Agilent Technologies

Advanced Design System 2002

Design Rule Checker

February 2002

Notice

Warranty

Restricted Rights Legend

Contents

Chapter 1: DRC Quick Start

The Design Rule Checker helps ensure that a layout design conforms to the physical constraints required to produce it. These constraints can be a requirement of the design itself, such as reducing noise, or a requirement of the process used to produce the design.

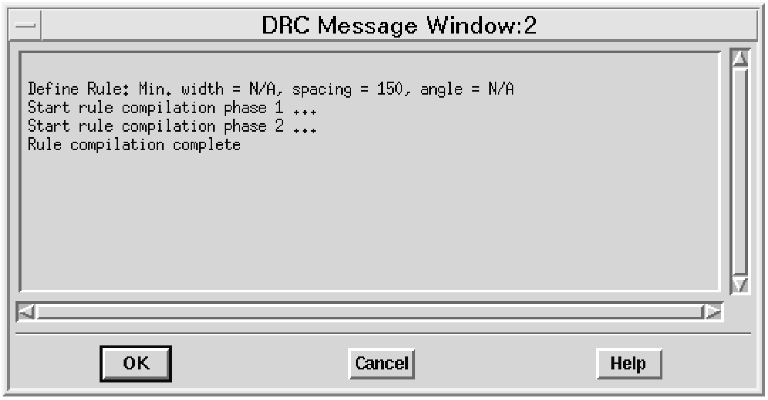
You can run a quick check to ensure conformance to basic design requirements, such as minimum width and spacing, or you can run a custom check using prewritten rules to ensure that a design meets manufacturing specifications. In either case, you can check all or part of the design.

Whether you run a quick check or a custom check, the procedure is essentially the same:

- Define or select a design rule.
- Run the design check using the defined or selected rule.
- Load the results.
- View any errors that were found.

DRC Message Window

The DRC Message window provides information on the status of the current Design Rule check. The window displays as you set up and run a DRC and then displays a summary of in the View Errors panel of the DRC or Custom DRC dialog box.



```
Define Rule: Min. width = N/A, spacing = 150, angle = N/A
Start rule compilation phase 1 ...
Start rule compilation phase 2 ...
Rule compilation complete
```

Rule Registry File

A rule registry file, called setrule.ael, is required in a rule directory to display the list of available rule files by file names.

The format of setrule.ael is as follows:

```
dve_set_rule_list(list (
    <rule_name_1>, <rule_file_1>,
    <rule_name_2>, <rule_file_2>,
    ........
    <rule_name_n>, <rule_file_n>
    ));
```

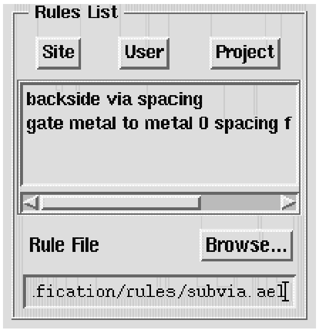
where <rule_name> can be a string that briefly describes the purpose of the rules and <rule_file> is the actual file name.

For example, if you create a setrule.ael file for the Project directory shown below, the Rule Selection dialog displays this list of rules when you click the Project button.

```
// Rule Registry File

dve_set_rule_list(list(
   "Substrate Via Design Rule", "viaRule.ael",
   "NiCr Thin Film Resistor Design Rule", "resistorRule.ael",
   "Gate Metal Spacing Rule", "gateSpacing.ael"
       ));
```

Rule Directories

You can store a rule file in any directory and find the file using the Rule File Browser invoked by the Browse button in the Rule Selection dialog box. However, it is better to use one of the three rule directories supported by the program to facilitate rule file browsing. The three rule directories for storing custom rules are: Site, User, and Project. Buttons corresponding to these directories are available in the dialog box. You can find prewritten design rules at each level.



backside via spacing
gate metal to metal 0 spacing f

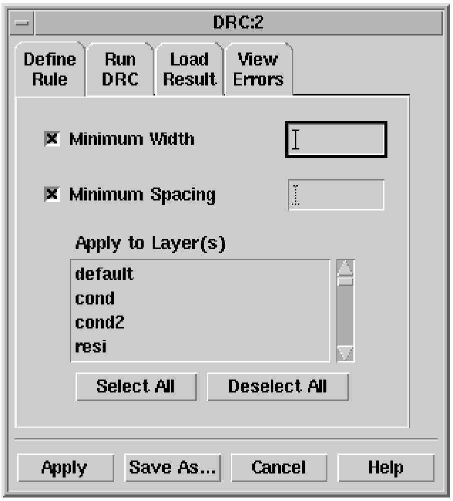
.fication/rules/subvia.ael

Setting Up a Quick DRC

You can use a quick DRC to check selected components or to check an entire design against basic design requirements. After you provide the information needed, the program writes a design rule for you that you can save and reuse again.

To set up a quick DRC:

1. In the Layout window, create a new layout or open an existing one.
2. From the menu, choose **Verify** > **DRC** to open the DRC dialog box. Use the first tab to define a basic design rule to be used.



☒ Minimum Width

☒ Minimum Spacing

Apply to Layer(s)

default
cond
cond2
resi

   Minimum Width defines the narrowest allowable value in the design.

   Minimum Spacing defines the narrowest allowable spacing between shapes in the design.

   Minimum Angle defines the smallest allowable angle in the design.

   Apply to Layer(s) displays a list of the layers in the current design. Choose the layers that you want the rule to apply to.

3. Select the parameter(s) you want to check and enter a value in the selected field. Do not include units when you enter a value in this panel.

Hint The value you enter should produce at least one error, so you can view results.

4. In the Apply to Layer(s) list select the design layer(s) you want checked.
5. Click Apply to start the process.

Setting Up a Custom DRC

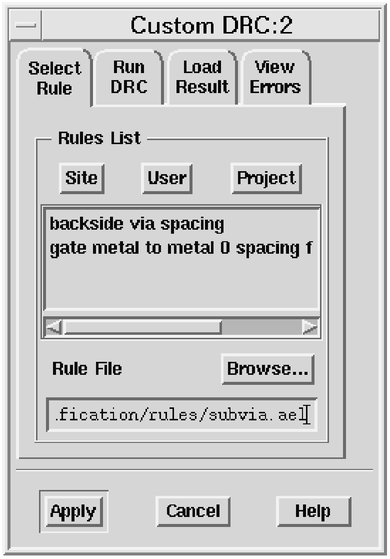
Typically you use a custom DRC to check a design against a manufacturing specification. A custom DRC differs from a quick DRC in two major ways:

- You specify a prewritten design rule.
- You must create a DRC layer in the design on which to display error segments.

To set up a custom DRC:

1. In the Layout window, open an existing layout or create a new one.
2. Choose **Options** > **Layers.** If the design does not have a *drc* layer, create one.

3. Choose **Verify** > **Custom DRC** to open the Custom DRC dialog box at the Select Rule tab.



backside via spacing
gate metal to metal 0 spacing f

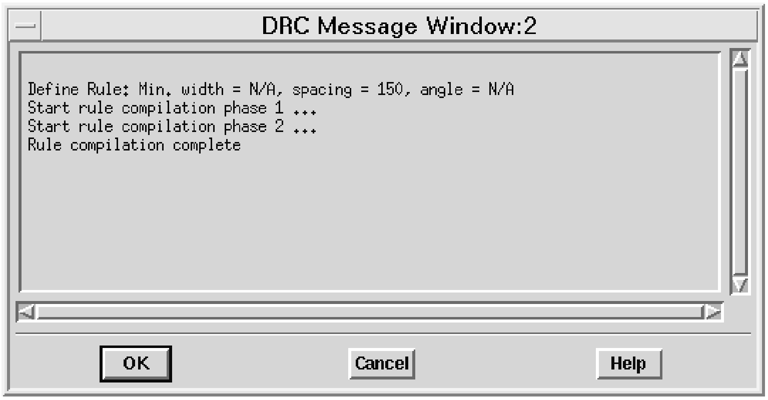
.fication/rules/subvia.ael

- The *Rules List* displays the design rules in the current project directory.
- The *Rules File* displays the selected rules file.
- *Browse...* displays the Select Rules File dialog box where you can select a rules file from a different project directory.

For details, see "Rule Directories" on page 1-3 and "Rule Registry File" on page 1-2.

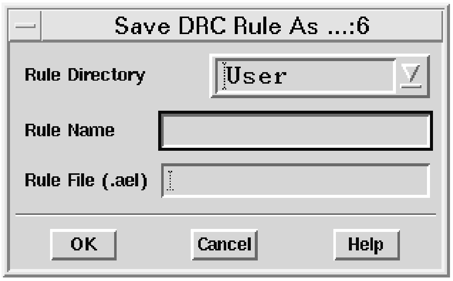
4. Choose Site, User, or Project to view predefined rules.
5. Select a rule from the Rules List. The description of the design rules displays in the selected list. If the rule you want is a rule file, you can browse to find it or you can enter the path and rule file name in the Rule File field.

6. Click Apply to compile the selected rule. The DRC Message Window opens and displays a running message similar to the example:



```
Define Rule: Min. width = N/A, spacing = 150, angle = N/A
Start rule compilation phase 1 ...
Start rule compilation phase 2 ...
Rule compilation complete
```

7. Click OK to save the file.

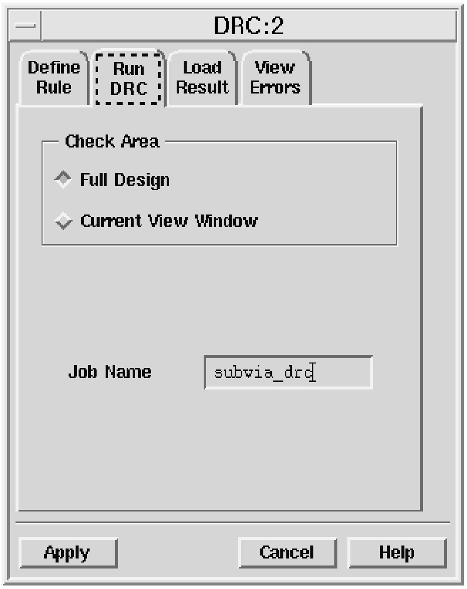


Rule Directory: User
Rule Name:
Rule File (.ael):

Running a DRC

To run a DRC:

1. In the DRC dialog, click **Run DRC**.
2. In the Check Area, choose whether you want the program to check the entire layout or only the area that is currently visible in the Layout window. You can save time on large designs by checking only the area of concern.



Check Area
[✓ Full Design]
Current View Window
Job Name: subvia_drc

3. You can accept the default Job Name or enter a different name. The default Job Name is the design name with the suffix _drc . In either case, include the suffix.
4. Click Apply to start the process.

   A message similar to the example displays in the message window:

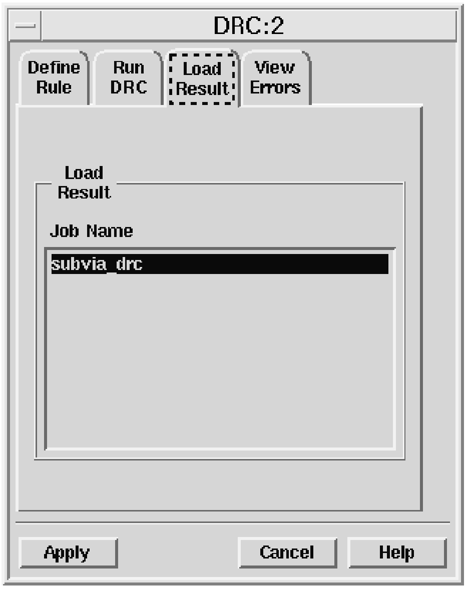
```
Run DRC Job <Job Name>_drc for full design...
DRC process complete
```

Viewing DRC Results

After running a DRC check, you can view the results. See also "Reloading DRC Results" on page 1-13 and "Viewing DRC Errors" on page 1-10.

To view results:

1. In the DRC dialog, select **Load Result**.
2. In the Job Name list, select the job name you want to view.



3. Click **Apply** to view the results. A message similar to the example displays in the message window.

```
Load results <Job Name>_drc
Load results complete
```

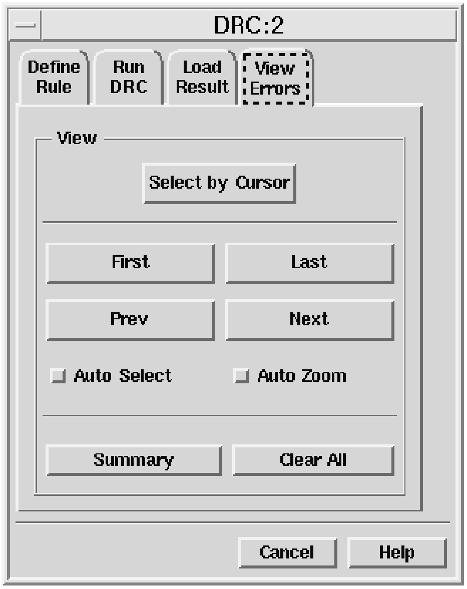
Viewing DRC Errors

After running a design rule check, you can view a summary of the results and any errors found by the check.

Note For a quick DRC, the program automatically creates a drc layer on which error segments are displayed. For a custom DRC, the program does not create a drc layer automatically. You must create an appropriate drc layer(s) before you run the check.

To view errors:

1. In the DRC dialog, select **View Errors**.



2. Click **First.**

   If there are no errors, the message window displays:

   ```
   No DRC error exists!
   ```

   If at least one error exists, the error segment(s) in the Layout window are highlighted and the message window displays:

```
Error #1:
<design rule>
```

where <design rule> is what you defined previously in the Define Rule tab. For example:

```
Width of layer cond must be >= 25.00
```

3. Enable **Auto Select** and **Auto Zoom**, then click **First** again. In the Layout window, the program zooms in on the area that contains the first error and the error segments are selected so you can delete the DRC segments as you fix problems in the layout.
4. Click **Next**. If there is more than one error, the message window displays the next error and the program moves the zoomed display in the Layout window to the area that contains the next error.
5. Click **Summary**. The message window displays a summary similar to the example.

```
Job Name: example_drc
Design Name: example
Design Rule: <current project directory
path>/verification/autorule.ael
Total Number of Errors: 1
```

Note If you prefer, you can view the summary *before* you view any errors.

To view specific error types:

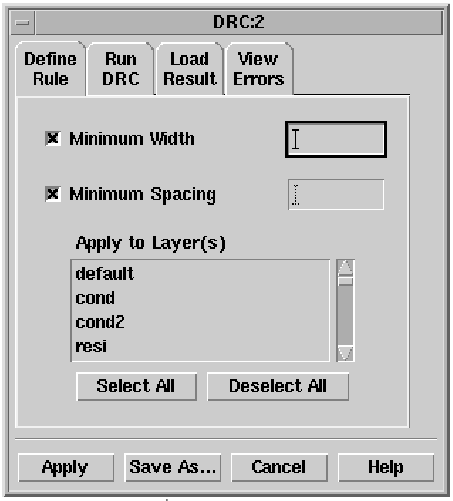
1. In the Layout window, choose **Options > Preferences** > **Select.**
2. Turn off all Select Filters except the specific error type you want to view (for example, Polylines).
3. Click **Select by Cursor** and experiment with selecting errors by dragging a select box around areas in the layout where errors are indicated.
4. Click OK to dismiss the message window, Cancel to dismiss the dialog box.

Saving a DRC Rule

You can save the rule that is created when you run a Quick DRC or when you write a Custom Design Rule. You can define a name for the rule, a name for the AEL file, and where you want the file stored before you define the rule or you can save the rule after you create it. When you save a Design Rule, the program automatically updates the rule registry file to include the new rule (see "Rule Registry File" on page 1-2).

To save a design rule:

1. After defining a rule, click **Save As** in the Define Rule tab.



☒ Minimum Width
☒ Minimum Spacing

default
cond
cond2
resi

Select Save As

2. In the Save DRC Rule As dialog:
   - Select the **Rule Directory** (User or Project) where you want to save the rule.
   - Enter a **Rule Name** that describes the rule briefly.
   - In the **Rule File (.ael)** field, enter a name for the rule file, with a suffix .ael.

   For details, refer to "Rule Directories" on page 1-3 and "Rule Registry File" on page 1-2.

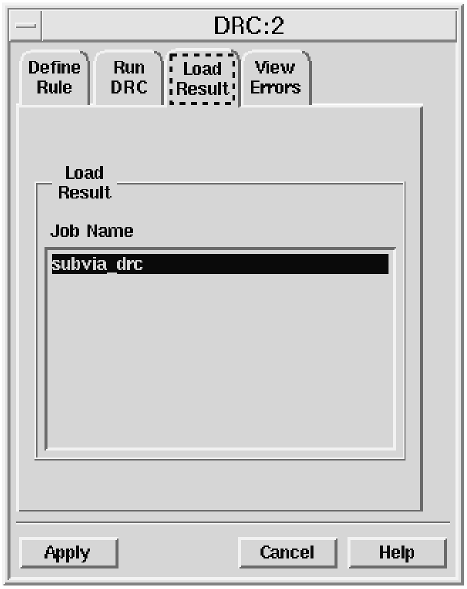
Reloading DRC Results

When you run a DRC, you specify the job name to which the program saves the results of the check. Often a designer has several different design rule files for a given design. All the DRC run results are saved, so you can reload these results when required.

To reload DRC results:

1. In the DRC dialog, select **Load Result**.
2. Select the **Job Name** for the results you want to view.

Select job name



subvia_drc

3. Select Apply to display the results.

Viewing DRC Examples

The Advanced Design System examples directory contains many DRC examples. Examples are constantly improved and new ones are added, so the files in your program may differ from what is shown here. However, the basic path is the same.

To view an example:

1. In the Advanced Design System Main window, select the Examples button on the toolbar.
2. In the File Browser, select the directory path to the example design. Figure 1-1 illustrates the path to the pwramp design:
   examples/MW_Ckts/LNA_Prj/drc_via_prj/networks/pwramp.dsn

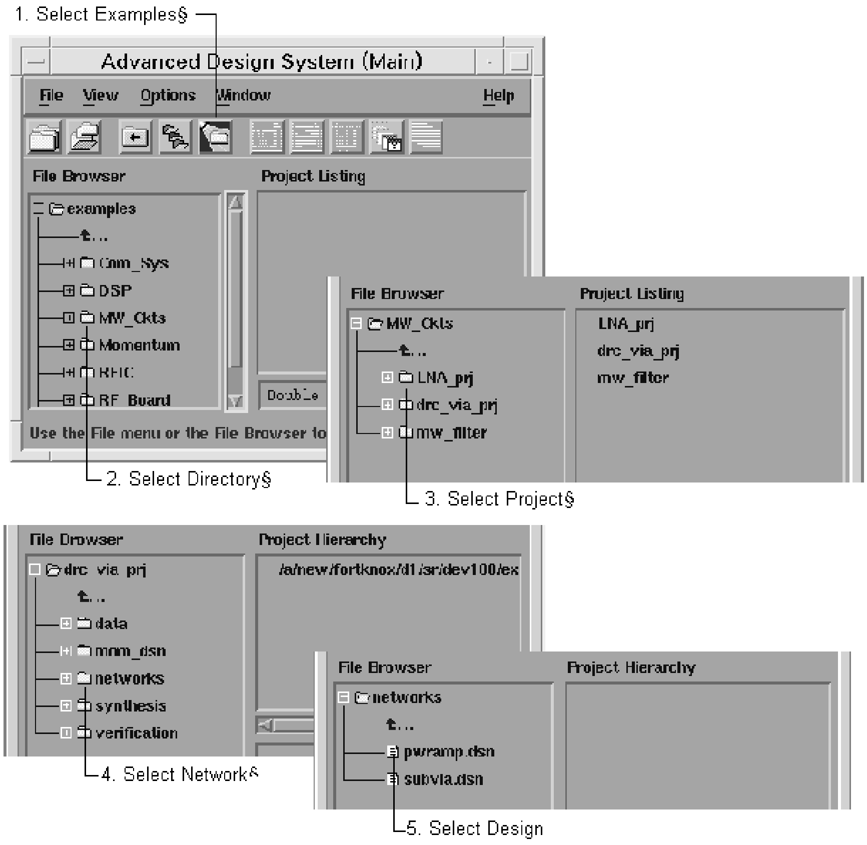


Figure 1-1. Selecting an Example Design

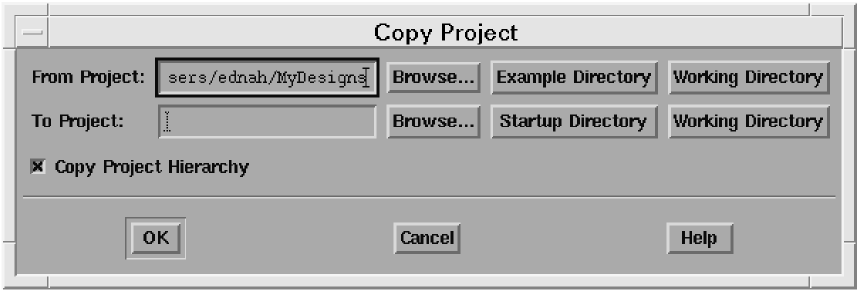
Copying a DRC Example

The files in the examples directory are read-only, so you must copy them to your directory before you can run the examples.

To copy a DRC example:

1. In the Main window, select **File** > **Copy Project**.

2. In the Copy Project dialog box, select **Example Directory**.



3. Select **Browse**.
4. In the Copy From File Browse dialog box, double-click the project directory.
5. From the list of files in the selected project directory, select a project.
6. Click OK.
7. In the Copy Project dialog box, click **Startup Directory** as the To Project.
8. Click **Browse**.
9. In the Copy To File Browse dialog box, select the project directory, then click OK.
10. In the Copy Project dialog box, click at the end of the path displayed as the To Project and enter a file name (including suffix) for the copied project.
11. Confirm that Copy Project Hierarchy is selected.
12. Click OK to copy the project.

Chapter 2: Writing Design Rules

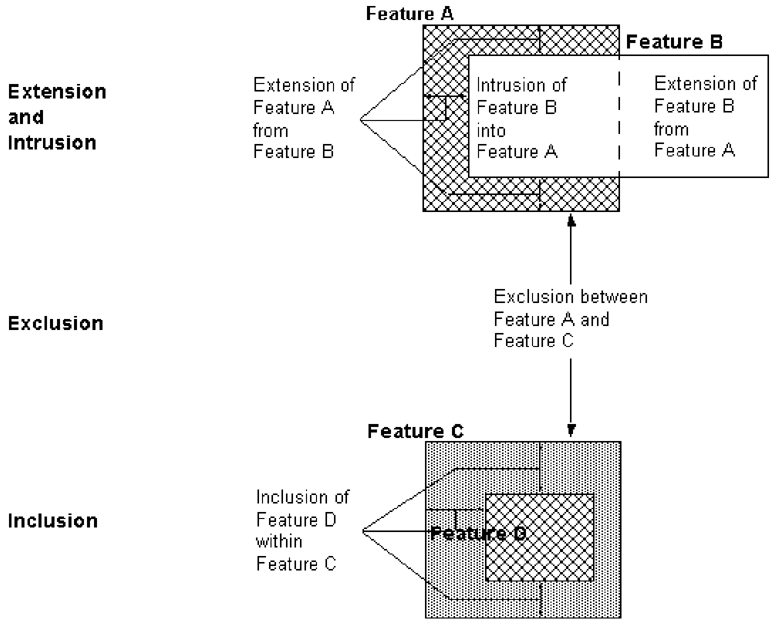
This chapter provides information for writing design rules. Design verification rules produce this information:

- Graphical data showing the location of each violation.
- An error message showing the nature of the violation.

A complete DRC example is included in this section. For detailed information on specific commands, see the command reference chapters.

Extension and Intrusion Definitions

The terms, Extension and Intrusion, used in creating design rules, are defined in the following illustration.



Feature A
Feature B
Extension and Intrusion
Extension of Feature A from Feature B
Intrusion of Feature B into Feature A
Extension of Feature B from Feature A
Exclusion
Exclusion between Feature A and Feature C
Feature C
Inclusion
Inclusion of Feature D within Feature C
Feature D

Anatomy of a Simple DRC Rule File

A DRC rule file is written in Application Extension Language (AEL). The illustration shows a simple DRC rule file. Typically, a rule file consists of a Layer section and a Rule section. The Layer section declares all the design layers used or checked and all the output DRC layers for displaying errors. The Rule section consists of rule checking statements.

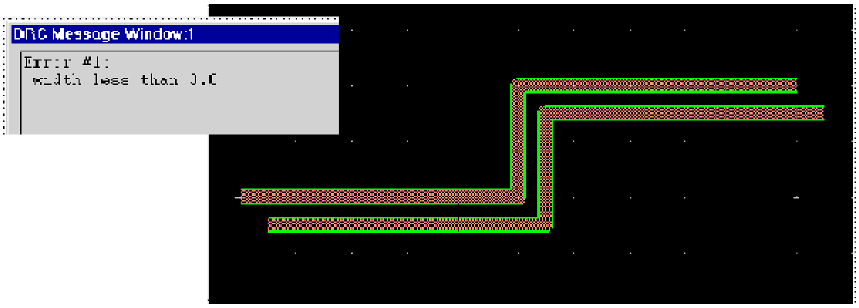
```
// ael rule file: subvia.ael
// Purpose: To check Via to Via spacing rule

//  declare input design layers

decl backVia = dve_import_layer(20);

// declare output layer                                   Layer Section

decl lyrDRCError = dve_export_layer(101);

//
// Substrate Via Spacing Design Rules
// Rule A - Substrate Via to Via minimum spacing 150 um
//
                                                                        Rule Section
lyrDrcError += dve_drc(gap(backVia) < 150,
             "Substrate via edge to via edge min. is 150 um"
                                 );
```

Note A comment starts with a // or is enclosed by `/*` and `*/`.

Layer Management

The rules file illustrated in this section analyzes data on the physical design layer cond. The width command checks the inside clearance distance between edges of the same polygon. Edges that are less than 3.0 layout units apart are exported as line segments to design layer error101. Each violation has an associated error message: width less than 3.0.

The AEL variable lyrCond references an import layer and the AEL variable lyrError101 references an export layer.



```
Error #1:
 width less than 3.0
```

Import Layers

When performing a design rule check, you must specify the design layers you want checked for design violations. Design layers from your layout design are imported into the verification process using the command *dve_import_layer*.

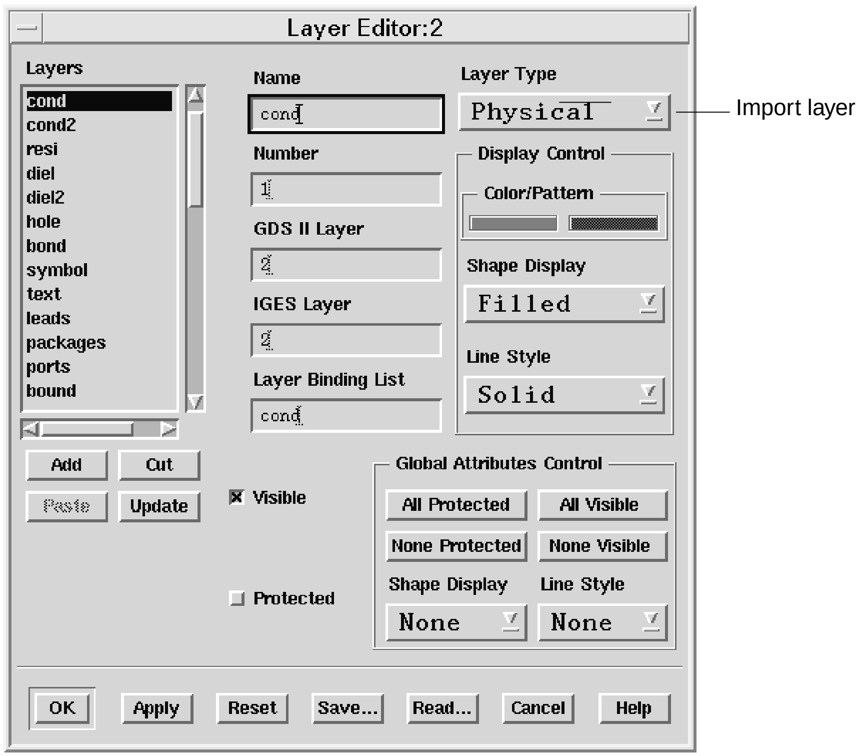
You can specify an import layer by using a layer name or a layer number:

```
decl lyrCond = dve_import_layer ("cond");
```

or

```
decl lyrCond = dve_import_layer (1);
```

Import layers can be used only as input to a DRC command. An import layer must be an existing *Physical* design layer and can only be used for import (that is, it cannot appear again on the left-hand side of a rule command).



Layers: cond, cond2, resi, diel, diel2, hole, bond, symbol, text, leads, packages, ports, bound

Name: cond

Layer Type: Physical

Number: 1

GDS II Layer: 2

IGES Layer: 2

Layer Binding List: cond

Shape Display: Filled

Line Style: Solid

[✓ Visible]

Protected

Shape Display: None

Line Style: None

Export Layers

Data is exported back to the layout editor by sending the output of the *dve_drc* command to an export layer. You create export layers using the command *dve_export_layer*.

You can specify an export layer by using a layer name or a layer number:

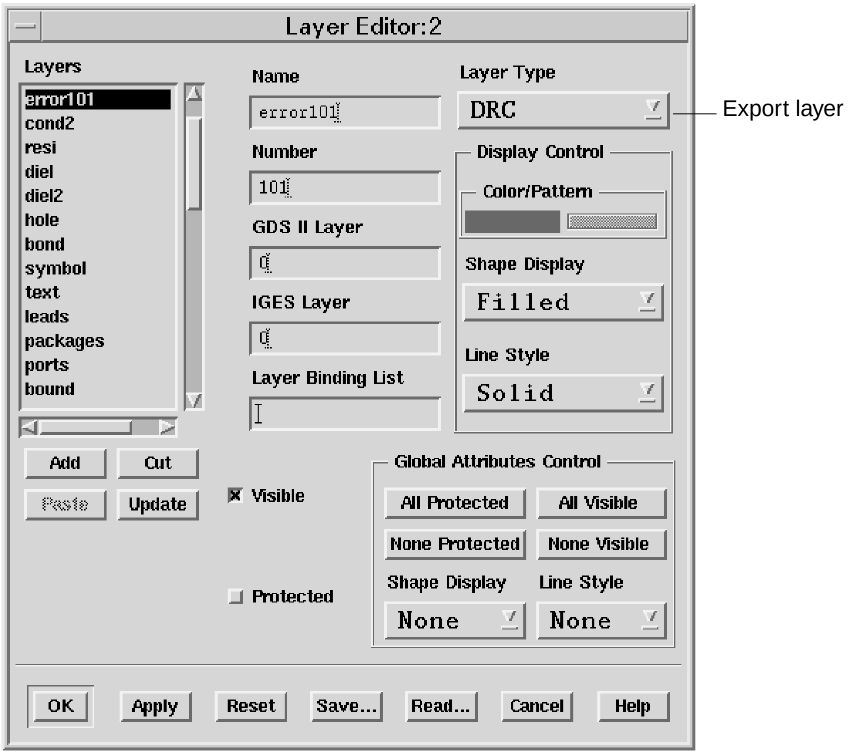
```
decl lyrError101 = dve_export_layer ("error101");
```

or

```
decl lyrError101 = dve_export_layer (101);
```

An export layer must be an existing DRC design layer. The Design Rule Checker will not display DRC errors on a Physical layer.

When sending a DRC error to an export layer, the += assignment is used to signify that you are performing an append operation. Export layers are always empty at the beginning of each DRC invocation, so it is safe to use the += append assignment when sending data to an export layer.



Layers: error101, cond2, resi, diel, diel2, hole, bond, symbol, text, leads, packages, ports, bound

Name: error101

Number: 101

GDS II Layer: 0

IGES Layer: 0

Layer Binding List:

Layer Type: DRC

Shape Display: Filled

Line Style: Solid

[✓ Visible]

Protected

Global Attributes Control: Shape Display: None; Line Style: None

Export layers cannot be used as input to a DRC command. Export layers can appear only on the left-hand side of a rule command.

Work Layers

Work layers are used to reference intermediate data generated by a rule command. Work layers exist only temporarily while the DRC process is running, and are not part of the layout editor environment.

Use work layers when it is necessary to filter or process data on an import layer before generating a DRC error.

As good practice, you should always initialize a work layer to *NULL*.

Rules File Layers Example

This rules file example analyzes physical design data on layers cond and cond2. New polygons are created that represent the area where polygons on layer cond overlap polygons on layer cond2. The new polygons are placed in a work layer lyrPolyOverlap.

The *all_edges* command identifies the entire polygon as an error and the data is exported to DRC layer error101.

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");
decl lyrPolyOverlap = NULL;
lyrPolyOverlap = dve_bool_and (lyrCond, lyrCond2);
lyrError101 += dve_drc (all_edges (lyrPolyOverlap),
    "Conductive metal cond overlaps cond2");
```

Complete DRC Example

The example in this section illustrates writing design rules for Substrate Vias and NiCr Thin Film Resistors and manufacturing rules for Gate Metal. The example covers most of the functionalities and features of the DRC commands.

Note The DRC file used in this example is included in the drc_via_prj directory of the program's examples directory. For information on accessing the examples directory, see “Viewing DRC Examples” on page 1-14.

To set up a DRC check, you must define the design layers and the error layers. For information on setting up a DRC, refer to “Defining the Design Layers” on page 2-7 and “Defining the Error Layers” on page 2-8.

Table 2-1 shows the layer definitions for the process used in this example.

Table 2-1. Layer Definition

Mask Level	Layer	Description
Alignment Key	13	Defines fields in which alignment artifacts will be etched.
N+ Implant	2	Mask during alignment artifact etch, Implant mask for N+ regions.
D- Implant	1	Implant mask for DFET channels, Half DFET Diodes, D-Resistors.
NiCr	3	Liftoff layer for NiCr Resistors
Ohmic	5	Liftoff layer for ohmic contact on GaAs devices. Ohmic Metal may NOT be used for interconnect.
Isolation Implant	6	Implant mask for Isolation Implant
Gate Metal	7	Liftoff layer Schottky Gate/Anode contact on GaAs devices. Gate Metal may NOT be used for interconnect
Metal 0	9	Liftoff layer for Metal 0
MIM	23	Liftoff layer for MIM metal
Via 1	14	First via etch layer
Metal 1	15	First plated Au metal layer. Labels are done in this layer
Air Bridge Post	10	Support Posts for Air Bridge and Via to Metal1
Air Bridge	11	Second plated Au metal layer
Passivation Via	12	Opens vias over bond pads and saw streets
Backside Via	20	Via holes (Via Option Only)
Backside Via Coat	21	Prevent solder wetting in vias (Via Option Only)

Defining the Design Layers

The rule section declares these imported design layers:

```
//  declare input design layers
decl nImplant = dve_import_layer(2);
decl dImplant = dve_import_layer(1);
decl niCr = dve_import_layer(3);
decl ohmic = dve_import_layer(5);
decl isoImplant = dve_import_layer(6);
decl gateMetal = dve_import_layer(7);
decl metal0 = dve_import_layer(9);
decl mIM = dve_import_layer(23);
decl via1 = dve_import_layer(14);
decl metal1 = dve_import_layer(15);

decl airBridgePost = dve_import_layer(10);
decl airBridge = dve_import_layer(11);
decl passVia = dve_import_layer(12);
decl backVia = dve_import_layer(20);
decl backViaCoat = dve_import_layer(21);
```

Although every layer is declared here, you do not need to declare a design layer if you will not be checking it. This example does not use all of these layers, because you are not checking the complete design.

Defining the Error Layers

After defining the design layers, declare three DRC error layers to display errors from a set of rules. When writing DRC rules, you decide how many DRC error layers are needed to best view the results of a check.

```
// declare some DRC error layers
decl viaError = dve_export_layer(107); // for substrate via design rule
decl niCrError = dve_export_layer(103); // for thin film resistor rule
decl gateMetalError = dve_export_layer(120); // for gate metal rule
```

Checking the Clearance Rules

DRC checks clearance rules by selecting the edges that violate the clearance constraints and sending these to a DRC error layer. Clearance rules can be checked from either inside or outside of an edge to another edge of polygons. The types of clearance rules are: width, spacing, external, contains, nests, and internal. Of these, the simplest rule is width.

width

The width command is used to check the width of polygons on a given layer. The command checks the distance from the inside of one edge to the inside of another edge of the same polygon.

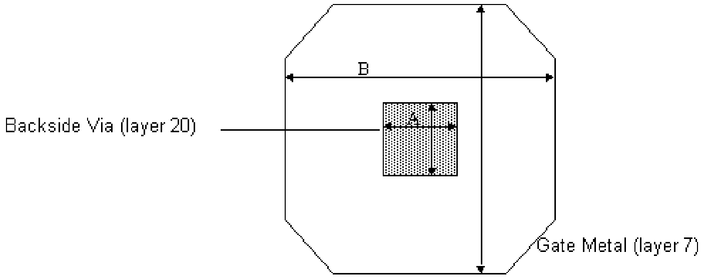


Table 2-2. Substrates Via Design Rules

Item	Description	Minimum (um)
A	Coded Substrate Via Feature, Square (layer 20)	30
B	Substrate Via Target (layer 7)	120

Width rules for the substrate via are written as follows:

```
// Rule A: substrate via feature minimum 30 um

viaError += dve_drc(width(backVia) < 30,
    "Substrate via feature size < 30");

// Rule B: Substrate Via target size minimum 120 um

viaError += dve_drc(width(gateMetal) < 120,
    "Substrate via Target size < 120");
```

spacing

The spacing command is used to checked spacing constraints on a given layer. The command checks the distance from the outside of an edge to the outside of another edge.

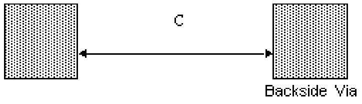
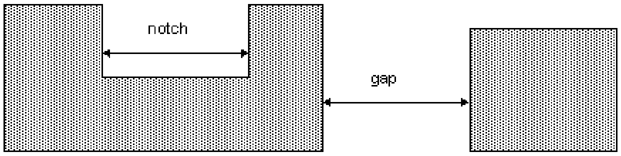


Table 2-3. Substrate Via Design Rule

Item	Description	Minimum (um)
C	Substrate Via (layer 20) to Via (20), Edge to Edge	150

```
//
// Substrate Via Spacing Design Rule
// Rule C - Substrate Via to Via minimum spacing 150 um
//

viaError += dve_drc(spacing(backVia) < 150,
    "Substrate via edge to via edge min. is 150 um");
```

Two other simple spacing rule commands are notch and gap. The notch command checks the spacing within the same polygon and the gap command checks the spacing between two different polygons. The spacing command checks both cases**.**



Checking Clearance Between Layers

All the clearance commands mentioned to this point work only on polygons that are on the same layer. Next you will see clearance commands that check the clearance from one layer to another. The layers checked can be a design or work layer, so you cand send a design layer to a work layer and perform a two-layer rule command with the original design layer. An example of this capability is shown in the "Using Rule Conjunction" on page 2-19.

external

The external command checks the external spacing between polygons on two different layers.

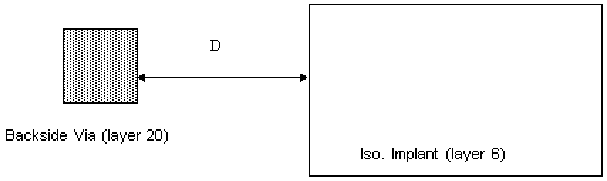


Table 2-4. Substrate Via Design Rule

Item	Description	Minimum(um)
D	Substrate Via (layer 20) to Active Device Edge (layer 6)	90

```
//
// Rule D - Substrate Via to Iso. Implant minimum spacing 90 um
//

viaError += dve_drc(external(backVia, isoImplant) < 90,
    "Substrate via edge to Iso. Implant Edge min. is 90 um");
```

contains

The contains command is used to check the inclusion of one polygon within another polygon. The command checks the distance from the inside edge of polygons on the first layer to the outside edge of polygons on the second layer.

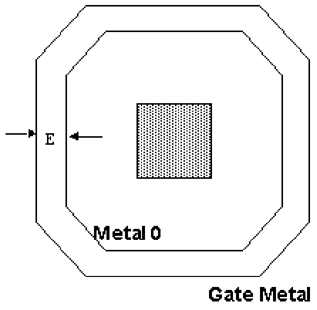


Table 2-5. Substrate Via Design Rule

Item	Description	Minimum(um)
E	Metal 0 (layer 9) Inclusion in Gate Metal (layer 7)	1.0

```
//
// Rule E - Metal 0 Inclusion in Gate Metal min is 1 um

gateMetalError += dve_drc(contains(gateMetal, metal0) < 90,
    "Metal 0 Inclusion in Gate Metal min is 1 um");
```

You can use the contains command to check the extension of one polygon outside another polygon on a different layer. The illustration uses this design rule on NiCr Thin Film Resistors.

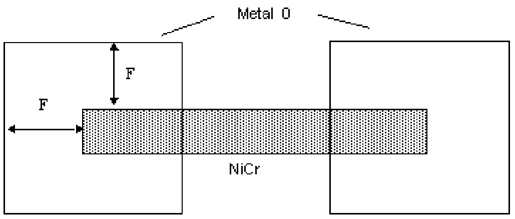


Table 2-6. NiCr Thin Film Resistors Design Rule

Item	Description	Minimum(um)
F	Metal 0 (layer 9) Extension from NiCr (layer 3)	0.5

```
// Rule F - Metal 0 Extension from NiCr  min is 0.5 um
//
niCrError += dve_drc(contains(metal0, niCr) < 0.5,
    "Metal 0 Extension from NiCr  min is 0.5 um",
        DVE_RN_EDGE_ANGLES, DVE_RV_PARALLEL);
```

nests

The nests command checks the distance from the outside edge of polygons on the first layer to the inside edge of polygons on the second layer. It is exactly the same command as the contains command except the two layer arguments are switched.

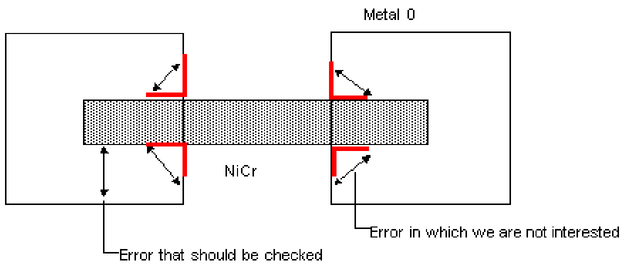
The example writes the previous extension rule (Rule F) using the nests command.

```
// Rule F - Metal 0 Extension from NiCr  min is 0.5 um
//
niCrError += dve_drc(nests(niCr, metal0) < 0.5,
    "Metal 0 Extension from NiCr  min is 0.5 um",
        DVE_RN_EDGE_ANGLES, DVE_RV_PARALLEL);
```

Notice that a qualifier was used in Rule F. A qualifier is defined as a name-and-value-pair:

Qualifier_Name, Qualifier_Value

Clearance Rule Qualifiers filter in (or out) tests between pairs of edges for a rule step. If no qualifier is specified, a rule command normally checks all the edge pairs. However, in this example, we are interested only in the edge pairs that are parallel to each other. Without the Parallel qualifier, we would get an unpleasant surprise from errors caused by non-parallel edges as shown in the following figure. Remember, *contains* checks from outside of the first polygon (on NiCr) to the inside of the second polygon (on Metal 0).


Metal 0
NiCr
Error in which we are not interested
Error that should be checked

A width command appears to work well without a qualifier. What happens to the adjacent edges? Actually, the width command has a default qualifier to filter out all the adjacent edges during the rule operation:

DVE_RN_SEPARATE, DVE_RV_SEPARATE

Nearly all clearance commands have some type of default qualifiers to tell how the rule works. An example would be the Polarity qualifier. The fact that a command checks from the inside (or outside) of an edge to the inside (or outside) of another edge is dictated by the Polarity qualifier.

Two generic clearance commands (single_clearance and double_clearance) demand a polarity qualifier to tell them what to check. The single_clearance command is equivalent to a width command:

```
dve_drc(single_clearance(layer) < distance,
    DVE_RN_POLARITY, DVE_RV_INSIDE);
```

The double_clearance command is equivalent to a contains command:

```
dve_drc(double_clearance(layer1, layer2) < distance,
    DVE_RN_POLARITY_FROM, DVE_RV_INSIDE,
    DVE_RN_POLARITY_TO, DVE_RV_OUTSIDE);
```

Details on the qualifiers for a given command are provided in the command reference chapters.

internal

This internal command checks the distance from the inside edge of one polygon to the inside edge of another polygon. The command is used to check the intrusion from one polygon into another polygon.

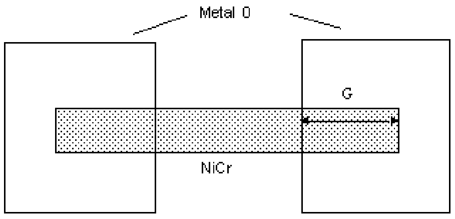


Table 2-7. NiCr Thin Film Resistors Design Rule

Item	Description	Minimum(um)
G	NiCr (layer 3) Intrusion into Metal 0 (layer 9)	2.5

```
//
// NiCr Thin Film Design Rules
// Rule G - NiCr  Intrusion into Metal0  min is 2.5 um
//

niCrError += dve_drc(internal(niCr, metal0) < 2.5,
        "NiCr  Intrusion into Metal0  min is 2.5 um");
```

Selecting Polygons

Several polygon selection commands are provided. In this example, only the poly_path_length and poly_inter_layer commands are described, but all polygon selection commands work similarly. For more details, see Chapter 4, Conditional Selection.

poly_path_length

The command poly_path_length selects polygons based on the path length property of overlapping polygons on two layers.

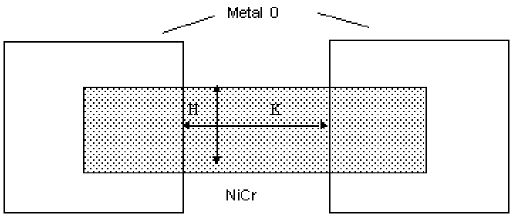


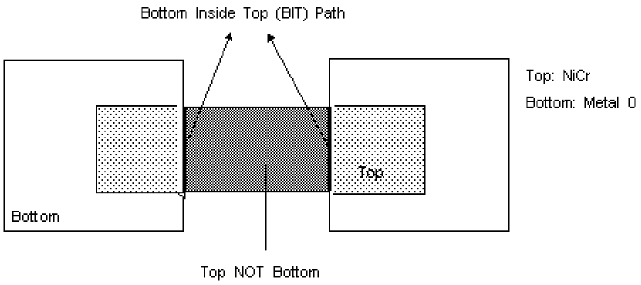
Table 2-8. NiCr Thin Film Resistors Design Rules

Item	Description	Minimum(um)
H	Resistor (layer 3) Width	2.0
K	Resistor (layer 3) Length	3.0

To check the width of a Thin Film Resistor, first do a boolean merge-NOT between the NiCr and Metal 0 layers to produce the resistor polygons. The path consisting of Bottom Inside Top (BIT) edges is the width of the resistor (see the illustration). Then select the bad resistors by checking the Bottom Inside Top (BIT) path length.

For details on determining the path code from merged polygons, refer to "Polygon Selection Based on Merge Properties" on page 4-28.

In this rule example, you begin to use work layers . Also, the result of a *poly_path_length* command is a polygon layer, so you need an *all_edges* command to send the polygon layer to a DRC error layer for displaying.



```
// declare some work layers

decl lyrResistor, widthShort;

//
// NiCr Thin Film Design Rules
// Rule H - Resistor width min is 2um
//
// To produce the resistor polygons
lyrResistor = dve_bool_not(niCr, metal0);

// Select if the BIT path length is less than 2
widthShort = dve_drc(poly_path_length(lyrResistor) < 2,
    DVE_RN_PATH_CODE, DVE_RV_BIT,    // set path code
    DVE_RN_PATH_LENGTH, DVE_RV_MIN_PATH  // check minimum
);

// Attach error message & send error polygons to DRC error layer
niCrError += dve_drc(all_edges(widthShort),
   "NiCr Thin Film Resistor min width 2.0 um"
);
```

The Rule K checks the length dimension of a resistor. It does not require a poly_path_length command, you can implement this rule by using a boolean command and a clearance command. Try this as an exercise.

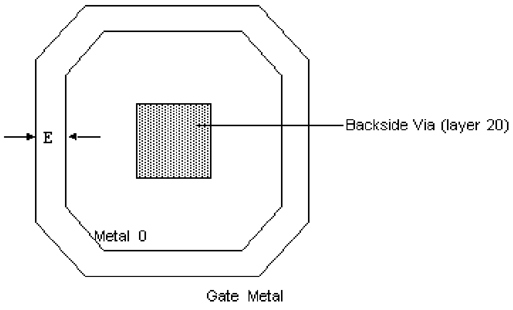
poly_inter_layer

The command poly_inter_layer selects polygon based on its relationship to another polygon. The command is very useful for selecting a subset of polygons out of a polygon layer and then performing a rule check on the subset.

Table 2-9. Substrate Via Design Rule

Item	Description	Minimum(um)
E	Metal 0 (layer 9) Inclusion in Gate Metal (layer 7)	1.0

Go back to rule *E*, which was done previously without filtering out unwanted polygons before applying the clearance command. This rule catches many errors that occur outside of substrate vias because both the Metal 0 and Gate Metal layers are used in the construction of other devices (such as DFET). The clearance rule brings in all of the polygons from these two layers, including the polygons used for DFET.



Fortunately, you can tell when a Metal 0 or a Gate Metal polygon is used for a substrate via: it must enclose a polygon from the Backside Via layer (layer 20), as shown on the illustration. The poly_inter_layer command is used to select polygons like this. Here is the rewritten Rule E:

```
//
// Substrate Via Spacing Design Rules
// Rule E - Metal 0 Inclusion in Gate Metal min is 1 um
//

// declare some work layers

decl viaGateMetal,viaMetal0; // these are work layers,
                             // that do not map to a real
```

```
                              // process layer
//
// First, derive gate metal used for substrate vias by using
// only the gate metal that encloses the backside via layer
//
viaGateMetal = dve_drc(poly_inter_layer(gateMetal, backVia),
    DVE_RN_INTER_CODE, DVE_RV_ENCLOSE_ONLY);
//
// In a similar way, derive the metal0 used for substrate vias
//
viaMetal0 = dve_drc(poly_inter_layer(metal0, backVia),
    DVE_RN_INTER_CODE, DVE_RV_ENCLOSE_ONLY);
//
//  Use contains cmnd to check Inclusion between 2 work layers
//
viaError += dve_drc(contains(viaGateMetal, viaMetal0) < 1,
    "Metal 0 Inclusion in Gate Metal min is 1 um");
```

You can use the poly_inter_layer to detect whether two polygon layers overlap in a wrong manner. The command selects polygons by filtering in or out the overlapping conditions, such as Inside, Outside, Touch, and Cut, and then sends the polygons through an all_edges command to a DRC error layer. For more details, see "poly_inter_layer()" on page 4-35.

Using Rule Conjunction

In general, the result of deriving a work layer from one rule command and later feeding that work layer to another rule command is the combining of more than one rule constraint. This is called rule conjunction. In fact, you have seen rule conjunction in earlier examples of polygon selection commands. Here a more complicated example shows how to use rule conjunction to check Gate Metal manufacturing rules.

Table 2-10. Gate Metal Manufacturing Rules

Item	Description		Minimum (um)
L	Gate Metal (layer 7) spacing when	width < 1.5	1.0
		1.5 <= width < 2.0	1.5
		2.0 <= width < 3.0	2.0
		3.0 <= width	3.0

```
// declare output layer
decl gateMetalError = dve_export_layer(120);
//
```

```
// Gate Metal spacing Rule
// Rule L - Min. spacing is
//     1.0 if width < 1.5
//     1.5 if 1.5 <= width < 2.0
//     2.0 if 2.0 <= width < 3.0
//     3.0 if        width >= 3.0

// declare some work layers

decl gatMet15Lt, gatMet15Ge, gatMet20Lt, gatMet20Ge;
decl gatMet30Lt, gatMet30Ge;

// Rule: Min. spacing is 1.0  if width < 1.5
// 1. select the edges with width < 1.5 from gateMetal, save in
//    gatMet15Lt
// 2. select the edges with spacing error by checking the distance
//    between gateMetal and gatMet15Lt

gatMet15Lt = dve_drc(width(gateMetal) < 1.5);
gateMetalError += dve_drc(external(gateMetal, gatMet15Lt) < 1.0,
    "Gate Metal min spacing 1.0um when its width < 1.5um");

// Rule: Min. spacing is 1.5  if  1.5 <= width < 2.0
// 1. select the edges with width >= 1.5 from gateMetal, save in
//    gatMet15Ge
// 2. select the edges with width < 2.0 from gatMet15Ge, save in
//    fateMet20Lt
// 3. select the edges with spacing error by checking the distance
//    between gateMetal and gatMet20Lt

gatMet15Ge = dve_drc(width(gateMetal) >= 1.5);
gatMet20Lt = dve_drc(width(gatMet15Ge) < 2.0);
gateMetalError += dve_drc(external(gateMetal, gatMet20Lt) < 1.5,
    "Gate Metal min spacing 1.5um when its width within [1.5, 2)");

// Rule: Min. spacing is 2.0  if  2.0 <= width < 3.0
gatMet20Ge = dve_drc(width(gateMetal) >= 2.0);
gatMet30Lt = dve_drc(width(gatMet20Ge) < 3.0);
gateMetalError += dve_drc(external(gateMetal, gatMet30Lt) < 2.0,
    "Gate Metal min spacing 2.0um when its width within [2.0, 3)");

// Rule: Min. spacing is 3.0  if width >= 3.0
gatMet30Ge = dve_drc(width(gateMetal) >= 3.0);
gateMetalError += dve_drc(external(gateMetal, gatMet30Ge) < 3.0,
    "Gate Metal min spacing 3.0um when its width > 3.0 um");
```

Congratulations. You have finished writing your first rule file. If you would like to save it to a file, remember to use the file extension .ael. For details, see "Saving a DRC Rule" on page 1-12.

Chapter 3: DRC Layer Management Commands

This section describes the DRC Layer Management commands used to import and export layers.

dve_import_layer()

Used to get design data from the layout editor into the design verification process. Copies layer data from the layout editor onto an import layer that can be used in a rule command. Returns an import layer.

See also: “dve_export_layer()” on page 3-1

Syntax

inputLayer = dve_import_layer (layerId);

where:

layerId is the string layer name or integer layer number of an existing design layer

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (width (lyrCond) < 4.0,
    "Metal width less than 4.0");
```

dve_export_layer()

Used to export DRC error information. Data written to an export layer will be directly exported back to the layout editor. Returns an export layer.

See also: “dve_import_layer()” on page 3-1

Syntax

exportLayer = dve_export_layer (layerId);

where:

layerId is the string name or integer layer number of an existing design layer

Example

```
// Import layers
```

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");

// Export layers
decl lyrError101 = dve_export_layer ("error101");
decl lyrError102 = dve_export_layer ("error102");

// Work layer
decl lyrOverlap = NULL;

// Export DRC error directly to an export layer
lyrError101 += dve_drc (width (lyrCond) < 4.0,
    "Metal width less than 4.0");

lyrOverlap = dve_bool_and (lyrCond, lyrCond2);
lyrError102 += dve_drc (all_edges (lyrOverlap),
    "Metal layers overlap");
```

Chapter 4: Conditional Selection

This section describes the DRC command used for conditional selection.

dve_drc()

Used to select edges and polygons conditionally based upon intrinsic properties and information derived during an operation on one or more layers. Returns: a layer containing selected edge segments.

Syntax

dve_drc (drc_expression [, msgString][, qualifierName, qualifierValue]);

where:

drc_expression is an AEL expression in the format:

drc_subfunction ([parameter, ...]) [operator rValue])

drc_subfunction is a selection function to be performed on the polygons or edges on a given layer. Edges and polygons that meet the criteria are selected and copied to the output layer. The subfunctions are:

Edge Selection Based On Clearance (output layer contains polygons with selected edges) selection functions include: contains, double_clearance, external, gap, internal, nests, notch, single_clearance, spacing, width

Edge Selection By Select All or Inversion selection functions include: all_edges, invert_edges

Edge Selection Based on Corners selection functions include: corner_edges

Edge Selection Based on Grid selection functions include: off_grid

Edge Compensation selection functions include: compensate

Polygon Selection Based on Intrinsic Properties (output layer contains polygons) selection functions include: poly_area, poly_hole_count, poly_line_length, poly_perimeter

Polygon Selection Based on Merge Properties (output layer contains polygons) selection functions include: poly_edge_code, poly_path_count, poly_path_length

Polygon Selection Based on Edge Relationships (output layer contains polygons) selection functions include: poly_inter_layer

parameter A parameter to a dve_drc subfunction command.

operator function A relational function that is applied to the value returned from the dve_drc function. A relational function includes these operators:

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

rValue A real or integer value that depends upon the DRC subfunction.

msgString A string that will be attached to the selected edges. Only pertains to selected edges. Can only be used in conjunction with the export nomenclature (such as, “+=”).

qualifierName A constant the represents the name of the qualifier. Qualifiers are documented for each dve_drc subfunction.

qualiferValue A value that will be applied to the named qualifier. Valid range of values are documented for each dve_drc subfunction.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (width (lyrCond) < 3.0,
    "Width of conductive metal < 3.0");
```

Edge Selection Based On Clearance

The Edge Selection Based On Clearance selection functions are used where the output layer contains polygons with selected edges. These functions include: contains, double_clearance, external, gap, internal, nests, notch, single_clearance, spacing, width.

contains()

A DRC function to measure enclosure distance from the outside of the contained polygon to the inside of the containing polygon.

Syntax

dve_drc (contains (inLayer1, inLayer2) operator distance [, msgString]
  [, qualifierName, qualifierValue...]);

where:

inLayer1 Contained polygon layer.

inLayer2 Containing polygon layer.

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

distance A distance value in layout units.

msgString A string value that will be attached to the selected error segments.

qualifierName, qualifierValue A name, value pair that qualifies the selection.

Edge Qualifiers

Qualifier Resource Name: DVE_RN_EDGE_ANGLES

Qualifier Resource Value:

DVE_RV_PARALLEL Select only parallel edges.

DVE_RV_NOT_PARALLEL	Select only non-parallel edges.
DVE_RV_PERPENDICULAR	Select only perpendicular edges.
DVE_RV_NOT_PERPENDICULAR	Select only non-perpendicular edges.
DVE_RV_ANY_ANGLE	(default) Select edges at any angle.

Qualifier Resource Name: DVE_RN_ANGLE_TOLERANCE

Qualifier Resource Value:

<real value>	Edge angle tolerance in degrees.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (contains (lyrCond, lyrCond2) < 3.0,
    "Enclosure clearance < 3.0");
```

double_clearance()

Measures the distance between edges of polygons on different layers.

See also: "dve_drc()" on page 4-1

Syntax

dve_drc (double_clearance (inLayer1, inLayer2) operator distance

[, msgString] [,qualifierName, qualifierValue...]);

where:

inLayer1, *inLayer2* A polygon or edge layer.

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

distance A value in layout units.

msgString A string value that will be attached to the selected error segments.

qualifierName, qualifierValue A name, value pair that qualifies the selection.

Edge Qualifiers

Qualifier Resource Name: DVE_RN_EDGE_ANGLES

Qualifier Resource Value:

DVE_RV_PARALLEL Select only parallel edges.

DVE_RV_NOT_PARALLEL Select only non-parallel edges.

DVE_RV_PERPENDICULAR Select only perpendicular edges.

DVE_RV_NOT_PERPENDICULAR Select only non-perpendicular edges.

DVE_RV_ANY_ANGLE (default) Select edges at any angle.

Qualifier Resource Name: DVE_RN_ANGLE_TOLERANCE

Qualifier Resource Value:

<real value> Edge angle tolerance in degrees.

Qualifier Resource Name: DVE_RN_POLARITY, DVE_RN_POLARITY_FROM, DVE_RN_POLARITY_TO

Qualifier Resource Value:

DVE_RV_INSIDE Direct search toward inside of polygon

DVE_RV_OUTSIDE (default) Direct search toward outside of polygon

Qualifier Resource Name: DVE_RN_TEMPLATE, DVE_RN_TEMPLATE_FROM, DVE_RN_TEMPLATE_TO

Qualifier Resource Value:

DVE_RV_ARC Extend search area using arced corners.

DVE_RV_BOTHSIDES Extend search area on both sides of edge.

DVE_RV_OPPOSITE (Default) Extend search area just opposite the edge.

DVE_RV_ROUND Extend search area using rounded corners.

DVE_RV_SQUARE Extend search area treating corners as squares.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (double_clearance (lyrCond, lyrCond2) < 3.0,
    "Metal layers run parallel and close",
    DVE_RN_POLARITY, DVE_RV_OUTSIDE,
    DVE_RN_TEMPLATE, DVE_RV_OPPOSITE,
    DVE_RN_EDGE_ANGLES, DVE_RV_PARALLEL,
    DVE_RN_ANGLE_TOLERANCE, 1.2);
```

external()

Measures the distance between outside edges of polygons of different layers.

See also: "dve_drc()" on page 4-1

Syntax

dve_drc (external (inLayer1, inLayer2) operator distance [, msgString]
[, qualifierName, qualifierValue...]);

where:

inLayer1, *inLayer2* A polygon layer

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

distance A distance value in layout units.

msgString A string value that will be attached to the selected error segments.

qualifierName, *qualifierValue* A name, value pair that qualifies the selection.

Edge Qualifiers

Qualifier Resource Name: DVE_RN_EDGE_ANGLES

Qualifier Resource Value:

DVE_RV_PARALLEL Select only parallel edges.

DVE_RV_NOT_PARALLEL	Select only non-parallel edges.
DVE_RV_PERPENDICULAR	Select only perpendicular edges.
DVE_RV_NOT_PERPENDICULAR	Select only non-perpendicular edges.
DVE_RV_ANY_ANGLE	(default) Select edges at any angle.

Qualifier Resource Name: DVE_RN_ANGLE_TOLERANCE

Qualifier Resource Value:

<real value>	Edge angle tolerance in degrees.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (external (lyrCond, lyrCond2) < 4.0,
    "Outside edges of metal layers < 4.0",
    DVE_RN_EDGE_ANGLES, DVE_RV_PARALLEL);
```

gap()

Measures the distance between outside edges of different polygons of the same layer.

See also: "dve_drc()" on page 4-1

Syntax

dve_drc (gap (inLayer) operator distance [, msgString]
[,qualifierName, qualifierValue...]);

where:

inLayer A polygon or edge layer.

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

distance A distance value in layout units.

msgString A string value that will be attached to the selected error segments.

qualifierName, qualifierValue A name, value pair that qualifies the selection.

Edge Qualifiers

Qualifier Resource Name: DVE_RN_EDGE_ANGLES

Qualifier Resource Value:

DVE_RV_PARALLEL Select only parallel edges.

DVE_RV_NOT_PARALLEL Select only non-parallel edges.

DVE_RV_PERPENDICULAR Select only perpendicular edges.

DVE_RV_NOT_PERPENDICULAR Select only non-perpendicular edges.

DVE_RV_ANY_ANGLE (default) Select edges at any angle.

Qualifier Resource Name: DVE_RN_ANGLE_TOLERANCE

Qualifier Resource Value:

<real value> Edge angle tolerance in degrees.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

// Check between outside edges of polygons on same layer
lyrError101 += dve_drc (gap (lyrCond) < 4.0, "Outside edges < 4.0");
```

internal()

Measures clearance from the inside of one edge of a polygon to the inside of another edge of a different polygon.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (internal (inLayer1, inLayer2) operator distance [, msgString] [,qualifierName, qualifierValue...]);

where:

inLayer1, inLayer2 A polygon layer.

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

distance A distance value in layout units.

msgString A string value that will be attached to the selected error segments.

qualifierName, qualifierValue A name, value pair that qualifies the selection.

Edge Qualifiers

Qualifier Resource Name: DVE_RN_EDGE_ANGLES

Qualifier Resource Value:

DVE_RV_PARALLEL	Select only parallel edges.
DVE_RV_NOT_PARALLEL	Select only non-parallel edges.
DVE_RV_PERPENDICULAR	Select only perpendicular edges.
DVE_RV_NOT_PERPENDICULAR	Select only non-perpendicular edges.
DVE_RV_ANY_ANGLE	(default) Select edges at any angle.

Qualifier Resource Name: DVE_RN_ANGLE_TOLERANCE

Qualifier Resource Value:

<real value>	Edge angle tolerance in degrees.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (internal (lyrCond, lyrCond2) < 4.0,
    "Inside edges < 4.0");
```

nests()

Measures enclosure distance from the outside of the contained polygon to the inside of the containing polygon.

See also: "dve_drc()" on page 4-1

Syntax

```
dve_drc (nests (inLayer1, inLayer2) operator distance
   [, msgString] [,qualifierName, qualifierValue...]);
```

where:

inLayer1	The contained polygon layer.
inLayer2	The containing polygon layer.
operator	

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

distance	A distance value in layout units.
msgString	A string value that will be attached to the selected error segments.
qualifierName, qualifierValue	A name, value pair that qualifies the selection.

Edge Qualifiers

Qualifier Resource Name: DVE_RN_EDGE_ANGLES

Qualifier Resource Value:

DVE_RV_PARALLEL	Select only parallel edges.
DVE_RV_NOT_PARALLEL	Select only non-parallel edges.
DVE_RV_PERPENDICULAR	Select only perpendicular edges.
DVE_RV_NOT_PERPENDICULAR	Select only non-perpendicular edges.
DVE_RV_ANY_ANGLE	(default) Select edges at any angle.

Qualifier Resource Name: DVE_RN_ANGLE_TOLERANCE

Qualifier Resource Value:

<real value>	Edge angle tolerance in degrees.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");
decl lyrError102 = dve_export_layer ("error102");

lyrError101 += dve_drc (nests (lyrCond, lyrCond2) < 4.0,
    "Clearance from contained to containing layers < 4.0");
lyrError102 += dve_drc (nests (lyrCond2, lyrCond) < 4.0,
    "Clearance from contained to containing layers < 4.0");
```

notch()

Measures the distance between outside edges of the same polygon on the given layer.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (notch (inLayer) operator distance [, msgString]
[,qualifierName, qualifierValue...]);

where:

inLayer A polygon layer.

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

distance A distance value in layout units.

msgString A string value that will be attached to the selected error segments.

qualifierName, qualifierValue A name, value pair that qualifies the selection.

Edge Qualifiers

Qualifier Resource Name: DVE_RN_EDGE_ANGLES

Qualifier Resource Value:

DVE_RV_PARALLEL	Select only parallel edges.
DVE_RV_NOT_PARALLEL	Select only non-parallel edges.
DVE_RV_PERPENDICULAR	Select only perpendicular edges.
DVE_RV_NOT_PERPENDICULAR	Select only non-perpendicular edges.
DVE_RV_ANY_ANGLE	(default) Select edges at any angle.

Qualifier Resource Name: DVE_RN_ANGLE_TOLERANCE

Qualifier Resource Value:

<real value>	Edge angle tolerance in degrees.

Example

```
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (notch (lyrCond2) < 15.0,
    "Outside edges same polygon < 15.0");
```

single_clearance()

Measures the distance between edges of a single polygon.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (single_clearance (inLayer) operator distance [, msgString]
[,qualifierName, qualifierValue...]);

where:

inLayer A polygon or edge layer.

operator

<	Less than
<=	Less than or equal to

==	Equal to
>	Greater than
>=	Greater than or equal to

distance A distance value in layout units.

msgString A string value that will be attached to the selected error segments.

qualifierName, qualifierValue A name, value pair that qualifies the selection.

Edge Qualifiers

Qualifier Resource Name: DVE_RN_EDGE_ANGLES

Qualifier Resource Value:

DVE_RV_PARALLEL	Select only parallel edges.
DVE_RV_NOT_PARALLEL	Select only non-parallel edges.
DVE_RV_PERPENDICULAR	Select only perpendicular edges.
DVE_RV_NOT_PERPENDICULAR	Select only non-perpendicular edges.
DVE_RV_ANY_ANGLE	(default) Select edges at any angle.

Qualifier Resource Name: DVE_RN_ANGLE_TOLERANCE

Qualifier Resource Value:

<real value> Edge angle tolerance in degrees.

Qualifier Resource Name:

DVE_RN_POLARITY
DVE_RN_POLARITY_FROM
DVE_RN_POLARITY_TO

Qualifier Resource Value:

DVE_RV_INSIDE	Direct search toward inside of polygon
DVE_RV_OUTSIDE (default)	Direct search toward outside of polygon

Qualifier Resource Name:

DVE_RN_TEMPLATE
DVE_RN_TEMPLATE_FROM
DVE_RN_TEMPLATE_TO

Qualifier Resource Value:

DVE_RV_ARC	Extend search area using arced corners.
DVE_RV_BOTHSIDES	Extend search area on both sides of edge.
DVE_RV_OPPOSITE	(Default) Extend search area just opposite the edge.
DVE_RV_ROUND	Extend search area using rounded corners.
DVE_RV_SQUARE	Extend search area treating corners as squares.

Qualifier Resource Name: DVE_RN_STRUCTURE

Qualifier Resource Value:

DVE_RV_ANY_POLYGON	(Default) Test applies to any edge.
DVE_RV_SAME_POLYGON	Test applies only between edges of same polygon.
DVE_RV_DIFF_POLYGON	Test applies only between edges of different polygons.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (single_clearance (lyrCond) < 3.0,
    "Parallel clearance < 3.0",
    DVE_RN_POLARITY, DVE_RV_OUTSIDE,
    DVE_RN_TEMPLATE, DVE_RV_OPPOSITE,
    DVE_RN_EDGE_ANGLES, DVE_RV_PARALLEL,
    DVE_RN_ANGLE_TOLERANCE, 1.2);
```

spacing()

Simultaneously measures the distance between outside edges of different polygons of the same layer (gap) and outside edges of the same polygon (notch).

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (spacing (inLayer) operator distance [, msgString]
[,qualifierName, qualifierValue...]);

where:

inLayer A polygon layer.

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

distance A distance value in layout units

msgString A string value that will be attached to the selected error segments

qualifierName, qualifierValue A name, value pair that qualifies the selection

Edge Qualifiers

Qualifier Resource Name: DVE_RN_EDGE_ANGLES

Qualifier Resource Value:

DVE_RV_PARALLEL	Select only parallel edges.
DVE_RV_NOT_PARALLEL	Select only non-parallel edges.
DVE_RV_PERPENDICULAR	Select only perpendicular edges.
DVE_RV_NOT_PERPENDICULAR	Select only non-perpendicular edges.
DVE_RV_ANY_ANGLE	(default) Select edges at any angle.

Qualifier Resource Name: DVE_RN_ANGLE_TOLERANCE

Qualifier Resource Value:

<real value> Edge angle tolerance in degrees.

Example

```
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (spacing (lyrCond2) < 15.0,
    "Gap and notch spacing < 15.0");
```

width()

A DRC clearance function to check from the inside of one edge of a polygon to the inside of another edge of the same polygon.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (width (inLayer) operator distance [, msgString] [, qualifierName, qualifierValue, ...]);

where:

inLayer A polygon layer.

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

distance A distance value in layout units

msgString A string value that will be attached to the selected error segments

qualifierName, qualifierValue A name, value pair that qualifies the selection

Edge Qualifiers

Qualifier Resource Name: DVE_RN_EDGE_ANGLES

Qualifier Resource Value:

DVE_RV_PARALLEL	Select only parallel edges.
DVE_RV_NOT_PARALLEL	Select only non-parallel edges.
DVE_RV_PERPENDICULAR	Select only perpendicular edges.
DVE_RV_NOT_PERPENDICULAR	Select only non-perpendicular edges.
DVE_RV_ANY_ANGLE	(default) Select edges at any angle.

Qualifier Resource Name: DVE_RN_ANGLE_TOLERANCE

Qualifier Resource Value:

<real value>	Edge angle tolerance in degrees.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (width (lyrCond) < 3.0,
    "Width of metal layer < 3.0");
```

Edge Selection By Select All or Inversion

Edge Selection By Select All or Inversion selection functions include: all_edges, invert_edges.

all_edges()

Sends all the edge segments of polygons of a layer to an output error layer.

See also: "dve_drc()" on page 4-1

Syntax

dve_drc (all_edges (inLayer) [, msgString]);

where:

inLayer	A polygon layer.
msgString	A string value that will be attached to the selected error segments

Example

```
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

decl lyrWork = NULL;

lyrWork = dve_drc (poly_area (lyrCond2) < 10.0);

lyrError101 += dve_drc (all_edges (lyrWork),
    "Conductive metal area < 10.0")
```

invert_edges()

Deselects selected edges and simultaneously selects unselected edges.

See also: "dve_drc()" on page 4-1

Syntax

dve_drc (invert_edges (inLayer) [, msgString]);

where:

inLayer	A layer eith selected edge segments.
msgString	A string value that will be attached to the selected error segments

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

decl lyrEdgesGap = NULL;
decl lyrEdgesInvert = NULL;
decl lyrEdges = NULL;
decl lyrPoly = NULL;

lyrEdgesGap = dve_drc (single_clearance (lyrCond) <= 3.0,
    DVE_RN_POLARITY, DVE_RV_OUTSIDE,
    DVE_RN_TEMPLATE, DVE_RV_OPPOSITE,
    DVE_RN_EDGE_ANGLES, DVE_RV_PARALLEL,
    DVE_RN_STRUCTURE, DVE_RV_DIFF_POLYGON,
    DVE_RN_ANGLE_TOLERANCE, 1.2);

lyrEdgesInvert = dve_drc (invert_edges (lyrEdgesGap));

lyrEdges = dve_drc (double_clearance (lyrEdgesGap, lyrEdgesInvert) < 3.0,
    DVE_RN_POLARITY, DVE_RV_INSIDE,
    DVE_RN_TEMPLATE, DVE_RV_OPPOSITE,
    DVE_RN_EDGE_ANGLES, DVE_RV_PARALLEL,
    DVE_RN_ANGLE_TOLERANCE, 1.2, "par");

lyrPoly = dve_quadout (lyrEdges);

lyrError101 = dve_drc (all_edges (lyrPoly),
    "Parallel interconnect < 3.0");
```

Edge Selection Based on Corners

Edge Selection Based on Corners selection function includes: corner_edges.

corner_edges()

Generates error segments around corners of specified angles.

See also: "dve_drc()" on page 4-1

Syntax

dve_drc (corner_edges (inLayer, segmentLength, beginningAngle,

endingAngle) [,msgString]);

where:

inLayer A polygon layer.

segmentLength A real value in layout units that represents the length of the error segment that will be drawn around the corner.

beginningAngle A real value that represents the minimum angle that will be selected

endingAngle A real value that represents the max angle that will be selected

msgString A string value that will be attached to the selected error segments

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");
decl lyrEdgesCvex = NULL;
decl lyrEdgesStub = NULL;
decl lyrStub = NULL;
lyrEdgesCvex = dve_drc (corner_edges (lyrCond, 0.5, 1, 91));
lyrEdgesStub = dve_drc (single_clearance (lyrEdgesCvex) < 3.0,
    DVE_RN_POLARITY, DVE_RV_INSIDE,
    DVE_RN_TEMPLATE, DVE_RV_OPPOSITE,
    DVE_RN_EDGE_ANGLES, DVE_RV_PARALLEL,
    DVE_RN_STRUCTURE, DVE_RV_SAME_POLYGON);
lyrStub = dve_quadout (lyrEdgesStub);
lyrError101 += dve_drc (all_edges (lyrStub), "Stub");
```

Edge Selection Based on Grid

Edge Selection Based on Grid selection function includes: off_grid.

off_grid()

Flags edges whose end points fall off a specified grid.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (off_grid (inLayer, grid) [,msgString]);

where:

inLayer A polygon layer.

grid A specified grid.

msgString A string value that will be attached to the selected error segments

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

lyrError101 += dve_drc (off_grid (lyrCond, 0.5),
    "Conductive metal is off grid");
```

Edge Selection Based on Grid 4-21

Edge Compensation

Edge Compensation selection function includes: compensate.

compensate()

Moves error segments on a given layer by a given distance. Output layer can only be used as input to dve_quadout and dve_plgout commands. Returns: A layer with selected edge segments.

See also: “dve_quadout()” on page 5-1, “dve_plgout()” on page 5-1

Syntax

edgeLayerOut = dve_compensate (edgeLayerIn, distance [,resourceName, resourceValue]);

where:

edgeLayerIn, edgeLayerOut	An edge layer.
distance	A real value.

Compensate Template Qualifier

Qualifier Resource Name: DVE_RN_COMP_TEMPLATE

Qualifier Resource Value

DVE_RV_CHAMFER	Compensate using an angle from the orthogonal.
DVE_RV_ALIGN	(Default) Compensate using an alignment to the adjacent edge.
DVE_RV_BISECT	Compensate where the angle is bisected at the corner.
DVE_RV_OPPOSITE	Compensate directly opposite the edge.

Qualifier Resource Name: DVE_RN_CHAMFER_ANGLE

Qualifier Resource Value

<real value>	Edge angle offset in degrees.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");
decl lyrEdges = NULL;
decl lyrEdgesComp = NULL;
decl lyrPolyCond = NULL;
decl lyrPolyComp = NULL;
decl lyrPolyOversize = NULL;

// Generate an oversized polygon

lyrEdges = dve_drc (width (lyrCond) < 5.0);
lyrEdgesComp = dve_drc (compensate (lyrEdges, 0.5),
                DVE_RN_COMP_TEMPLATE, DVE_RV_CHAMFER,
                DVE_RN_CHAMFER_ANGLE, 45);
lyrPolyCond = dve_quadout (lyrEdges);
lyrPolyComp = dve_quadout (lyrEdgesComp);
lyrPolyOversize = dve_bool_or (lyrPolyCond, lyrPolyComp);

// Check gap clearance
lyrError101 += dve_drc (gap (lyrPolyOversize) < 4.0, "Gap clearance < 4.0");
```

Polygon Selection Based on Intrinsic Properties

Polygon Selection Based on Intrinsic Properties (output layer contains polygons) selection functions include: poly_area, poly_hole_count, poly_line_length, poly_perimeter.

poly_area()

Selects polygons based upon area. Returns: A polygon layer.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (poly_area (inLayer) operator value);

where:

inLayer A polygon layer.

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

value An real value in layout units.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

decl lyrPoly = NULL;

lyrPoly = dve_drc (poly_area (lyrCond) < 15.0);

lyrError101 += dve_drc (all_edges (lyrPoly), "Polygon area < 15.0");
```

poly_hole_count()

Selects polygons based upon the number of holes. Returns: A polygon layer.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (poly_hole_count (inLayer) operator numHoles);

where:

inLayer	A polygon layer.
operator	

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

numHoles	An integer value representing the number of holes

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

decl lyrPoly = NULL;
decl lyrPolyHole = NULL;

lyrPoly = dve_bool_not (lyrCond, lyrCond2);

lyrPolyHole = dve_drc (poly_hole_count (lyrPoly) >= 1);

lyrError101 += dve_drc (all_edges (lyrPolyHole), "Polygon contains holes");
```

poly_line_length()

Selects polygons based upon the minimum line length. Returns: A polygon layer.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (poly_line_length (inLayer) operator distance [, qualifierName,
    qualifierValue]);

where:

inLayer A polygon layer.

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

value An real value in layout units

qualifierName, qualifierValue A name-value pair that qualifies the rule

Line Length Resource Qualifiers

Qualifier Resource Name: DVE_RN_LINE_LENGTH

Qualifier Resource Value

DVE_RV_MIN_LINE (Default) Select based upon minimum line length of polygon.

DVE_RV_MAX_LINE Select based upon maximum line length of polygon.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

decl lyrPoly = NULL;

lyrPoly = dve_drc (poly_line_length (lyrCond) <= 10.0,
    DVE_RN_LINE_LENGTH, DVE_RV_MAX_LINE);

lyrError101 += dve_drc (all_edges (lyrPoly), "Polygon length < 10.0");
```

poly_perimeter()

Selects polygons based upon the total length of the outside edges. Returns: A polygon layer.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (poly_perimeter (inLayer) operator distance);

where:

inLayer A polygon layer.

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

distance A real value in layout units.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

decl lyrPoly = NULL;

lyrPoly = dve_drc (poly_perimeter (lyrCond) < 20.0);

lyrError101 += dve_drc (all_edges (lyrPoly), "Polygon perimeter < 20.0");
```

Polygon Selection Based on Merge Properties

Polygon Selection Based on Merge Properties selection functions include: poly_edge_code, poly_path_count, and poly_path_length.

Polygon merge qualifier commands constrain the selection of edges based upon a specified edge code. All of the commands in this section are based upon edge information computed during a merge operation.

When polygon TOP and polygon BOTTOM merges, a set of vertices (shown as '*') consisting of the intersection points is derived. Each resultant edge between pairs of these vertices has a unique 'edge_code' that describe its relationship to other edges.

```
TTTTTTTTTTTTTT  polygon TOP
T            T
*bbbbbbbbbbbb*BBBBBBBBBBB   polygon BOTTOM
I            t          B
I            t          B
*bbbbbbbbbbbb*BBBBBB    B
T            T     B    B
T            T     B    B
T            *BBBBBB    B
T            E          B
T            E          B
T            *BBBBBBBBBBB
T            T
TTTTTTTTTTTTTT
```

where:

TOP_OUTSIDE_BOTTOM (T) is the polygon TOP outside polygon BOTTOM.

BOTTOM_OUTSIDE_TOP (B) is the polygon BOT outside polygon TOP.

TOP_INSIDE_BOTTOM (t) is the polygon TOP inside polygon BOTTOM.

BOTTOM_INSIDE_TOP (b) is the polygon BOT inside polygon BOTTOM.

INTERNAL (I) is the edges of TOP and BOTTOM butting internally.

EXTERNAL (E) is the edges of TOP and BOTTOM butting externally.

Path Code Qualifiers

Qualifier Resource Name: DVE_RN_PATH_CODE

Qualifier Resource Value:

DVE_RV_TOP	(Default) Select edges on top that are outside bottom.
DVE_RV_BOT	Select edges on bottom that are outside top.
DVE_RV_TIB	Select edges on top that are inside bottom.
DVE_RV_BIT	Select edges on bottom that are inside top.
DVE_RV_INT	Select edges on top and bottom that are butting internally.
DVE_RV_EXT	Select edges on top and bottom that are butting externally.

poly_edge_code()

Select polygons based upon edge code information computed during a merge operation. Select only polygons with have all the given path types. Input layer must be the result of a merge command. Returns: A polygon layer.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (poly_edge_code (inLayer) [,qualifierName,qualifierValue]);

where:

inLayer A polygon layer produced by a merge operation between two layers.

qualifierName, qualifierValue A name, value pair that qualifies the selection.

Selection Qualifier

Qualifier Resource Name: DVE_RN_SELECT

Qualifier Resource Value

DVE_RV_ACCEPT_ANY	Select polygon if any path codes are found.
DVE_RV_ACCEPT_ALL	(default) Select polygon if all path codes are found.
DVE_RV_REJECT_ANY	Reject polygon if any one of the path codes are found.
DVE_RV_REJECT_ALL	Reject polygon if exactly all the path codes are found.

Edge Code Qualifiers

Qualifier Resource Name: DVE_RN_PATH_CODE

Qualifier Resource Value:

DVE_RV_TOP	(Default) Select edges on top that are outside bottom.

DVE_RV_BOT	Select edges on bottom that are outside top.
DVE_RV_TIB	Select edges on top that are inside bottom.
DVE_RV_BIT	Select edges on bottom that are inside top.
DVE_RV_INT	Select edges on top and bottom that are butting internally.
DVE_RV_EXT	Select edges on top and bottom that are butting externally.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

decl lyrPolyCombine = NULL;
decl lyrPolyOverlap = NULL;
decl lyrPolyMerge = NULL;
decl lyrPoly = NULL;

lyrPolyCombine = dve_bool_or (lyrCond, lyrCond2);
lyrPolyOverlap = dve_bool_and (lyrCond, lyrCond2);
lyrPolyMerge = dve_bool_and (lyrPolyCombine, lyrPolyOverlap);
lyrPoly = dve_drc (poly_edge_code (lyrPolyMerge),
    DVE_RN_EDGE_SELECT, DVE_RV_ACCEPT_ANY,
    DVE_RN_PATH_CODE, DVE_RV_INT);

lyrError101 += dve_drc (all_edges (lyrPoly), "Conductive metal overlaps");
```

poly_path_count()

Select polygons based upon path count information computed during a merge operation. Input layer must be the result of a merge command. Returns: A polygon layer.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (poly_path_count (inLayer) operator distance [, qualifierName, qualifierValue]);

where:

inLayer	A polygon layer produced by a merge operation between two layers.
operator	

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

value An real value in layout units.

qualifierName, qualifierValue A name-value pair that qualifies the rule.

Path Count Qualifier

Qualifier Resource Name: DVE_RN_PATH_COUNT

Qualifier Resource Value:

DVE_RV_PATH_COUNT (Default) Select based upon path count of top polygon.

DVE_RV_ANTI_PATH_COUNT Select based upon path count of bottom polygon.

Path Code Qualifiers

Qualifier Resource Name: DVE_RN_PATH_CODE

Qualifier Resource Value:

DVE_RV_TOP (Default) Select edges on top that are outside bottom.

DVE_RV_BOT Select edges on bottom that are outside top.

DVE_RV_TIB Select edges on top that are inside bottom.

DVE_RV_BIT Select edges on bottom that are inside top.

DVE_RV_INT Select edges on top and bottom that are butting internally.

DVE_RV_EXT Select edges on top and bottom that are butting externally.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");
decl lyrPolyMerge = NULL;
decl lyrPoly = NULL;
lyrPolyMerge = dve_bool_not (lyrCond, lyrCond2);
lyrPoly = dve_drc (poly_path_count (lyrPolyMerge) >= 1,
    DVE_RN_PATH_CODE, DVE_RV_TOP,
    DVE_RN_PATH_CODE, DVE_RV_INT);
lyrError101 += dve_drc (all_edges (lyrPoly),
    "Metal layer outside and butting internally");
```

poly_path_length()

Select polygons based upon path length properties computed during a merge operation. Input layer must be the result of a merge command. Returns: A polygon layer.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (poly_path_length (inLayer) operator distance [qualifierName, qualiferValue]);

where:

inLayer A polygon layer produced by a merge operation between two layers

operator

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

value An real value in layout units

qualifierName, qualifierValue A name-value pair that qualifies the rule

Path Type Qualifiers

Qualifier Resource Name: DVE_RN_PATH_LENGTH

Qualifier Resource Value

DVE_RV_MIN_PATH Select based upon minimum path length of top polygon.

DVE_RV_MAX_PATH Select based upon maximum path length of top polygon.

DVE_RV_TOTAL_PATH Select based upon total path length of top polygon.

DVE_RV_ MIN_ANTI_PATH Select based upon minimum path length of bottom polygon.

DVE_RV_MAX_ANTI_PATH Select based upon maximum path length of bottom polygon.

DVE_RV_TOTAL_ANTI_PATH Select based upon total path length of bottom polygon.

Path Code Qualifiers

Qualifier Resource Name: DVE_RN_PATH_CODE

Qualifier Resource Value:

DVE_RV_TOP (Default) Select edges on top that are outside bottom.

DVE_RV_BOT Select edges on bottom that are outside top.

DVE_RV_TIB Select edges on top that are inside bottom.

DVE_RV_BIT Select edges on bottom that are inside top.

DVE_RV_INT Select edges on top and bottom that are butting internally.

DVE_RV_EXT Select edges on top and bottom that are butting externally.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");
decl lyrPolyMerge = NULL;
decl lyrPoly = NULL;
lyrPolyMerge = dve_bool_not (lyrCond, lyrCond2);
lyrPoly = dve_drc (poly_path_length (lyrPolyMerge) < 20.0,
    DVE_RN_PATH_CODE, DVE_RV_TOP,
    DVE_RN_PATH_LENGTH, DVE_RV_MIN_PATH);
lyrError101 += dve_drc (all_edges (lyrPoly),
    "Polygon path length < 20.0");
```

Polygon Selection Based on Edge Relationships

Polygon Selection Based on Edge Relationships (output layer contains polygons) selection function includes: poly_inter_layer.

poly_inter_layer()

Select polygons on one layer (inLayer1) in relation to edges of polygons on another layer (inLayer2) if any of the given constrains are true. Returns a polygon layer.

See also: “dve_drc()” on page 4-1

Syntax

dve_drc (poly_inter_layer (inLayer1, inLayer2) [, qualifierName, qualifierValue]);

where:

inLayer1, inLayer2 A polygon layers.

qualifierName, qualifierValue A name, value pair that qualifies the selection.

Selection Qualifier

Qualifier Resource Name: DVE_RN_SELECT

Qualifier Resource Value

DVE_RV_ACCEPT_ANY Select polygon if any path codes are found.

DVE_RV_REJECT_ANY Reject polygon if any one of the path codes are found.

Poly Code Qualifiers

Qualifier Resource Name: DVE_RN_INTER_CODE

Qualifier Resource Value

DVE_RV_INSIDE_ONLY (Default) Top is completely inside bottom and does not touch the inside of bottom.

DVE_RV_INSIDE_TOUCH, DVE_RV_INSIDE The contained top does touch the inside of bottom (DVE_RV_INSIDE_ONLY or DVE_RV_INSIDE_TOUCH).

DVE_RV_OUTSIDE_ONLY Top is completely outside bottom and does not touch the outside of bottom.

DVE_RV_OUTSIDE_TOUCH, DVE_RV_OUTSIDE   Top does touch the outside of bottom (DVE_RV_OUTSIDE_ONLY or DVE_RV_OUTSIDE_TOUCH).

DVE_RV_CUT_ONLY   Top is partly inside bottom and partly outside bottom with no internal-butt with bottom, (that is, it does not touch the inside of bottom).

DVE_RV_CUT_TOUCH   Top is partly inside bottom and partly outside bottom and does internal-butt (touch) bottom (DVE_RV_CUT_ONLY or DVE_RV_CUT_TOUCH).

DVE_RV_CUT_ANY, DVE_RV_ENCLOSE_ONLY   Bottom is completely inside top, and does not touch the inside of top.

DVE_RV_ENCLOSE_TOUCH, DVE_RV_ENCLOSE   The contained bottom does touch the inside of top (DVE_RV_ENCLOSE_ONLY or DVE_RV_ENCLOSE_TOUCH).

DVE_RV_CUT   DVE_RV_CUT_ANY or DVE_RV_ENCLOSE

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

decl lyrPoly = NULL;

lyrPoly = dve_drc (poly_inter_layer (lyrCond, lyrCond2),
    DVE_RN_INTER_CODE, DVE_RV_OUTSIDE);

lyrError101 += dve_drc (all_edges (lyrPoly), "Conductive metal outside");

lyrPoly = dve_drc (poly_inter_layer (lyrCond, lyrCond2),
    DVE_RN_INTER_SELECT, DVE_RV_REJECT,
    DVE_RN_INTER_CODE, DVE_RV_OUTSIDE_TOUCH,
    DVE_RN_INTER_CODE, DVE_RV_INSIDE_TOUCH);

lyrError101 += dve_drc (all_edges (lyrPoly),
    "Conductive metal outside and inside touch");
```

Chapter 5: Operations for Polygon Extraction from Edges

This section describes the DRC command used for polyextraction from edges. These functions include: dve_plgout and dve_quadout.

dve_plgout()

Extracts entire polygons from selected edges. If any section of a polygon is in error, then the entire polygon is extracted. Returns: A polygon layer.

See also: “dve_quadout()” on page 5-1

Syntax

dve_plgout (edgeLayer);

where:

edgeLayer A layer containing selected edge segments. Edge segments are selected using the dve_drc command.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrError101 = dve_export_layer ("error101");

decl lyrEdges1 = NULL;
decl lyrEdges2 = NULL;
decl lyrEdges3 = NULL;
decl lyrPolyInterconnect = NULL;

//Identify sections of interconnect metal w/width >=2.0 and width <= 3.0

lyrEdges1 = dve_drc (width (lyrCond) < 2.0);
lyrEdges2 = dve_drc (invert_edges (lyrEdges1));
lyrEdges3 = dve_drc (width (lyrEdges2) < 3.0);
lyrPolyInterconnect = dve_plgout (lyrEdges3);

lyrError101 += dve_drc (all_edges (lyrPolyInterconnect),
    "Valid interconnect");
```

dve_quadout()

Extracts a quadrilateral from the selected error segments on the given layer. Returns: A polygon layer.

See also: “dve_plgout()” on page 5-1

Syntax

dve_quadout (edgeLayer);

where:

edgeLayer A layer containing selected edge segments. Edge segments are selected using the *dve_drc* command.

Example

```
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

decl lyrEdges = NULL;
decl lyrPoly = NULL;
decl lyrPolySmall = NULL;

lyrEdges = dve_drc (width (lyrCond2) < 3.0,
            DVE_RN_EDGE_ANGLES, DVE_RV_PARALLEL,
            DVE_RN_TEMPLATE, DVE_RV_OPPOSITE);

lyrPoly = dve_quadout (lyrEdges);

lyrPolySmall = dve_drc (poly_line_length (lyrPoly) < 4.0,
            DVE_RN_LINE_LENGTH, DVE_RV_MAX_LINE);

lyrError101 += dve_drc (all_edges (lyrPolySmall),
            "Conductive metal length less than 4.0");
```

Chapter 6: Merge Operations on Polygons

This section describes the DRC commands used for merge operations on polygons. These functions include: dve_bool_and, dve_bool_not, dve_bool_or, dve_combine, dve_merge, and dve_self_merge.

dve_bool_and()

Merges overlapping polygons on two given layers. Returns: A polygon layer.

Syntax

dve_bool_and (inLayer1, inLayer2);

where:

inLayer1, *inLayer2*	A polygon layer.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

decl lyrPoly = NULL;

lyrPoly = dve_bool_and (lyrCond, lyrCond2);

lyrError101 += dve_drc (all_edges (lyrPoly), "Conductive metal overlapping");
```

dve_bool_not()

Subtracts shapes in the second layer from shapes in the first layer. Returns: A polygon layer.

Syntax

dve_bool_not (inLayer1, inLayer2);

where:

inLayer1, *inLayer2*	A polygon layer.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");
decl lyrPoly = NULL;

lyrPoly = dve_bool_not (lyrCond, lyrCond2);

lyrError101 += dve_drc (all_edges (lyrPoly),
            "Conductive metal not overlapping");

lyrPoly = dve_bool_not (lyrCond2, lyrCond);

lyrError101 += dve_drc (all_edges (lyrPoly),
            "Conductive metal not overlapping");
```

dve_bool_or()

Merges overlapping shapes on a given layer. Returns: A polygon layer.

Syntax

outLayer = dve_bool_or (inLayer1 [, inLayer2]);

where:

inLayer1, *inLayer2*	A polygon layer.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

decl lyrPoly = NULL;

lyrPoly = dve_bool_or (lyrCond, lyrCond2);

lyrError101 += dve_drc (width (lyrPoly) < 3.0,
            "Conductive metal less than 3.0");
```

dve_combine()

Combines shapes on multiple layers into one layer without modifying the shapes. Results of a *combine* command can only be used in a *dve_drc* command. Returns: A polygon layer.

Syntax

dve_combine (inLayer1 [, inLayer2, . . ., inLayerN])

where:

inLayer1, *inLayer2*, *inLayerN* A polygon layer.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrDiel = dve_import_layer ("diel");
decl lyrError101 = dve_export_layer ("error101");

decl lyrPolyAll = NULL;

lyrPolyAll = dve_combine (lyrCond, lyrCond2, lyrDiel);

lyrError101 += dve_drc (single_clearance (lyrPolyAll) < 2.0,
                  "Metal less than 2.0",
                  DVE_RN_POLARITY, DVE_RV_INSIDE,
                  DVE_RN_TEMPLATE, DVE_RV_ROUND);
```

dve_merge()

Merge shapes on multiple layers into one layer. Returns: A polygon layer.

Syntax

dve_merge (inLayer1 [, inLayer2, . . ., inLayerN])

where:

inLayer1, *inLayer2*, *inLayerN* A polygon layer that is not the result of a dve_merge.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrDiel = dve_import_layer ("diel");
decl lyrError101 = dve_export_layer ("error101");

decl lyrMerge = NULL;

lyrMerge = dve_merge (lyrCond, lyrCond2, lyrDiel);

lyrError101 += dve_drc (corner_edges (lyrMerge, 1.5, 90.5, 360.0),
                "Concave corner edges");
```

dve_self_merge()

Merge shapes on multiple layers into one layer, selecting only polygons on the first that come from the same original merge group as the second layer. Returns: A polygon layer.

Syntax

dve_self_merge (inLayer1, inLayer2)

where:

inLayer1, inLayer2	A polygon layer.

Example

```
decl lyrCond = dve_import_layer ("cond");
decl lyrCond2 = dve_import_layer ("cond2");
decl lyrError101 = dve_export_layer ("error101");

decl lyrPolyMerge = NULL;
decl lyrPoly1 = NULL;

lyrPolyMerge = dve_self_merge (lyrCond2, lyrCond);

lyrPoly1 = dve_drc (poly_edge_code (lyrPolyMerge),
            DVE_RN_EDGE_SELECT, DVE_RV_ACCEPT_ALL,
            DVE_RN_PATH_CODE, DVE_RV_TOP);

lyrError101 += dve_drc (all_edges (lyrPoly1),
            "Conductive metal outside");

lyrPoly1 = dve_drc (poly_edge_code (lyrPolyMerge),
            DVE_RN_EDGE_SELECT, DVE_RV_ACCEPT_ALL,
            DVE_RN_PATH_CODE, DVE_RV_TIB);

lyrError101 += dve_drc (all_edges (lyrPoly1),
            "Conductive metal inside");
```

Chapter 7: Troubleshooting

If a dve_drc command is not producing the expected output, try the following debugging techniques:

- Resolve any compile errors or warnings.
- Check to make sure the dve_drc command has an error message.
- If possible, always use < for clearance rules to ensure a bounded check.
- Inspect the input layers using the layer editor. Send the data to an export layer (be sure to include an error message), and view the data using *Load Results*.

Layer Management Errors (101-199)

101 Import layer must be a design layer

Import and export layers must be defined as physical design layers

102 Export layer must be a design layer

Import and export layers must be defined as physical design layers

103 No output layer

An output layer is required on the left-hand side of the equal sign (=).

104 Layer parameter is uninitialized

Input layers must have previously appeared on the left-hand side of an equal sign (=).

105 Layer parameter is an export layer

An input layer that has been declared as an export layer cannot be used as an input layer.

106 No import layers defined

At least one input layer must be defined.

107 No export layers defined

At least one export layer must be defined.

108 Rules do not generate output

At least one rule must assign data to an export layer.

Layer Management Warnings (201-299)

201 Redefining an import layer

A layer that has been declared as an import layer is being redefined.

202 Redefining an export layer

A layer that has been declared as an export layer is being redefined.

Command Usage Errors (301-399)

301 Expecting layer parameter

Parameter is uninitialized or is not a layer. Please see documented command syntax.

302 Expecting a string parameter

Parameter is uninitialized or is not a string. Please see documented command syntax.

303 Expecting an integer parameter

Parameter is uninitialized or is not an integer number. Please see documented command syntax.

304 Expecting a real parameter

Parameter is uninitialized or is not a real number. Please see documented command syntax.

305 Invalid angle parameter

Expecting a real number greater than 0 and less than 360 with only one decimal point of precision.

306 Command is a dve_drc subfunction

Command must appear as the first parameter to a dve_drc subfunction. Command is not valid outside the context of a dve_drc command.

307 Unsupported operator

The dve_drc expression contains an unrecognized operator. Valid operators are

<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

308 Unsupported set operator

The command is missing the left-hand equal sign for assignment to an output layer.

309 Missing elements of expression

The command requires an expression. Please see the documented command syntax.

310 Expecting polygon layer

Polygon layers are produced as the result of polygon selection or boolean commands. Edge operation commands perform segment merging, sizing and polygon extraction on selected edges.

311 Expecting edge layer

Edge layers are produced as the result of an edge selection, edge compensation, or edge operation command. Polygon commands perform polygon selection and boolean operations on polygons.

312 Expecting boolean merge layer

Polygon selection commands based on merge properties only accept input layers that are the direct result of a boolean polygon merge operation such as dve_bool_and.

313 Expecting dve_drc subfunction

The dve_drc command must always appear with a dve_drc subfunction as the first parameter.

314 Nested merge not allowed

The result of a dve_merge command cannot be used as the input to another dve_merge command.

Command Usage Warnings (401-499)

401 Expression ignored

Command does not require an expression

402 Qualifier ignored

Resource qualifiers that do not apply are ignored. Please see documented command syntax.

403 Using default polarity

A polarity specification is required for commands double_clearance and single_clearance. If no polarity is specified, a polarity DVE_RV_OUTSIDE is used.

404 Using default template

A template specification is required for commands double_clearance and single_clearance. If no template is specified, a template DVE_RV_OPPOSITE is used.

405 Clearance qualifiers ignored

Clearance qualifiers require an upper bound and are currently not supported for unbounded greater-than (>) or greater-than-or-equal (>=). This can be corrected by using range comparisons.

Index