Connectivity Operations in *Guardian* DRC

1. Introduction

The article "Connectivity options for DRC spacing check operations in ***Guardian***" (*Simulation Standard*, vol. 11, no. 12) [1] describes layout netlist connectivity based on DRC checks. Earlier versions of Silvaco ***Guardian*** required connectivity before netlist extraction by the ***Maverick*** extractor. ***Maverick*** often used the ***Guardian DRC*** engine in order to generate derived layers for device/connectivity definition. This required a complicated tool setup that consisted of two DRC scripts and an extraction-related technology setup section. Connectivity construction operations have been introduced directly into recent versions of ***Guardian***. This article describes their proper usage.

Note: If a DRC script requires connectivity-based checks but does not have the necessary connectivity extraction statements, it is still possible to use the older ***Maverick*** *netlist* extraction method.

2. Connectivity Zones

Guardian features the option of building, modifying, and reusing electrical connectivity information for use in connectivity related operations during DRC script execution. Previous connectivity information is easily discarded by starting a new **connectivity zone**. CONNECT_ORDER statements split the DRC script into connectivity zones. All other connectivity related operations use or modify connectivity only in the corresponding zone.

Syntax:

```
CONNECT_ORDER: [<comma-separated layer list>];
```

This statement starts a new connectivity zone and defines the connect sequence for the conducting layers from bottom to top. All previously built connectivity information is disregarded.

If the CONNECT_ORDER statement includes a non-empty layer list, the conducting layer shielding by intervening layers in all CONNECT statements is then based on the conducting layer sequence defined in the CONNECT_ORDER statement.

If CONNECT_ORDER statement is omitted or includes an empty layer list (CONNECT_ORDER: ;), the conducting layer shielding by intervening layers in each CONNECT statement is then based on the sequence of conducting layers in this particular statement.

Example:

```
CONNECT_ORDER: &BULK, &DIFF, POLY, M1;
```

3. Building Connectivity Information.

To build (modify) connectivity information within a connectivity zone, use the CONNECT and ATTACH_LABELS operations. CONNECT statements establish electrical connectivity between conducting layers. ATTACH_LABELS statements label electrical nodes with layout texts and virtually connect nodes by means of labels containing colons.

3.1. CONNECT Statement

Syntax:

Bidirectional Connect Statements

```
Connect: <Conducting Layer Spec>,
LayerC=<ContactLayer> [,Options =(I-)];
Connect: <Conducting Layer Spec>
[,Options =([I-][[,]T+])];
```

Directional Connect Statements

```
Connect: <Conducting Layer Spec>,
LayerC=<ContactLayer>, Options=(S[,I-]),
   [, LayerR=<OutputLayer>];
Connect: <Conducting Layer Spec>,
Options=(S[,I-][, T+]), [,LayerR=<OutputLayer>];
```

<Conducting Layer Spec> may be one of:

- Layer1=<ConductLayer1>, Layer2=<ConductLayer2>
- Layers = (Layer_List)

In the above example, Layer_List is a comma-separated list of layer names. Notice the use of parentheses.

LayerC specifies the contact layer. If no contact layer is defined, then all layers are connected pairwise at points of overlap. If LayerC is defined, the first conducting layer is connected to each of the remaining layers with the defined layer if the three overlap.

The actual connection occurs if no intervening layer is present. A layer is considered intervening for layers L1 and L2 if it lies between L1 and L2 in the nearest preceding CONNECT_ORDER statement, or if the latter exists and has a non-empty layer list. Otherwise, a layer is considered intervening if it lies between L1 and L2 in the conducting layer list of the CONNECT statement itself.

Example 1:

```
CONNECT_ORDER: PWELL, DIFF, M1;
CONNECT: LAYER1 = M1, LAYER2 = PWELL, LAYERC = CNT;
```

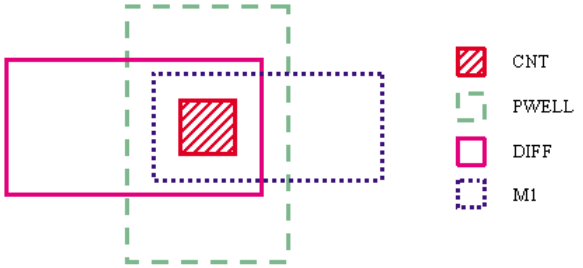

Figure 1. Connectivity Shielding by Intervening Layer

In Figure 1, the intervening layer DIFF prevents layers M1 and PWELL from connecting by means of contact layer CNT.

Option I- suppresses conducting layer shielding by the intervening layers. If this option is specified, a pair of conducting layers is connected, regardless of the geometry of any other conducting layer.

Option T+ is used only when the contact layer is not used in the statement (LayerC keyword is absent). If T+ is specified, connection occurs through both overlapping and the touching of shapes from conducting layers.

Option S sets a unidirectional connection. The node information is passed from the upper-layer to the lower-layer(s), but not in the reverse direction. The upper layer is first in the list of conducting layers. The LayerR is used with option S only. It then contains shapes from the lower layers connected to more than one node of the upper-layer (the node number to pass from upper-layer to lower-layer is chosen at random).

The following is a complete example for an antenna rule check that illustrates the usage of CONNECT_ORDER and CONNECT statements.

Example 2:

```
// Define input layers
LAYERS:
   POLY (5),
   CNT  (8),
   M1   (9),
   ACT (21);

// Define remaining basic layers
Substrate: LayerR=&BULK;
And: Layer1=ACT, Layer2=POLY, LayerR=&FETGATE;
Dif: Layer1=ACT, Layer2=&FETGATE, LayerR=&DIFF;
And: Layer1=&DIFF, Layer2=&BULK, LayerR=&DFBL;

CONNECT_ORDER: &BULK, &DIFF, POLY, M1;

// Define connections
CONNECT: layer1 = M1,    layer2 = POLY,  layerC = CNT;
CONNECT: layer1 = M1,    layer2 = &DIFF, layerC = CNT;
CONNECT: layer1 = &DIFF, layer2 = &BULK,  layerC = &DFBL;

// Transfer nodal data to gates
Select: Relation=STAMP, Options=(Touch-),
     Layer1=&FETGATE, Layer2=POLY,
     LayerR=&FETGATE_STM;

// Extract gate area per node
Get_Node_Params: Options=(AREA), Layer=&FETGATE_STM,
         LayerR=GTAREA;

// Extract M1 area per node
Get_Node_Params: Options=(AREA), Layer=M1,
         LayerR=M1AREA;

// Find gates with antenna ratio > 150
Check_Node_Params: Formula=(SUM.M1AREA.Area /
         SUM.GTAREA.Area),
         Layer=POLY, Type=GT, Value=150,
         LayerR=&ERPOLY;
Copy: Layer=&ERPOLY; // report as violations
```

3.2. ATTACH_LABELS Statement

Syntax:

```
ATTACH_LABELS: Layer = <LabelLayer> [, LayerC
        = <ConnectLayer>];

ATTACH_LABELS: Layers = (Layer_List)  [,
        LayerC = <ConnectLayer>];
```

<LabelLayer> is a layout layer that contains text used to name electrical nodes.

Layer_List is a comma-separated list of names of multiple layers that contains text used to name electrical nodes. Notice the use of parentheses.

<ConnectLayer> is a conducting layer. It must be present in the nearest preceding CONNECT_ORDER statement or in the current connectivity zone's LABEL_ORDER statement.

If the LayerC is specified, then labels from <LabelLayer> are assigned only to shapes of <ConnectLayer>. If the origin of a text is located over <ConnectLayer> geometry, then the text assumes the name of the electrical node containing this geometry. If the text's origin is outside <ConnectLayer> geometry, then the text is disregarded.

If the LayerC keyword is omitted, the labels from <LabelLayer> are first assigned to the rightmost conducting layer of CONNECT_ORDER statement's layer list. Texts assigned to this layer are excluded from further consideration. If some texts are not assigned to this layer, an attempt is made assign them to the next conducting layer (from right to the left in the CONNECT_ORDER statement). Texts that are not assigned to a connection layer are disregarded. If the LABEL_ORDER statement is present, the precedence of conducting layers is determined by this operator rather than by the CONNECT_ORDER statement.

Text labeling is used in DRC for:

- SELECT ... LABEL operation with "C" option.
- LINK operation
- Virtual connection of nodes by means of labels that contain colons. Nodes with identical names up to the first colon are considered connected even there is no real connection between them. The colon symbol at first position of a node name is considered a regular symbol.

4. Using Connectivity Information

A number of DRC operations requires the following electrical connectivity information:

- Geometric check operations with options C and C'.
- SELECT: relaiton=(CONNECT), ...;
- SELECT: options=(C), ...;
- SELECT: options=(Nodes=<n1>[:<n2>]), ...;
- SELECT: relation=STAMP, ...;
- LINK: .....; (This command assigns the same node number (the node number corresponding to the specified label) to all primitives of the specified layer),
- Get_Node_Params, Check_Node_Params (Antenna checks).

If this operation is performed before the required connectivity information is built, then a warning is issued and the operation is either skipped, or if possible, performed without the connectivity option

In addition, some operations use existing connectivity information, even though it's not required. Output layers of selection, Merge, And and Dif operations automatically inherit electrical node information from the first input layer.

In addition to the mainstream approach of building electrical connectivity information by means of the CONNECT_ORDER, CONNECT and ATTACH_LABEL commands, there is another way for obtaining connectivity information if DRC for the ***Expert*** layout database is run from the ***Expert's*** user interface. ***Expert*** generates connectivity information using ***Maverick*** Netlist Extractor. If a running DRC script does not contain operations that build connectivity information but contains commands that requiring this information, then the information generated by ***Maverick*** Netlist Extractor is used. If the operations are present, the ***Maverick*** information is ignored.

5. Compatibility with Other DRC Systems

Guardian DRC is versatile and compatible with products of other DRC vendors. The many connectivity related operation options allow a user to build and use electrical connectivity information in DRC scripts, similar to DRACULA™ or Calibre™. To achieve this, use these options consistently throughout the script.

The following are crucial compatibility points:

5.1. DRACULA™ Compatibility

- Always associate a CONNECT_ORDER statement with a non-empty conducting layers list, analogous to the CONNECT_LAYER command.
- In CONNECT statements, use <Conducting Layer Spec> in form of Layer1 = <ConductLayer1>, Layer2 = <ConductLayer2>
- Always use the LayerC keyword in CONNECT statements
- Do not use the I- and T+ options in CONNECT statements
- Use the CONNECT statement with S option as an analog of SCONNECT command
- Use the CONNECT statement with S option and the LayerR keyword as an analog of SOFTCHK command

5.2. Calibre™ Compatibility

- Do not use a CONNECT_ORDER statement with a non-empty conducting layers list
- In CONNECT statements, use <Conducting Layer Spec> in form of Layers = (Layer_List) to specify more than two conducting layers
- Always use I- and T+ options when omitting the LayerC keyword in CONNECT statement without the S option
- Do not use the I- and T+ options if the LayerC keyword is present in CONNECT statement without S option
- Use the T+ option in a CONNECT statement with an S option as an analog of the ABUT_ALSO secondary keyword in a SCONNECT operation
- Use a CONNECT statement with S option and LayerR keyword as an analog of LVS SOFTCHK command

6. Conclusion

Adding connectivity construction operations into ***Guardian DRC*** language simplifies execution and increases flexibility of connectivity-based checks. Future developments include incorporation device extraction means into ***Guardian DRC*** language.

References

[1] Connectivity options for DRC spacing check operations in Savage, *Simulation Standard*, Vol. 11 (2000), no.12, pp.1-2, 5.