just below the pattern list. Select one of the types by clicking on it and then click on the "highlight" button, the patterns of the selected type will be highlighted. Click on "reset" to return to the normal display.

**GUI: Have a matrix view of 2x2 in xftclout (DDIsa07142)**

A new button “matrix view” is added in the menu bar “Display” of the xftclout interface. Clicking on this button will toggle on and off the matrix view display mode.

When it is toggled on, it will bring up a configuration dialog which will allow the users to define the overlap margins of the external scribes. If the overlap margins are both zero (in X and Y directions), then the frames will be placed side by side. If the X overlap is defined to be x and the Y overlap y, then the frames will overlap x units between the right and left external scribes, and they will overlap y units between the bottom and top scribes. By default, the size of right scribe width is proposed for X overlap margin and the size of bottom scribe width is proposed for Y overlap margin.

Toggle off this button will bring back the normal display.

**Improvements****The keyword DISTRIBUTE changed from a special type to a property (DDIsa07070)**

In release 6.3, the keyword DISTRIBUTE indicate a special type, with at most two patterns belonging to this type. Starting from this release, DISTRIBUTE is a property which can be used by a single pattern or a type. There is no more limit on the number of patterns being placed homogeneously.

Usage: insert DISTRIBUTE=TRUE in the type definition or layout definition, together with the REPEATMIN and REPEATMAX.

Example:

```
TYPES {
...
OPTIC DISTRIBUTE=TRUE GSDIR="gdsdir_name" GDSFILE="gdsfile_name"...
}
LAYOUTS {
OPTIC DISTR_V ORIENT=VER REPEATMIN=4L REPEATMAX=10L...
OPTIC DISTR_H ORIENT=HOR REPEATMIN=6L REPEATMAX=12L...
SOMETYPE PATTERN1 DISTRIBUTE=TRUE ORIENT=HOR REPEATMIN=6L REPEATMAX=12L...
...
}
```

Warning: The fdf using the previous definition of DISTRIBUTION as a special TYPE will NOT continue to work. Only the syntax DISTRIBUTE=TRUE will trigger the homogeneous placement.

**Handling of homogeneously placed patterns with respect to the crossing area (DDIsa07073, DDIsa07074)**

In the previous release, the homogeneously placed patterns get into the crossing areas even if the NOCROSSING is defined. Starting from this release, when NOCROSSING/TRYNOCROSSING is defined, the homogeneously placed patterns are placed close to the crossing area but not inside.

**Homogeneous placement should be done only once (DDIsa07317)**

In the case where the input `fdf` to `fmpplace` is the output of a previous pass of `fmpplace`, the homogeneously placed patterns are already in place, and should not be placed again. Starting from this release of GDSReticle a check has been added to ensure that the homogeneous placement is applied only once, regardless of how many times it is submitted to `fmpplace`.

**Handling of optional patterns when corner exclusion is set(DDIsa07140)**

In previous releases, when `TRYNOCROSSING` is true, if there are mandatory patterns left unplaced, the crossing areas will be released one by one in the attempt to place all the mandatory patterns. Starting from this release, the same procedure is done for the optional patterns as well.

**Provide the possibility to align the SPM patterns as sequences(DDIsa07075)**

Starting from this release, the existing keyword `SQUENCE` may be used in the declaration of a SPM pattern. The sequences of SPM patterns are placed before the other SPM patterns. The patterns in the same sequence will be placed in the same scribe line, if there is enough space. The order of the placement within the same sequence is the order of the declaration.

**Bug corrections****Partially anchored patterns are not placed correctly (DDIsa07208)**

In the previous releases, if a mobile pattern has part of its coordinates fixed, for example `L=xxx` or `B=xxx`, it was not handled correctly. this has been corrected in release 6.4.0.