# SPACE APPLICATION NOTE
# ABOUT NODEPOINTS

*S. de Graaf*

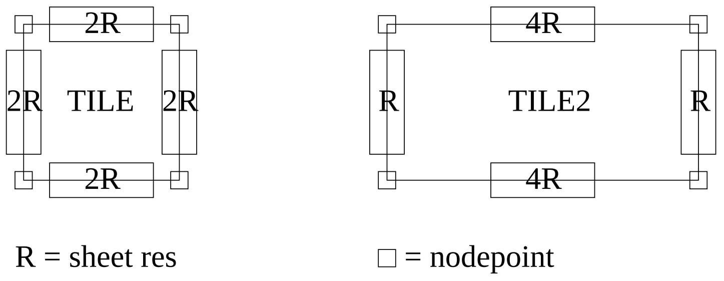
Circuits and Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
Delft University of Technology
The Netherlands

Report EWI-ENS 03-08
October 29, 2003

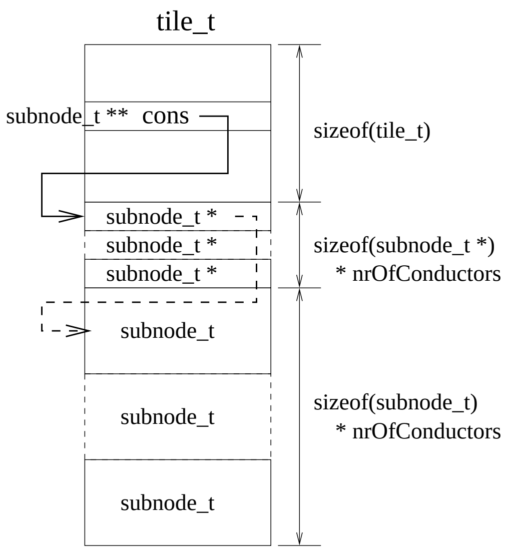
Last revision: August 16, 2004.

## 1. INTRODUCTION

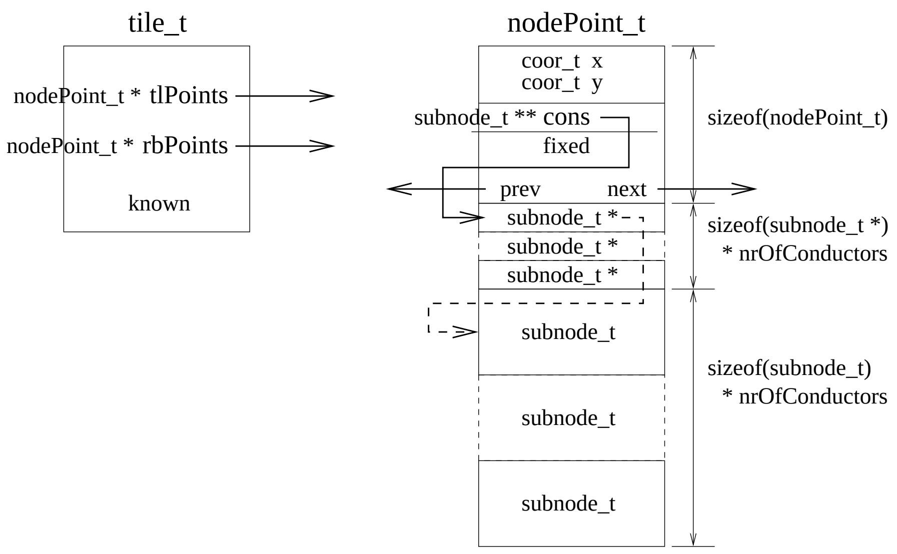
Nodepoints are used by the space program in the extract pass, when variable optRes is set. There are two cases, wherefor variable optRes is set, (1) for interconnect resistance extraction, and (2) for substrate resistance extraction. The nodepoints are intended for interconnect res extraction, in combination with or without substrate res extraction. The nodepoints are needed for high res interconnect, because there are resistors on the tile edges. See the two examples in the following figure:



Note that the functions enumPair and enumTile are not used by optRes, but the functions resEnumPair and resEnumTile. Functions enumPair and enumTile do not work with nodepoints, but only directly with subnodes in the tiles (tile→cons[cx], cx ≥ 0 and cx < nrOfConductors). The tile has only one subnode for each conductor found in that tile. See the following figure:



The functions resEnumPair and resEnumTile do not work with the subnodes in the tiles, but with subnodes in the nodepoints. The tile needs a double linked list of nodepoints, because a conductor in the tile can have more than one subnode (and node). See the following figure:



There are three tile nodepoint possibilities:

1. A tile has no nodepoints (tile→rbPoints = 0 and tile→tlPoints = 0).
   The tile contains no conductors.

2. A tile has one nodepoint (tile→rbPoints = tile→tlPoints = nodepoint).
   The tile contains only low sheet res conductors.

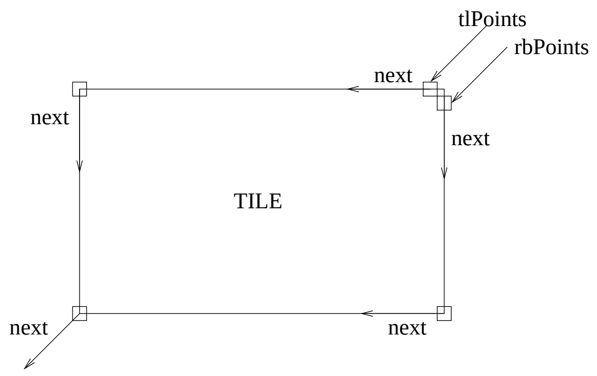
3. A tile has four or more nodepoints (tile→rbPoints != tile→tlPoints).
   The tile contains one or more high sheet res conductors.

The tile "known" bitmap is set, when there are high res conductors (for selective res extraction only, when the conductor is selected). Only for high res interconnect are resistors extracted. See code fragment resEnumPair:

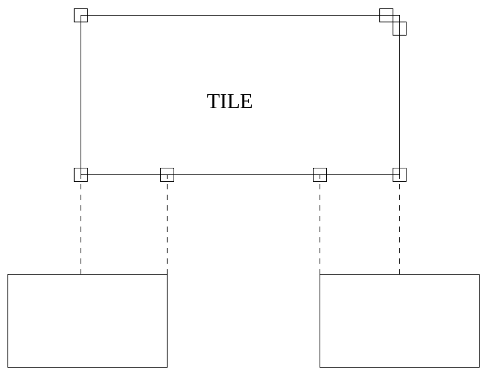
```
if (optIntRes && el -> s.res.val > low_sheet_res) {
    if (!extSelRes || selectSplit && selectSplit[cx] > 0)
        newerTile -> known |= (1 << cx);
}
```

The tile "tlPoints" pointer points to the last nodepoint for the tile top edge and left edge. And the tile "rbPoints" pointer points to the last nodepoint for the tile right edge and bottom edge. The "tlPoints" is moving from the tile lower left corner, along the tile left edge to the tile upper left corner, and along the tile top edge to the tile upper right corner. The "rbPoints" is moving from the tile lower left corner, along the tile bottom edge to the tile lower right corner, and along the tile right edge to the tile upper right corner. Both pointers start at the same nodepoint and end with a nodepoint in the tile upper right corner. A high res tile has always two nodepoints in the upper right corner position. The nodes of these two nodepoints are joined by resEnumTile.

The tile nodepoint construction process is done by resEnumPair. The first tile nodepoint (in lower left corner) has no "next" pointer. A new constructed nodepoint points always with its "next" pointer to a previous constructed nodepoint. Note that the nodepoint "prev" pointers are not used by resEnumPair. See the following figure:



Thus, each edge begin and end point (resEnumPair) gives nodepoints. When there are terminals, there can be extra nodepoints on vertical edges of a tile. Note that extra nodepoints on non-vertical edges are also created in the following case:



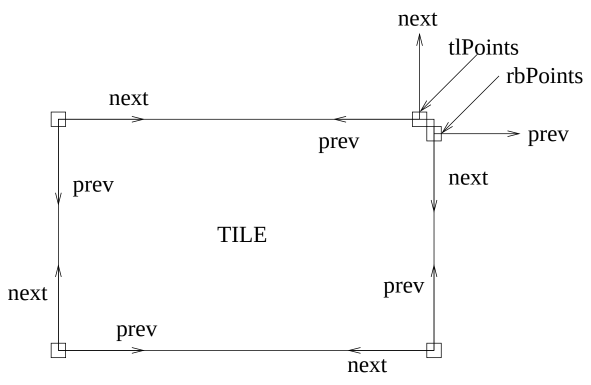
Extra nodepoints for a tile are only created, when the tile contains one or more high res conductors. The extra nodepoints for terminals are always fixed (see function placePoint). Other extra nodepoints are only fixed, when the tile and the adjacent tile contain the same conductor(s). In that case the subnodes point to the same node (by subnodeJoin/Copy).

When a tile edge is adjacent to a transistor gate area and the tile contains the transistor d/s conductor, then the d/s conductor subnodes along the d/s tile edge are joined and the nodepoints are set fixed (see function updateTorEdge).

When the tile and adjacent tile subnodes must be connected (by bipolar connect element), then the subnodes are joined and the nodepoints are set fixed (see function resBipoPair).

The unfixed extra nodepoints are removed in resEnumTile by rmUnfixedPoints (before calling function triangular). However, this is not done by optCap3D extraction. Function subnodeReconnect reconnects possible connected capacitances. Note that the tile corner nodepoints are never removed.

Function triangular changes the datastructure that links the nodepoints of the tile. For both "rbPoints" and "tlPoints" the "prev" pointer is set. For "tlPoints" the "next" pointer and "prev" pointer are swapped. The "rBPoints" and "tlPoints" are connected in one double linked list. See the following figure:



In case parameter "equi_line_ratio" is not set (the default), function triangular shall directly call function triangular_core. Function triangular_core performs the triangularization. It uses a tlP pointer, which starts with tlPoints and follows the prev pointer. And it uses a rbP pointer, which starts with rbPoints and follows the next pointer. The tlP and rbP pointers create triangles (and rectangles) until they meet. Each time function tryTriangle is called with three nodepoints. The first time (when pbS is null), it saves the triangle and waits for the next triangle. The second time, tryTriangle shall try combine the two triangles to a rectangle. When it is a rectangle, it calls doRectangle and resets pbS to null. When it is not possible to construct a rectangle, it calls doTriangle and saves the last triangle. On the end, when pbS is set, function triangular_core shall flush the last saved triangle (this is done by flushTryTriangle).

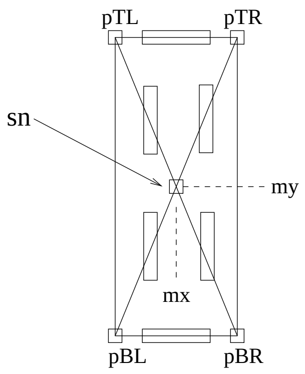
In practice, a tile which is a rectangle and does not have extra nodepoints, shall stay a rectangle (not be split in triangles). Otherwise, how the tile is split into triangles, is dependend of the length of the edges and the extra nodepoints found.

### 1.1 Equi-Potential Line Nodes

When the tile is a rectangle and equi_line_ratio > 0, then a check is done of the tile can be covered by an equi-potential line to introduce a node with an articulation degree > 1. The tile "conTB" or "conLR" flag needs to be set (not both). When "conTB" is set, there may not be extra vertical nodepoints (flag vPoint), and tile dy/dx ratio must be ≥ equi_line_ratio. When "conLR" is set, there may not be extra horizontal nodepoints (flag hPoint), and tile dx/dy ratio must be ≥ equi_line_ratio. In the simple case, function doEquiRectangle is called, when there are no extra nodepoints. Else, separate triangulation (by triangular_core) for each tile half is done.

The tile conTB flag is set for two horizontal adjacent tiles in resEnumPair, when both tiles contain the same high res conductor. And also for a tile, when the tile contains the high res d/s conductor of a horizontal adjacent transistor. The same rules apply for the conLR flag, but for vertical adjacent tiles.

For conTB are the following conductors added, see figure:



Default the area flag is set (to 1) for the sn→node (parameter "equi_line_area" is "on"), by a call to function makeLineNode. When the subnode "sn" is deleted, the node becomes ready, but is not delayed. And it makes the node more important by node elimination.

### 1.2 Equi-Potential Area Nodes

When the nodepoints contain nodes of a low res conductor, then the nodes are joined by subnodeJoin. This is done in functions doRectangle, doTriangle and doEquiRectangle. Only doRectangle and doEquiRectangle call also makeAreaNode, this sets the area flag (to 2) of the joined node. This makes the node more important by node elimination.

### 1.3 Node Groups

The doRectangle, doTriangle and doEquiRectangle functions add conductor elements between the nodes of the nodepoints of a tile. When these conductors are added, the nodes become in the same node group (for the nodes with the same conductor number). Later on, most nodes of the node group are eliminated. Only important nodes (like terminal nodes) are kept.

### 1.4 Conductor Mesh

The conductor or resistor mesh is constructed for high res conductors. The triangulation step does it. On the edges of the triangles and rectangles are the conductors placed. These edges (or the connections of elements between nodes) form the mesh. Note that extra nodepoints in tiles are responsible for triangles in the mesh.

### 1.5 Mesh Refinement

With option **-z**, a mesh refinement prepass is done. This step adds horizontal edges, which split the high res conductor tiles, especial on places where otherwise extra nodepoints were situated. During the extraction, the high res tiles are also split in vertical direction. The result is a mesh, which is build up of only rectangles. Note that terminal positions can also split tiles in vertical direction. Note that another type of refinement takes place for substrate res extraction, when the area of a tile can be too large.