Guardian DRC – Recent Development

Edge-Based Operations

Recent versions of Guardian DRC introduce operations that involve edge layers, either on input or on output.

An edge layer is a layer that contains line segments that are edges or parts of edges of shapes from other layers. These edges have "inner" and "outer" surfaces defined according to the shape layers they were derived from.

Edge Selection by Topological Relations

```
select_edges:  relation={inside | outside}
               [, options=(not)],
               layer1=LayerA,Layer2=LayerB,
                  LayerR=EL1;
```

These commands (without "Not" option) select edges or parts of edges from Layer1 that lay strictly inside (respectively, outside) of shapes from Layer2.

"Not" option is for inverting the selection.

```
select_edges: relation = {coincide | touch}
  [, options=(not[,{inside | outside}])],
  layer1=LayerA, Layer2=LayerB,LayerR=EL1;
```

"Coincide" relation selects parts of edges from Layer1 that coincide with (parts of) edges from Layer2.

"Touch" relation selects complete edges whose parts coincide with (parts of) edges from Layer2.

"Inside" option means that at points of coincidence the insides of edges overlap.

"Outside" option means that at points of coincidence the insides of edges do not overlap.

"Not" option is for inverting the selection.

Shape Selection by Edges

```
select: relation=touch,
  [, options=(options_list)],
        layer1=LayerA, Layer2=LayerB,
  LayerR=EL1;
```

Layer2 is allowed to be edge layer. In this case, the operation selects shapes from Layer1 whose edges touch edges from layer2. In this case "Touching" can be both inside and outside touching. (When both layers are shape layers, touching is considered only from the outside.)

Edge Layers on Input of Old DRC Operations

- All distance check operations accept edge layers on input
- Substrate (new syntax allows input layers)
- Copy (if input is edge layer, then LayerR is edge layer)
- Delete
- Select...Touch may have Layer2 to be edge layer.

Size_Edge Operation

```
Size_edge: layer = <L1>,
[{In | In_Factor} = <NIn>,] [{Out | Out_
  Factor} = <NOut>,]
[{Extend | Extend_factor} =<NExt>,]
layerr = <L2>;
```

"*_Factor" parameters mean that the numeric parameter is the factor by which the edge length must be multiplied to obtain the actual sizing parameter.

"In" parameter specifies the amount of expanding the edge in the "inside" direction. This parameter must be nonnegative.

"Out" parameter specifies the amount of expanding the edge in the "outside" direction. This parameter must be nonnegative.

"Extend" parameter specifies the amount of elongating the edge from both sides. This parameter may be negative, meaning edge contraction.

An edge produces no output, if its length does not exceed twice the negated extension value (e.g., in the case "Extend_factor = -0.5").

NOTE: If all in/in_factor/out/out_factor parameters are zero, the output is an edge layer, otherwise it is a shape layer.

Edge Selection by Slope

The following command selects edges with slopes within specified limits.

The limits are specified in degrees, with values between 0 and 90.

Slope: Layer=<name>, LayerR=<name>, Limits <range>;

The output is an edge layer. The input layer may be shape or edge layer.

Example:

Slope: layer= M1, layerR=M1Acute, limits >0 <=45;

Edge Selection by Adjacent Corners

Select_Edges: Relation=Corners,

[convex=<012-limits>,] [length <limits>,]

[angle1 <0.0-360.0>,][Length1 <limits>,]

[angle2 <0.0-360.0>,][Length2 <limits>,]

layer=<shape-layer>, layerR=<edge-layer>;

This command selects edges basing on the parameters of the adjacent corners: angles and side lengths.

convex specifies limits for the number of convex angles adjacent to the edge;

length specifies limits for the length of the segment itself

angle1,2 specify limits for the first/second adjacent angle, in range between 0 and 360 degrees.

length1, 2 specify limits for length of the first/second adjacent side.

Examples:

Select_Edges: Relation=Corners, convex <2, length >10, layer=m1, layerr = outbendM1;

Select_Edges: Relation=Corners, convex <2, angle1 = 90, angle2 == 90, length <0.3, layer=m1, layerr = m1Loop;

News for Antenna Checks

Notion of Layer of Origin

If LayerB is produced from LayerA by the following operations:

- Select operations with merged selection layer
- Select_edge operations
- Check operations with different LayerR1 and LayerR2 values
- Check_Node_Params
- Copy

then LayerA is called layer of origin for layerB.

Antenna Checks with Accumulation

'LayerA' keyword is introduced for Check_Node_Params command:

Check_Node_Params: Formula = (expression),

Value=<value1[:value2]>, Type=<check type>,

Layer = <layer identifier>, LayerR = <output layer identifier>,

LayerA = <accumulative layer identifier>;

LayerA must be the output layer of a previously executed Check_Node_Params operation. LayerA and Layer must have the same layer of origin.

If LayerA is not specified, then Check_Node_Params command attaches to each output polygon the corresponding value of computed Formula.

If LayerA is specified, then for each polygon from Layer (which belongs to an electrical node present in input parameter files) the following is done:

- If the polygon is not present in LayerA, then the calculated value of Formula itself is checked against the constraints.
- If the polygon is also present in LayerA, then the calculated value of Formula for its node is increased by the value attached (by the preceding Check_Node_Params operation) to the polygon in LayerA, and the result is checked against the specified constraints.

If the checked accumulated value meets the constraints, then the polygon is output to LayerR with the accumulated value attached.

Note: If LayerR is written into Expert layout database, the attached values are represented as used-defined properties with name specified in "Update_layout" command, parameter "Antenna".

Antenna Log Files

The 'LogR' and 'LogA' keywords are introduced for Check_Node_Params command:

Check_Node_Params: Formula = (expression),

Value=<value1[:value2]>, Type=<check type>,

Layer = <layer identifier>, LayerR = <output layer identifier>

[, LayerA = <accumulative layer identifier>]

[, LogR = <output geometry log-file name>],

[, LogA = <antenna violation log-file name>];

'LogR' keyword allows to generate a log-file containing the following information for each polygon in the output layer:

1) lower left vertex of the polygon (coordinates are in internal database units)

2) (accumulated) antenna value corresponding to the polygon (the value are in script measurement units).

The log-file also contains information about the operation performed, the internal database unit and the script measurement unit.

If file with specified name exists, the operation rewrites it.

'LogA' keyword allows to generate a log-file containing the following information for each electrical node failed the antenna check:

1) node number

2) computed formula value corresponding to the node (the values are in script measurement units).

The accumulation of antenna values (provided by 'LayerA') are not reflected in this log-file.

The log-file also contains information about the operation performed and the script measurement unit.

If file with specified name exists, the operation appends it.

'Shapes' Parameter

A new geometric parameter name, 'Shapes', can be used in Get_Node_Params and Check_Node_Params commands. It denotes the number of polygons that belong to a node. For this node parameter, all actions (Min, Max, Sum) are equivalent, so you can always use this parameter without specifying the action.

"Not" Operation

Unary ! (Not) mathematical operation can be used in Formula in Check_Node_Params command. The result of the operation is zero if its argument is non-zero and 1 if its argument is zero.

Preprocessor Directives

In addition to #ifdef and #ifndef, a new directive, #if is added.

Syntax:

```
#if <Boolean_expression>
//...
[#else]
//...
#endif
```

Boolean_expression can consist of preprocessor variables and constants, parentheses, Boolean operations (NOT, AND, OR, XOR, EQV (case insensitive)), and comparisons of expressions (==, != <=, >=, <,>).

Constant's Boolean value is "true" if it is nonzero and "false" otherwise.

Preprocessor variable's Boolean value is defined as before.

Recall that a preprocessor directive must be written in a single line.

Example:

```
#define A
#define B 1
#undef B

// at this point a=true, b=c=d=false
#if (A or B or C or D)
//...
#else
//...
#define C 12
// now C is true
#if (((C >= 1) and NOT A) or D)
//...
#endif
//...
#endif
//...
```

Input/Output of Layout Data

Input Merging

Merge_Input: {on | off};

If this setting is on, then all input layers are merged when layout data is read from them into the DRC system.

The default setting is off, according to the old behavior of the DRC engine.

Input Snapping to Grid

Snap_Input: {on | off};

If this setting is on, then all input layers are snapped onto the corresponding grids when layout data is read from them into the DRC system. If Merge_Input is on, input snapping is performed before input merging.

Grid setting is specified by Grid_Resolution directive.

The default grid setting is 1 Database unit.

The default setting is off, according to the old behavior of the DRC engine.

Returning Layers from DRC to Expert Layout Database

The following setup command controls writing layers from DRC layout data into ELD files.

Update_Layout:

```
[  input=yes|no]
[, new=yes|no]
[, ampersand="string"]
[, technology=Yes|No|Combine]
[, nonempty=  Yes|No|Combine]
[, layers=(<list>)]
[, no_layers=(<list>)]
[, combine_layers=(<list>)]
[, warning=Yes|No|Error];
[, Antenna_Attr_Name = <text>]
[, Node_Attr_Name = <text>…;]
[, output_cell= "name"]
```

No : do not write into ELD file.

Yes : overwrite existing data.

Merge: merge with existing data.

Input: Layers from "Layers" list

New: New (non-temporary) layers

Ampersand="string": Write back temporary layers by name, with '&' replaced by "string" (1st '&' is forbiddedn in Expert)

Technology: Non-scratch layers from ELD technology (by name)

Nonempty: Any nonempty ELD layers

Layers=(list): list of layers allowed to be written back

(overrides settings of switches)

Merge_Layers=(list): list of layers allowed to be merged by writing back

(overrides settings of switches)

No_Layers=(list): list of layers forbidden to be written back

(overrides settings of switches)

Warning=Yes|No|Error : if a DRC command outputs a layer that confilicts with the settings of the command, then write/nowrite warning into log or stop execution.

Default setting (chosen to match the old behavior of DRC):

If this command is not specified then input=no, all remaining switches =yes

If several switches apply to a layer (e.g. a layer is input and nonempty), then option "NO" takes precedence (for data safety).

Conflicts between "...Layers" lists are treated as error.

Passing Connectivity and Antenna Information into ELD Format

If a layer has node info and/or antenna info, then the corresponding shapes written into eld file will have user-defined attributes with the names specified in the Update_layout command and the corresponding values.

Antenna_Attr_Name specifies the name of a user-defined attribute attached to a shape when it is returned from DRC to Expert. The value of this attribute is the value calculated by Check_Node_Params operation.

Node_Attr_Name specifies the name of a used-defined attribute attached to a shape when it is returned from DRC to ***Expert***. The value of this attribute is the node name for the shape.

Cell for DRC-Generated Layers

output_cell= "name"

This parameter specifies the name of the cell to write the DRC-generated layers instead of the cell for which DRC was run.

If there is no such cell in the layout, it will be created.

Numeric Parameters of DRC Commands

This version allows you to write flexible checks using arithmetic expressions to specify check constraints, as shown in the example below.

variable: Gap = 0.3;

size: layer= AVDF, value = Gap/100, layerr= AVs;

merge: layer = AVs, layerr = AVm;

Outdistance: layer = Avm, limits >=Gap/100 < Gap*1.5, ID= "AVD3.1";

Check and Selector Limits

A simpler syntax for check constraints is introduced to replace the likes of "type=LT, value=0.3".

```
Limits <limit>,

Limits <range>

where <limit> is one of :

!= 'value'

== 'value'

 = 'value' (the same as ==<value>)

 > 'value'

 < 'value'

>= 'value'

<= 'value'
```

'value' cannot have measurement unit suffix. You must use "unit" statement, if necessary.

<range> is <limit1> <limit2>, where one of limits is the lower limit of the range and another one is the upper limit. The lower one must not exceed the upper one.

Examples

(1) width: layer=m1, limits !=1;

(2) width: layer=m1, limits <=2>=1;

(3) width: layer=m1, limits >1 <=1.5; // run time error: the upper limit less than the lower one

(4) width: layer=m1, limits >=1 <=1; // equivalent to (1)

(5) width: layer=m1, limits >1<1; // run time error: bad range

A similar usage is possible for count limits in select operations.

Examples

Select: relation= overlap, options=(shapes !=2),...;

Select: relation= overlap, options=(nodes >1 <=4),...;

Variables

Variables are components of arithmetic expressions.

variable: <name> = <value>;

<name> is the name of the variable. It is case-insensitive.

<value> is an arithmetic expression built from numbers and previously defined variables.

Examples

```
VARIABLE: MinW  = 0.2;

VARIABLE: MinS =0.301;

VARIABLE: Unders=MinW/2 - 0.001;

VARIABLE: Step = 2*Unders + MinS;
```

Arithmetic Expressions

Arithmetic expressions are constructed from numbers and variables, parentheses, arithmetic operators (+,-,*,/) and functions:

MIN(expr, expr), MAX(expr, expr) minimum and maximum of the two expressions

INT(expr) integer part of the numer

SQRT(expr), square root

POWER(expr1, expr2) expr1 raised into expr2 power.

Arithmetic expressions may be used in the following places:

- Variable definitions

- With "Limits" keyword in checks

- With "Value" keyword in sizing operations

- With "shapes" and "Nodes" keywords in select options.

Substrate Command: New Options and Parameters

substrate: [layer = <LName>,]

[T=<Ymax>,][B=<Ymin>,][L=<Xmin>,][R=<Xmax>,] layerr= <RLName>;

substrate: [layers = (Lnam1, ..., LnamN),]

[T=<Ymax>,][B=<Ymin>,][L=<Xmin>,][R=<Xmax>,] layerr= <RLName>;

substrate: [options=([Nosize][,][Input]),]

[T=<Ymax>,][B=<Ymin>,][L=<Xmin>,][R=<Xmax>,] layerr= <RLName>;

T, B, L, R parameters specify the top, bottom, left and right clipping coordinates for the produced box. In particular, the commands

variable: BoxSize = 0.1;

substrate: b=1, t=1+BoxSize, l=3, r= 3+BoxSize, layerr=Box13;

build 'Box13' layer with the specified box, if it is inside the "normal" substrate.

"Input" option means that the substrate is based only on the layers actually taking part in DRC operations (regardless technology file or layer table).

Note:

Compatibility with Batch Calibre:

substrate: options=(input, nosize),...

Compatibility with Interactive Calibre:

substrate: options=(nosize),...

substrate: [layer = <LName>,] layerr= <RLName>;

substrate: [layers = (Lnam1, …, LnamN),] layerr= <RLName>;

substrate: [options=(Nosize),] layerr= <RLName>;

If Layer or Layers parameters are present, then the command builds the bounding box for the listed layers only.

If "Nosize" option is absent, this box is resized by 1um from the minimal bounding box. "Nosize" option cancels this resizing.

NOTE: In the first two forms of syntax there is no resizing.