SPACE APPLICATION NOTE ABOUT SUBSTRATE CONTACTS AND NODE GROUPS

S. de Graaf

Circuits and Systems Group
Faculty of Electrical Engineering
Delft University of Technology
The Netherlands

Report ET-CAS 02-04
December 3, 2002

Last revision: December 9, 2003.

1. Introduction

By many substrate contacts the node groups can stay too long in memory, because the substrate conductors (resistors) connects conductor groups with each other. I have made a modification to the *space* program, thus, that the substrate conductors don't introduce this problem anymore. Now, the group merging is not more done for the substrate conductors. The substrate conductors are flagged as special conductors and they shall normally be situated between different groups. When both nodes of a substrate conductor become in the same group, then the special conductor flag is put off. The following sections describe how this substrate extraction process works.
For extraction with the *space* program see the "Space User's Manual" [1] and for substrate extraction see the "Space Substrate Resistance Extraction User's Manual" [2].

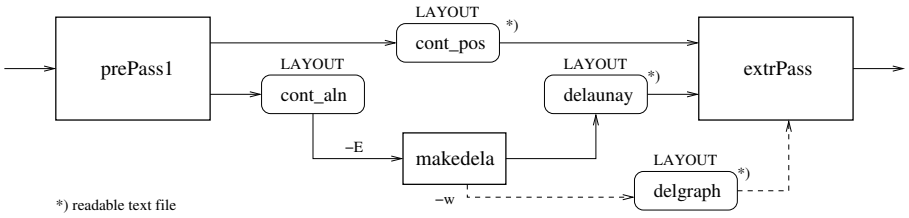
1.1 The Substrate Process

Substrate resistance extraction can be done with the *space* program with two different methods:

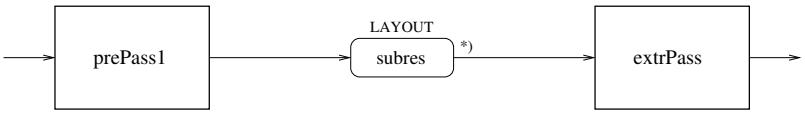
1. Simple 2D substrate resistance method (option **-b**).
2. Accurate 3D substrate resistance method (option **-B**).

Both methods are done by *space* in two passes (prePass1 and extrPass). See the figure below.

Simple Substrate Resistance Extraction (option –b)



Accurate 3D Substrate Resistance Extraction (option –B)



By the simple 2D substrate resistance method, prePass1 writes two files.
1) The "cont_pos" file contains the *area* and *perimeter* information of each substrate terminal (node) and an unique xl,yb coordinate (tile lowest left x coordinate and bottom y coordinate). The file contains also a substrate terminal number (1 t/m N) and a conductor

group number (1 t/m G).
2) The "cont_aln" file for the program *makedela*. This file contains all non-vertical substrate terminal edges. These edges must be horizontal (y1 == y2). Thus, in non-orthogonal layouts, the substrate terminals must only be build out of orthogonal layout parts. The program *makedela* writes a "delaunay" file for the extrPass. This file has a almost simular format as the "subres" file and contains distance information to the neighbor contacts of substrate terminals. It contains no group numbers and g_sub values.

In the extrPass, the files "cont_pos" and "delaunay" are read. The unique xl,yb coordinate is used to make the correct node link. In the extrPass the values are interpolated with technology data to calculate the wanted substrate resistance values. The technology file must contain "selfsubres" and "coupsubres" data.

By the accurate 3D substrate resistance method, prePass1 writes only one file. This "subres" file contains the calculated conductance values for the substrate terminals and between different substrate terminals. Which conductance values between neighbor contacts are calculated depends on the substrate extraction window used. The conductance values are calculated out of a spider mesh with Greens functions and a Schur matrix inversion. The technology file must contain "sublayers" data. The substrate calculations must be done in prePass1, because in the extrPass the same calculation method is used for capacitances. The "subres" file contains also a substrate terminal number (1 t/m N) and a conductor group number (1 t/m G).

Note that in the extrPass (optRes=1) the substrate conductance (resistance) elements are added between nodes in the network.

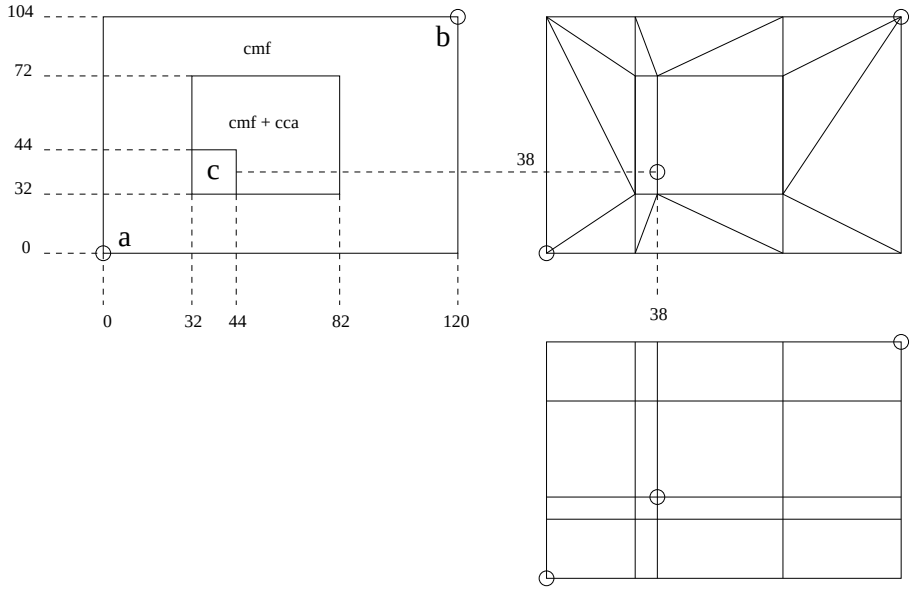
Question:
Contains one of the files "cont_pos" and "delaunay" not duplicated or unused data?

Answer:
Yes, the "delaunay" file contains xl,yb coordinates which are also found in the "cont_pos" file. This coordinates are not used.

> **NOTE:**
> Program *makedela* uses a Delaunay graph for making the relations between different substrate terminals and it calculates the distances.

2. Test Example 2

We use the following test example to test the use of substrate contacts. See the figure below (lambda=0.01 micron).



3. Test Example B1

We use a trivial test example to test the use of substrate contacts. See the figure below (lambda=0.01 micron).

Running the example (cmf=800 cmf/@sub=0) gives the following results:

```
% space3d -z -P space.testB.p -E space.testB.t testB
% xsls testB

network testB (terminal b, a)
{
    res 2.586892G (a, b);
    res 1.031535k (a, SUBSTR);
    res 1.031535k (b, SUBSTR);
}
```

Running the example with cmf/@sub=1 gives the following results:

```
% xsls testB
network testB (terminal b, a)
{
    res 4.374114k (a, b);
    res 2.5e15 (a, SUBSTR);
    res 2.5e15 (b, SUBSTR);
}
```

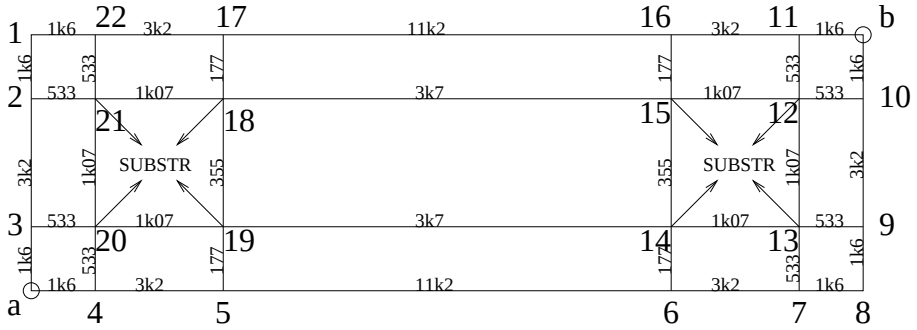
Thus, a contact resistance of 1 ohm gives very large substrate contact resistances. This because the contact (tile) surface is a important parameter in the calculation. In each corner there is initial a contact resistance of:

```
Rc = Rcontact / (0.25 * surface)
Rc = 1 / (0.25 * 2e-8 * 2e-8) = 1e16 ohm
```

In the first example is this contact resistance 0 ohm. Because the program can not work with 0 ohm resistors, the program shall use a fixed Rc of 1e-300 ohm.

Note: By setting parameter "max_res 1e15" we can remove the big resistors in the second example from the resulting network (the default max_res is 1e50).

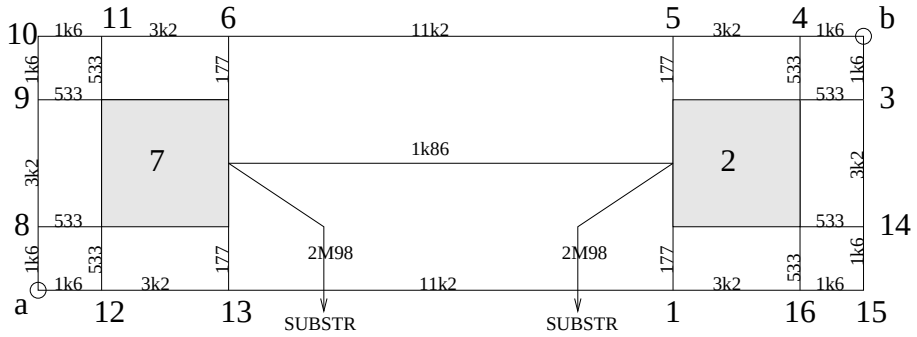
With parameter "node_elimination off" we can get the exact resistance mesh. The internal node numbering shows in which order the findGroup function finds the nodes.



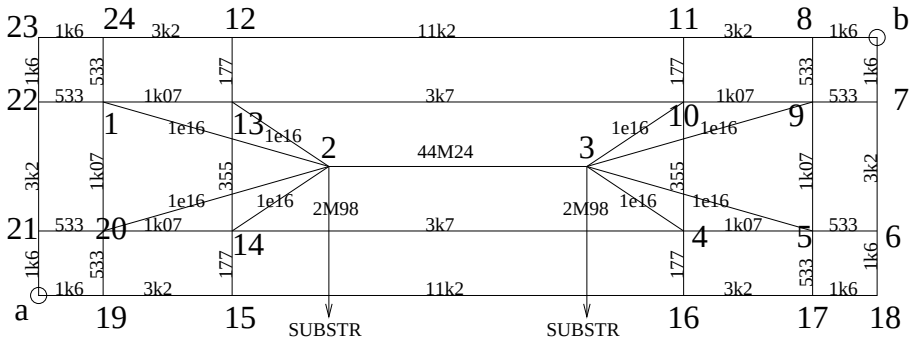
By substrate extraction the resistance mesh is a little different.

```
% space3d -zB -P space.testB.p -E space.testB.t testB
```

Mesh for contact res. of 0 ohm (nodes 2 and 7 are substrate nodes (cd=0)):



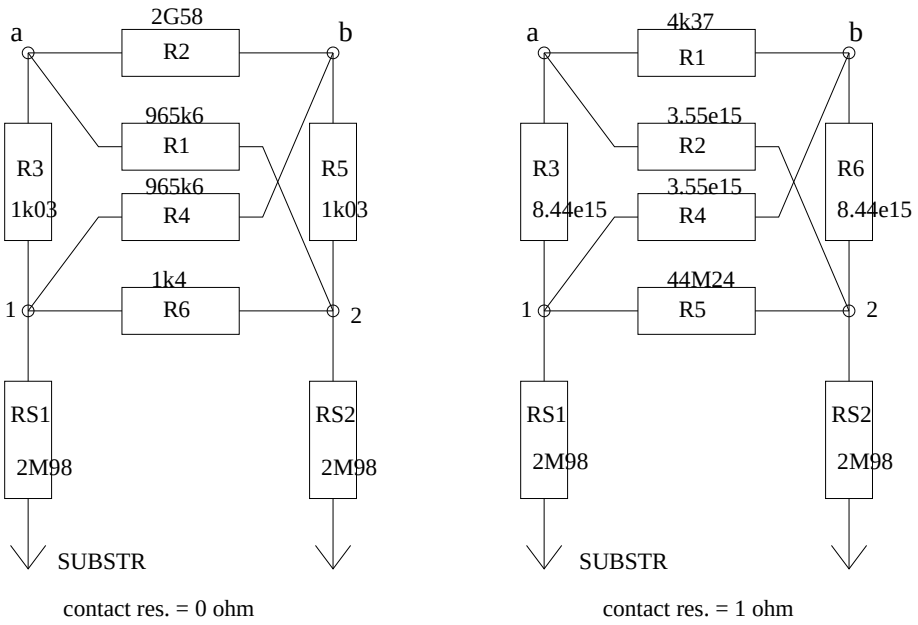
Mesh for contact res. of 1 ohm (nodes 2 and 3 are substrate nodes (cd=0)):



Note: There are only substrate nodes by substrate extraction!

The resulting networks after node elimination are:

```
network testB (terminal b, a) /* contact res. = 0 ohm */
{
    res 965.6185k (a, 2);
    res 2.586892G (a, b);
    res 1.032638k (a, 1);
    res 965.6185k (b, 1);
    res 1.032638k (b, 2);
    res 1.414445k (1, 2);
    res 2.982702M (1, SUBSTR);
    res 2.982702M (2, SUBSTR);
}

network testB (terminal b, a) /* contact res. = 1 ohm */
{
    res 4.374114k (a, b);
    res 8.438552e15 (a, 2);
    res 3.552445e15 (a, 1);
    res 8.438552e15 (1, b);
    res 44.23988M (1, 2);
    res 2.982702M (1, SUBSTR);
    res 3.552445e15 (2, b);
    res 2.982702M (2, SUBSTR);
}
```



Note: Nodes 1 and 2 are substrate nodes (cd=0), which are not eliminated.

With parameter "elim_sub_term_node" we can also eliminate the substrate nodes. The resulting networks with this parameter "on" after node elimination are:

```
network testB (terminal b, a) /* contact res. = 0 ohm */
{
    res 3.469047k (a, b);
    res 2.983733M (a, SUBSTR);
    res 2.983733M (b, SUBSTR);
}

network testB (terminal b, a) /* contact res. = 1 ohm */
{
    res 4.374114k (a, b);
    res 2.5e15 (a, SUBSTR);
    res 2.5e15 (b, SUBSTR);
}
```

When running only the prepass, the resulting network is:

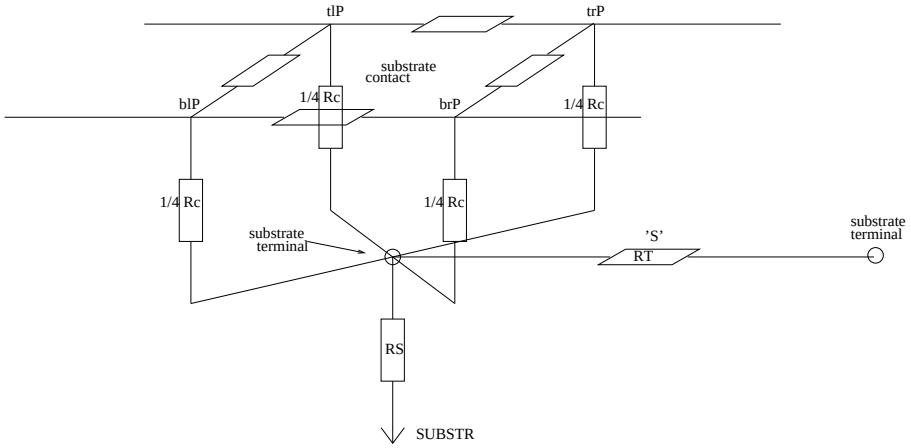
```
% space3d -%ZzB -P space.testB.p -E space.testB.t testB

network testB (terminal b, a)
{
    net {a, b};
    res 44.23988M (1, 2);
    res 2.982702M (1, SUBSTR);
    res 2.982702M (2, SUBSTR);
}
```

When running only **-B**, the resulting network is:

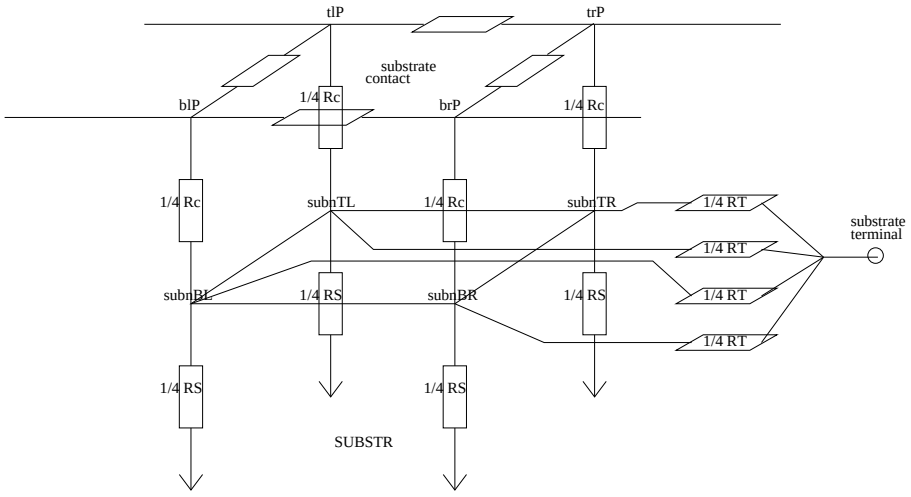
```
% space3d -B -P space.testB.p -E space.testB.t testB

network testB (terminal b, a)
{
    net {a, b};
    res 1.491351M (a, SUBSTR);
}
```

3.1 Test Example with distributed substrate contacts

Normally the substrate terminal contact node is one node. See following figure:



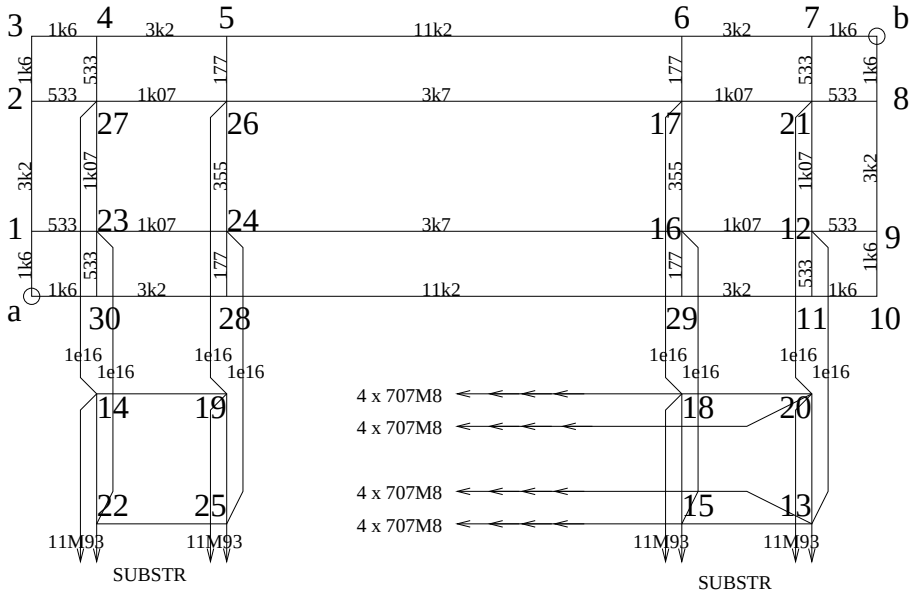
But it is possible (only with option **-B**) to get separate (distributed) substrate contact node points. This can be specified with parameter "sub_term_distr_*mask*", where "*mask*" is the name of a high res conductor mask, for example "cmf". See function makeContact and subnodeSubcontReconnect. Thus, when we specify "sub_term_distr_cmf", we get distributed substrate contacts for interconnect mask "cmf" (only if "cmf" is a high res conductor). Note that also the value of parameter "low_sheet_res" is important. See the result in the following figure:



The disadvantage of this method is, that the substrate resistors are distributed to the four seperate substrate contact node points. Thus you get a large number of resistors. Therefor, these contact nodes are not left in the network but eliminated. Note that the four subnodes are called subnTR, subnTL, subnBR and subnBL. These subnodes can be found in subcont -> subcontInfo. Because the information of subcont -> subn is distributed to these four subnodes, the subcont -> subn becomes empty. Thus the node of

subcont -> subn must not be outputted after a subnodeDel call. I have fixed the problem by joining subcont -> subn with one of the four new subnodes (subnBL).

The following resistance mesh is constructed with distributed nodes for a trivial contact resistance of 1 ohm:

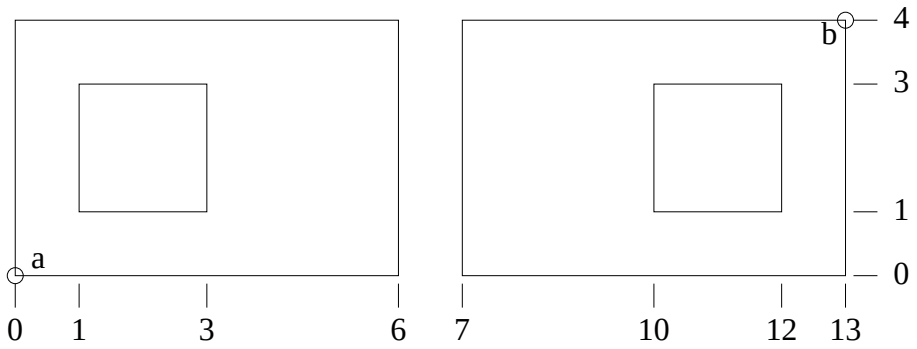


The resulting eliminated network is:

```
network testB (terminal b, a)
{
    res 4.374114k (a, b);
    res 2.5e15 (a, SUBSTR);
    res 2.5e15 (b, SUBSTR);
}
```

4. Test Example B2

We want to test substrate contacts which are in different groups. To do that, we split the layout of example 1 in two parts. See the figure below.



Running the example with *space3d* options **-zB** gives the following results:

```
network testB2 (terminal a, b) /* contact res. = 0 ohm */
{
    res 1.032119k (1, b);
    res 44.23988M (1, 2);
    res 2.982702M (1, SUBSTR);
    res 1.032119k (2, a);
    res 2.982702M (2, SUBSTR);
}

network testB2 (terminal a, b) /* contact res. = 1 ohm */
{
    res 2.5e15 (1, b);
    res 44.23988M (1, 2);
    res 2.982702M (1, SUBSTR);
    res 2.5e15 (2, a);
    res 2.982702M (2, SUBSTR);
}
```

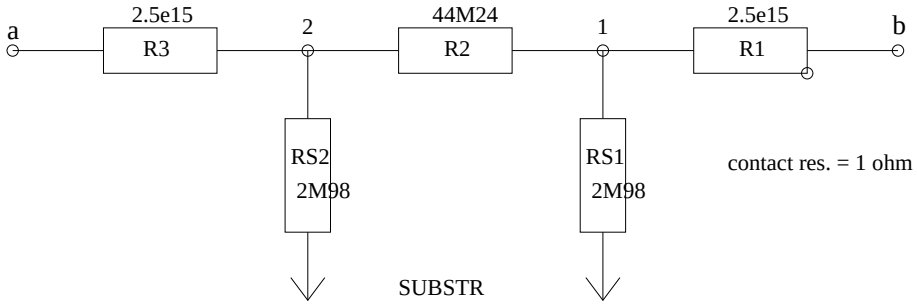
The group of terminal "a" is first ready, but must wait for adjacent group of terminal "b" becomes ready. When group "b" becomes ready, then are its adjacent groups ready.
Thus, group "b" is first outputted and gets conductor number cd=1. The adjacent group of "b" (group "a") can now also be outputted and gets conductor number cd=2.
Both groups had initial 17 nodes and 28 resistors (13 nodes and 20 resistors for contact res. = 0) and are both reduced to 2 nodes and 1 resistor.

Note that the special resistor between group "a" and "b" is not included in both group counts.

The "net" stream contains the following data (also option **-x** is used):

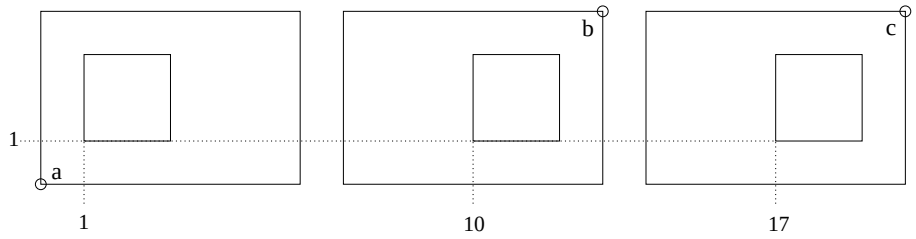
```
% dbcat -cs net testB2
=> circuit/testB2/net
net:"1" inst:"" attr:"cd=0;x=10;y=1" subnets:3
    subnet[0]:"p" inst:"_R1" attr:""
    subnet[1]:"p" inst:"_R2" attr:""
    subnet[2]:"p" inst:"_RS1" attr:""
net:"b" inst:"" attr:"cd=1;x=13;y=4" subnets:1
    subnet[0]:"n" inst:"_R1" attr:""
net:"2" inst:"" attr:"cd=0;x=1;y=1" subnets:3
    subnet[0]:"p" inst:"_R3" attr:""
    subnet[1]:"n" inst:"_R2" attr:""
    subnet[2]:"p" inst:"_RS2" attr:""
net:"a" inst:"" attr:"cd=2;x=0;y=0" subnets:1
    subnet[0]:"n" inst:"_R3" attr:""
net:"SUBSTR" inst:"" attr:"cd=0;x=0;y=0" subnets:2
    subnet[0]:"n" inst:"_RS1" attr:""
    subnet[1]:"n" inst:"_RS2" attr:""
```

A figure of the circuit:



5. Test Example B3

By three substrate contacts (and 3 groups) is the output order depended of parameter "sub3d.be_window". See the figure below (lambda=0.01).



With a be_window of "inf" we get the following network:

```
network testB3 (terminal c, b, a) /* contact res. = 1 ohm */
{
    res 2.5e15 (1, c);
    res 35.07018M (1, 2);
    res 91.39707M (1, 3);
    res 3.127989M (1, SUBSTR);
    res 2.5e15 (2, b);
    res 46.00796M (2, 3);
    res 3.227006M (2, SUBSTR);
    res 2.5e15 (3, a);
    res 3.07194M (3, SUBSTR);
}
```

The following steps are done: "a" ready, "b" ready, "c" ready, output "c" (cd=1), output "b" (cd=2), output "a" (cd=3).

With a be_window of 0.07 we get the following network:

```
network testB3 (terminal c, b, a) /* contact res. = 1 ohm */
{
    res 2.5e15 (1, a);
    res 44.23988M (1, 3);
    res 2.982702M (1, SUBSTR);
    res 2.5e15 (2, c);
    res 34.27222M (2, 3);
    res 3.033636M (2, SUBSTR);
    res 2.5e15 (3, b);
    res 3.241901M (3, SUBSTR);
}
```

There is no resistor between node 1 and 2. Thus group "a" and "c" are not adjacent.
The following steps are done: "a" ready, "b" ready, output "a" (cd=1), "c" ready, output "c" (cd=2), output "b" (cd=3).

Note that in both results the internal nodes have conductor number cd=0.

What happens, if we set parameter "elim_sub_term_node on"?

With a be_window of "inf" we get the following network:

```
network testB3 (terminal c, b, a) /* contact res. = 1 ohm */
{
    res 62.791e24 (c, a);
    res 27.39576e24 (c, b);
    res 2.5e15 (c, SUBSTR);
    res 35.27035e24 (b, a);
    res 2.5e15 (b, SUBSTR);
    res 2.5e15 (a, SUBSTR);
}
```

Note that "c" is cd=1, "b" is cd=2 and "a" is cd=3.

With a be_window of 0.07 we get the following network:

```
network testB3 (terminal c, b, a) /* contact res. = 1 ohm */
{
    res 27.39576e24 (c, b);
    res 433.7337e24 (c, a);
    res 2.5e15 (c, SUBSTR);
    res 35.27034e24 (a, b);
    res 2.5e15 (a, SUBSTR);
    res 2.5e15 (b, SUBSTR);
}
```

Note that "c" is cd=1, "a" is cd=2 and "b" is cd=3. Also in this case must group "a" wait for group "c". When the substrate terminal of group "b" is eliminated, there is added a conductor between group "a" and "c".

What happens, if we remove terminal "a"?
With a be_window of "inf" we get the following network:

```
network testB3a (terminal b, c) /* contact res. = 1 ohm */
{
    res 27.39576e24 (c, b);
    res 2.5e15 (c, SUBSTR);
    res 2.5e15 (b, SUBSTR);
}
```

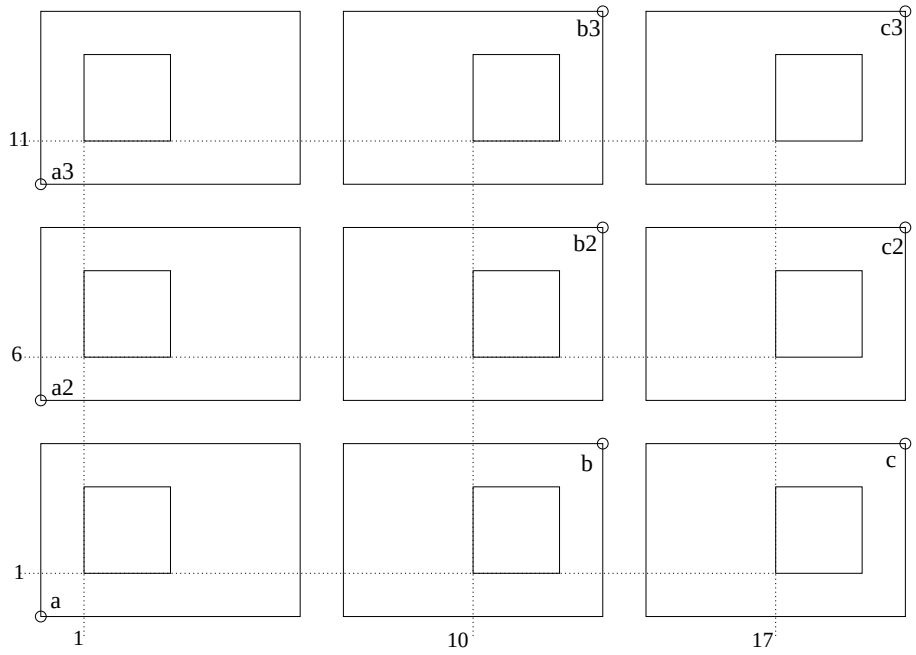
What happens, if we only remove terminal "b"?
With a be_window of 0.07 we get the following network:

```
network testB3b (terminal a, c) /* contact res. = 1 ohm */
{
    res 433.7337e24 (c, a);
    res 2.5e15 (c, SUBSTR);
    res 2.5e15 (a, SUBSTR);
}
```

Note that function elim must take care of the delete of the group.

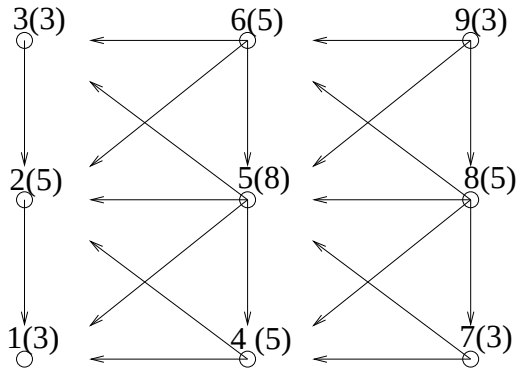
6. Test Example B4

We shall do the test also with an array of substrate contacts. See the figure below (lambda=0.01, be_window=0.07).



The following group output order is get:
"a" ready, "a2" ready, "a3" ready, "b" ready, "b2" ready, output "a" (cd=1),
"b3" ready, output "a2" (cd=2), output "a3" (cd=3),
"c" ready, "c2" ready, output "c" (cd=4), output "b" (cd=5),
"c3" ready, output "c3" (cd=6), output "c2" (cd=7), "b2" (cd=8), "b3" (cd=9).

The dependences between the substrate nodes is shown in the following figure (the number of neighbours is given between parentheses):



The prepass output file "layout/testB33/subres" gives also a good overview:

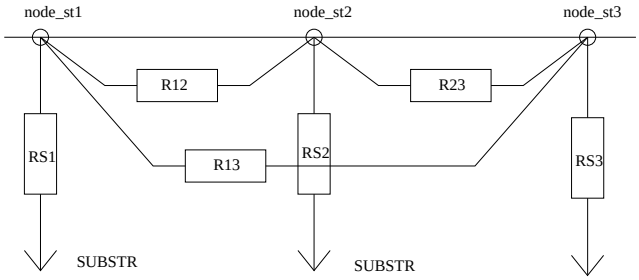
```
c 1 1 xl 4 yb 4 g_sub 2.891877e-07
nr_neigh 3
c 2 2 xl 4 yb 24 g_sub 2.431089e-07
nc 1 g 3.944055e-08
nr_neigh 5
c 3 3 xl 4 yb 44 g_sub 2.891877e-07
nc 2 g 3.944055e-08
nr_neigh 3
c 4 4 xl 40 yb 4 g_sub 2.518088e-07
nc 2 g 1.537202e-08
nc 1 g 1.899542e-08
nr_neigh 5
c 5 5 xl 40 yb 24 g_sub 1.951567e-07
nc 3 g 1.537202e-08
nc 2 g 1.538679e-08
nc 1 g 1.537202e-08
nc 4 g 3.665031e-08
nr_neigh 8
c 6 6 xl 40 yb 44 g_sub 2.518088e-07
nc 3 g 1.899542e-08
nc 2 g 1.537202e-08
nc 5 g 3.665031e-08
nr_neigh 5
c 7 7 xl 68 yb 4 g_sub 2.825258e-07
nc 5 g 1.808033e-08
nc 4 g 2.499786e-08
nr_neigh 3
c 8 8 xl 68 yb 24 g_sub 2.354143e-07
nc 6 g 1.808033e-08
nc 5 g 2.081756e-08
nc 4 g 1.808033e-08
nc 7 g 3.845923e-08
nr_neigh 5
c 9 9 xl 68 yb 44 g_sub 2.825258e-07
nc 6 g 2.499786e-08
nc 5 g 1.808033e-08
nc 8 g 3.845923e-08
nr_neigh 3
```

The circuit listing is given below:

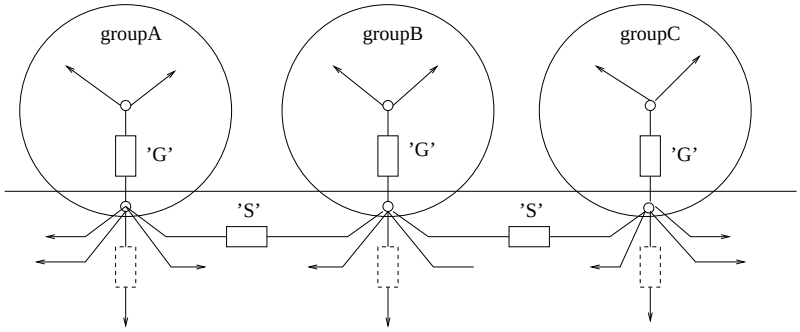
```
/* Generated by: xsls 2.34 27-Sep-2000 */
/* Date: 5-Dec-02 8:59:25 GMT */

network testB33 (terminal c3, c2, b3, b2, a3, a2, a, b, c)
{
    res 2.5e15 (1, a);
    res 65.05326M (1, 8);
    res 52.64427M (1, 5);
    res 25.35462M (1, 2);
    res 3.457962M (1, SUBSTR);
    res 2.5e15 (2, a2);
    res 65.05326M (2, 9);
    res 64.99081M (2, 8);
    res 65.05326M (2, 5);
    res 25.35462M (2, 3);
    res 4.113383M (2, SUBSTR);
    res 2.5e15 (3, a3);
    res 52.64427M (3, 9);
    res 65.05326M (3, 8);
    res 3.457962M (3, SUBSTR);
    res 2.5e15 (4, c);
    res 26.00156M (4, 7);
    res 40.00342M (4, 5);
    res 55.30873M (4, 8);
    res 3.5395M (4, SUBSTR);
    res 2.5e15 (5, b);
    res 55.30873M (5, 7);
    res 27.2849M (5, 8);
    res 3.971267M (5, SUBSTR);
    res 2.5e15 (6, c3);
    res 26.00156M (6, 7);
    res 55.30873M (6, 8);
    res 40.00342M (6, 9);
    res 3.5395M (6, SUBSTR);
    res 2.5e15 (7, c2);
    res 48.03637M (7, 8);
    res 55.30873M (7, 9);
    res 4.24783M (7, SUBSTR);
    res 2.5e15 (8, b2);
    res 27.2849M (8, 9);
    res 5.124087M (8, SUBSTR);
    res 2.5e15 (9, b3);
    res 3.971267M (9, SUBSTR);
}
```

7. What is a Substrate Terminal?

A node connected to the SUBSTR node is not always a substrate terminal. Only in case of substrate extraction we can speak about substrate terminals. And even in that case is a node connected to the SUBSTR node not always a substrate terminal, because substrate terminals can be eliminated. Substrate terminals are nodes in the substrate, which are caused by contacts and the surface (or a part of the surface) of transistor gate area or capacitor area (in case of capacitance extraction). These substrate connections are specified in the technology file. Thus can for transistors a bulk connection be specified. Thus can for capacitors, in place of capacitance to GND, a capacitance to substrate (a terminal node) be specified. By substrate extraction is this substrate node not the SUBSTR node. Note that in case of substrate extraction the substrate is modeled as a resistor network (see the figure below).



Hard connections with the substrate can be specified by contacts. A contact is a connection between interconnect and the substrate. There is normally a contact resistor between the interconnect node and the substrate terminal node. Note that the contact resistors are modeled in the substrCon array in case they are connected directly with the SUBSTR node. Thus, the SUBSTR group is never joined with another group. To prevent that interconnect resistor groups are joined with each other via the substrate resistors, these substrate resistors are flagged as special. This is done by giving them the type 'S' in place of 'G'. The substrate terminals belong to the conductor group cd=0, while the nodes itself can be in a interconnect group. For an impression, see the following figure:



8. Node Elimination

Tiles (and nodepoints) contain subnodes, these subnodes point to nodes. Subnodes can be very small layout pieces, thus, many subnodes can point to the same node. By resistance extraction (in the extract pass) are nodepoints used in the corners of each tile. Thus, on each edge of the tile can be a conductor element. And also inside the tile, when it is split in triangles. We speak about a 2D resistance mesh. The thickness of the interconnect is not taken into account. Note that only tiles, which have conductors, have nodepoints. And that only tiles, with a set known bitmask, have nodepoints in every corner. Nodes connected with each other by conductor elements are placed in the same node group.

When a node becomes ready (see readyNode), the node is possibly delayed. When the last node of the group becomes ready, the notReady group flag becomes zero and this last node is given to function readyGroup. Nodes that are delayed, are nominated to be eliminated. There is a node elimination priority queue. Each node is placed in the queue by creation. But delayed nodes are placed on other positions in the queue. The degree of the node, howmany connections the node has with other nodes is important. Nodes with the lowest degree are first eliminated. Note that the connected elements of an eliminated node are given to neighbour nodes (a network recalculation takes place). Note that not all nodes of a node group are delayed. The node "keep" and "term" flag prevents node elimination and if artReduc is true, also the node "area" flag does. (Also nodes with not ready "pols" nodeLinks are not delayed.) Nodes, which must be delayed, are given to function nqDelayElim. Note that function nqDelayElim shall eliminate a node, when there become too many delayed nodes.

Note that delayed nodes can also be undelayed. When the elimination process has eliminated some nodes, other connected nodes can get more priority (more connected elements), which set the keep flag again by function testRCelim.

8.1 The node keep flag

Where is the node "keep" flag used for? Keep means, that the node not must be deleted (eliminated). There is also a node "term" flag, which prevents the elimination. Note that function reducGroupI reuses the "keep" flag.

Function readyNode shall set the node "keep" flag for the first time, this is done by testRCelim. Function testRCelim sets the "keep" flag for a node when:
1) different sort of resistors are connected to the node.
2) different sort of capacitors or 'polar' capacitors are connected to the node and no resistors are connected to the node.

8.2 The node term flag

The node "term" flag is set for terminals and labels. Nodes, that are instance terminals (ports) of (bipolar) transistors, are also set. Also substrate terminal nodes are set (and node nSUB). Note that this is not done for substrate terminals, which may be eliminated

in the extract pass.

The "term" flag is more important than the "keep" flag. When function readyGroup calls function reducGroupI, shall function reducArtDegree not eliminate nodes with the "term" flag set. Note that reducArtDegree reuses the "keep" flag. Function reducMinRes shall use this new set "keep" flag. But this function (and the other reduction functions) don't look for the "term" flag! Thus, it is possible, that nodes with the "term" (or "keep") flag set are eliminated. But, note that reducMinRes/reducRmNegRes shall do a nodeRelJoin.

Note that option -%f (optFineNtw) or parameter "network_reduction off" set the "term" flag for all nodes (in function createNode).

8.3 Note about nqDelayElim

Function nqDelayElim delays the elimination of nodes which must be eliminated. The maximum number of nodes which are delayed if default 500 and can be controlled with parameter "max_delayed". Thus, when the maximum is reached, nqDelayElim shall retrieve a delayed node and eliminate this node. Note that function testRCelim is called. If the function has set the "keep" flag, then the node is not eliminated, but undelayed. Note that the node is removed from the priority queue with nqDelete and is not inserted again with nqInsert. Thus, this node can only be found back with another node in the same group, or with an adjacent group node.

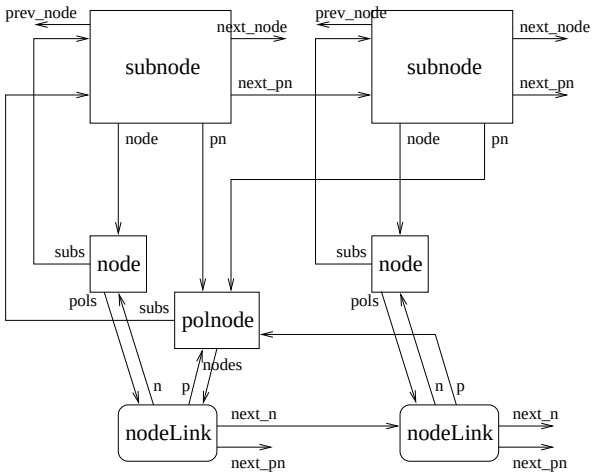
8.4 Note about eliminateGroup

Also function eliminateGroup (and nqEliminateGroup) can undelay a delayed node. This happens, when function testRCelim sets the "keep" flag. The node is removed from the priority queue with nqDelete and not inserted again with nqInsert. But the node is also put back in the readyGroup qn array (to be outputted). Note that outGroup shall call outNode and outNode does nodeDel. Note that nqDelete shall be done again, while the node is not more in the priority queue. But this has not given any problem yet.

9. Note About Polnodes

If there are bipolar technology elements (L/VBJTELEM, L/VJUNELEM, PNCONELEM) then the hasBipoElem flag is set. If the hasBipoElem flag is set, then not only subnodeNew is done, but also polnodeAdd (and nodeLinkAdd). In that case each subnode points to a node and a polnode. The link between node and polnode is made in datastruct nodeLink. Note that the node group notReady counter is incremented for each nodeLink. When there are conductors, there can be conductors between the nodes, while the polnodes can be joined. Thus, there can be more nodes than polnodes.

When a subnode is finished and subnodeDel is done, then the subnode is also removed from the polnode and node subnode-list. The polnode subnode-list is not yet empty (no polnodeDel is done), but the node subnode-list becomes empty and readyNode is called. Function readyNode shall possible set the node "keep" flag and shall decrement the node group notReady counter for this ready node. But, because the node polnode list (pointer pols) is not empty, we must wait till the nodeLink is removed. See the following datastruct figure:



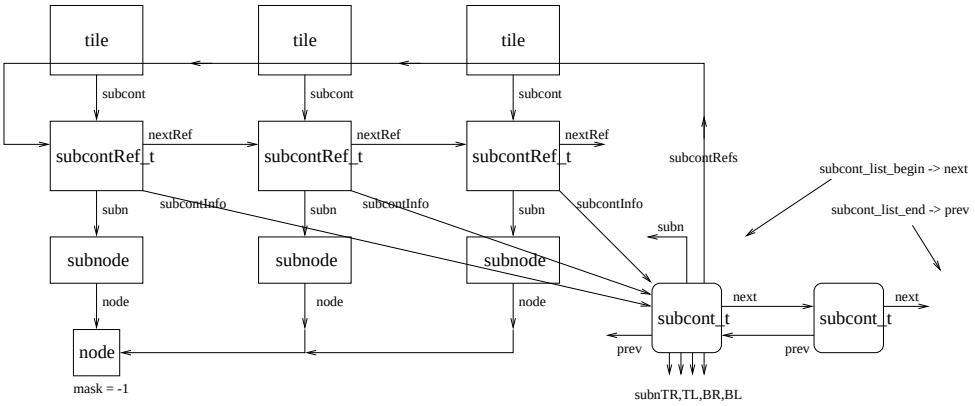
Note that nodes with a pols pointer unequal null are not given to function nqDelayElim.

When the second subnode is finished and subnodeDel is done, then polnode subnode-list becomes empty and polnodeDel is done. Function polnodeDel shall remove both nodeLinks. Thus, for both nodes the pols pointer becomes null. For the first node the group notReady counter becomes 0, which result in a call to readyGroup by nodeLinkDel. For the second node the group notReady counter becomes 1.

The second node subnode-list becomes also empty and readyNode is done. The group notReady counter is decremented and becomes 0. Now, because the pols pointer is null, readyGroup is called.

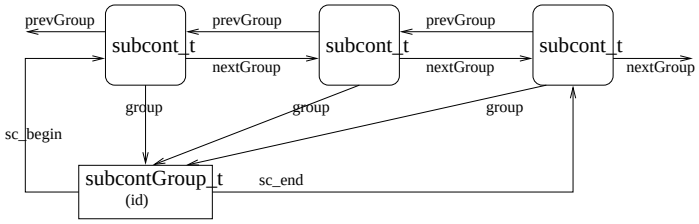
10. Note About Substrate Subnodes

Interconnect mask (conductor) subnodes can be found in the tile datastruct cons-array or with tl/rbPoints pointers in the nodePoint datastruct for resistance extraction. The subnodes for the substrate can be reached with the tile subcont pointer. The tile subcont pointer points to a substrate terminal, when there is one found during substrate resistance extraction. For the subcont datastructs see the following figure:



Several tiles together form one substrate contact. When the contact is ready, the last subnode is moved to subcontInfo -> subn. Thus, there is always a node available for handling of the substrate contact. All subcont_t are placed into the subcont_list and are outputted in sorted order in prepass1 (when ready). Note that a distributed subcont_t has four subnodes (and nodes).

A number of subcont_t's can also form a group, see following figure:



For optSimpleSubRes (**-b**), in prepass1, the adjacent substrate terminals (subcont_t) are always joined. For optSubRes (**-B**), in prepass1, only not distributed subcont_t's are normally joined, others not. For the other subcont_t's, only the subcontGroup_t's are joined. Note that a joined subcont_t has only one subcontGroup_t. A subcontGroup_t contains a subcont group identification number, which is outputted in prepass1. In the extract pass, this group id is stored in the group pointer and used in subContAssignRes (neighbour contact id must not be equal to current contact id).

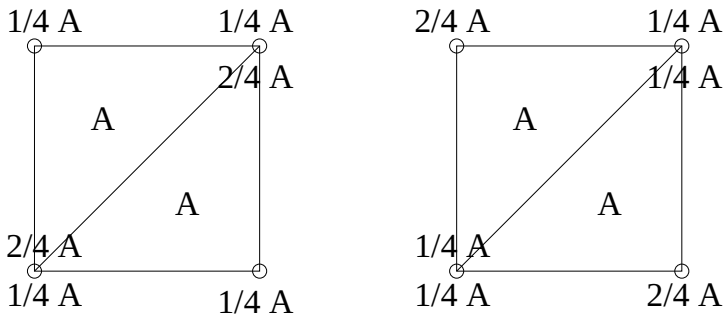
11. Note About Contact Resistances

If the surface of a tile is a contact area, then there are by resistance extraction contact resistors between the two pieces of interconnect. But it can also be a substrate contact, a contact between interconnect and substrate. A contact resistance of 0 ohm can give a problem for a substrate contact, because in that case an interconnect node is joined with a substrate node. When no substrate resistance extraction is done, it is a problem, because the substrate node is nSUB and this node may never be joined with other nodes. In that case we must add a trivial small contact resistance.

Note that normal via's (interconnect contacts) node positions are eliminated. This is not the case for substrate terminal contact nodes. But substrate terminals can also be eliminated with parameter "elim_sub_term_node".

I have chosen to model a resistance to nSUB always in the node array substrCon. Thus, there are never 'con' elements connected with nSUB. The same model is used for 'cap' elements. Couple caps to node nSUB are always in node array substrCap. Thus, these arrays are not only used for substrate resistance extraction.

In orthogonal layouts, the contact tiles are always rectangles. But this is not always the case for non-orthogonal layouts. But also in orthogonal layouts it is possible that tiles are divided in triangles. If the contact tile is a triangle, it is not a good idea to put 1/4 contact resistance in each node point, because there are only 3 node points. In practice, the wrong node point gets 1/2 the contact resistance. For example, if two corners are 45 degree (contribution of each = 45/180 * R_contact) and one corner is 90 degree (contribution = 90/180 * R_contact), then the 90 degree corner must get 1/2 the contact resistance. See the following figure:



Wrong solution! Correct solution?

12. Other Notes and Questions

If you are interrested in transistor info, don't extract anything. Resistors and capacitors are only extracted with certain options, but transistors are always extracted.

Prepass0 generates the mesh data, prepass1 generates the substrate resistance data. These prepasses can be reused, when you extract the same cell with the same parameters. Either this is not done.

Because selective resistance extraction has an undocumented **-s** option, transistor elements must be extracted in prepass2. Maybe we can remove this option?

The substrate resistance prepass1 works with **-b** and **-B**. Option **-b** does not generate conductor elements, but calculates area and perimeter values. The **-B** method uses spiders and nodes, and the conductor elements are between these nodes. Maybe **-B** can also use the method of option **-b**?

Why can a net name not be a terminal name which has a copy x and/or y?

The conductor values used in a tile must each time again be loaded in a conductor values conVal[] array (see enumtile.c). The reason for this is, that a conductor number can have different conductor values. The code can not directly access the values in the conductor technology elements. The tile color is used as an access key to retrieve the technology elements. Note that a very compact retrieval mechanism is possible, when we use one bit for each technology element. Each bit can directly tell us the position of the technology element in the element array. For each key we need only shift the bits to find the elements.

A tile has many nodepoints. A tile has one color. Contains each nodepoint the same conductors?

A "net" stream can contain references to non existing _R or _C instances. A "net" stream can contain double references to the same "d" of a transistor in the same net. The program *xsls* can handle this situations. A net can also contain unconnected internal nodes, *xsls* does not print them. The "net" stream can also contain dangling nodes or dangling res elements.

The conductor cd=0 nets can be everywhere in the "net" stream.

The code in source file "lump/out.c" is heavy modified. Note that i have comment out incomplete code parts for element instances of IND (_Lx), JUN (_Jx) and GJUN (_GJx). When these elements are outputted, there must be netEq's for both nodes and there must also be a value to be outputted.

References

1. A.J. van Genderen, N.P. van der Meijs, F. Beeftink, and P.J.H. Elias, ‘‘Space User’s Manual,’’ Report ET-NT 92.21, Delft University of Technology, Network Theory Section, Delft, the Netherlands (June 2001).

2. A.J. van Genderen, N.P. van der Meijs, and T. Smedes, ‘‘Space Substrate Resistance Extraction User’s Manual,’’ Report ET-NT 96.03, Delft University of Technology, Network Theory Section, Delft, the Netherlands (July 2002).

CONTENTS