Space node reduction and dangling nodes

S. de Graaf

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

Report EWI-ENS 11-04
August 8, 2011

Last revision: October 27, 2011.

1. INTRODUCTION

When doing a *space* resistance extraction, recent was detected that the node reduction proces can lead to dangling local nodes, which are connected with only one resistor. This happens by a small change to the "oscil" demo example. This report explains what is the case and what must be done to fix it.

First is explained how the node group of the "oscil" example normally is reduced without the occurance of the dangling node problem. The reduction step is done when the node group becomes ready. Then function readyGroup is called. First, all nodes are placed in the array qn. A summary of this array can be printed with parameter "debug.ready_group". Second, all delayed nodes are eliminated with function nqEliminateGroup. Third, the remaining nodes are reduced by calling function reducGroupI. I shall explain what is done by function reducGroupI.
Note that the following *space* command is used and a part of the output is given:

```
% space -rF -Sdebug.ready_group=1 oscil
space: warning: no node join of subnodes with different conductor-type
        for conductor mask 'caa' at position (16, 32).
ready_grp: nodes=12 (delayed=12 term=0 keep=0 area=0) adjgrps=0
  ...
ready_grp: nodes=69 (delayed=58 term=4 keep=0 area=7) adjgrps=0
ready_grp: nodes=136 (delayed=126 term=10 keep=0 area=0) adjgrps=0
ready_grp: nodes=137 (delayed=126 term=11 keep=0 area=0) adjgrps=0
ready_grp: nodes=76 (delayed=64 term=4 keep=0 area=8) adjgrps=0
ready_grp: nodes=1 (delayed=0 term=1 keep=0 area=0) adjgrps=0
```

First you want to ask, where is the first warning message come from? Well, different conductor types of active area are touching each other and can not be joined.
And second you want maybe ask, why are there so many ready groups? That is indeed a good question, there need not to be so many groups. I found out, that accidental nwell is laying under the poly to metal contacts. That results in 9 isolated ready groups with only 1 node. Second, the active area mask (caa) is also used for the 9 nwell and 9 substrate contacts. This is completely unnecessary when looking to the technology file. It results in the warning message and results in 18 isolated ready groups with 12 nodes each.

Note that the last ready group is of the "sens" and has 1 terminal node. The ready group with 76 nodes is of the last poly (cpg) area. The ready group with 137 nodes is of the "vdd" power supply line and nwell area. The ready group with 136 nodes is of the "vss" power supply line. The ready group with 69 nodes is of the second last poly area and is our group of interest. See the following page for a drawing of this group and the drawing of the resistor network after elimination of the delayed nodes. Function readyGroup is calling reducGroupI with this initial resistor network.

Figure 1 shows the resistor mesh between the nodes and shows also the special nodes with small circles. Figure 2 shows the remaining resistor network between the 11 special nodes after the elimination of the 58 delayed nodes.

Below, figure 1 shows the resistor mesh and figure 2 the initial resistor network.

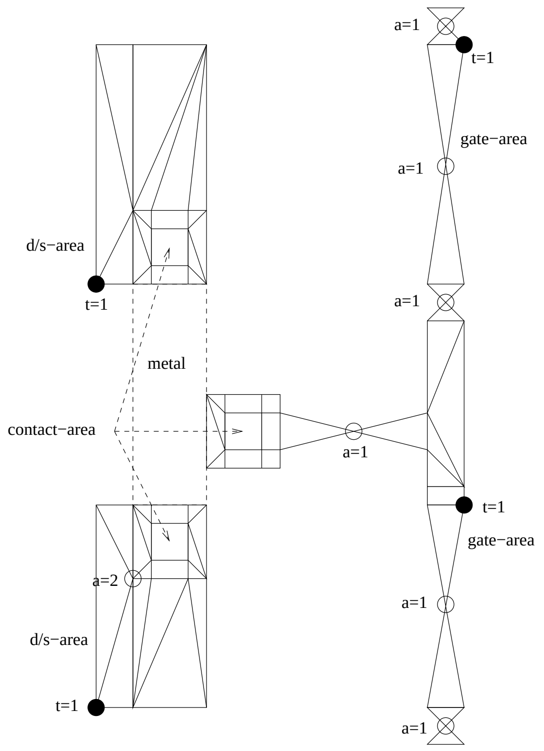


Figure 1

nodes:

2 x d/s−area	30
poly−area	38
metal area	1

total # nodes: 69

4 terminal nodes
1 area=2 node
6 area=1 nodes

total # nodes: 11

delayed nodes: 69 − 11 = 58

This is the resistor network after
the elimination of the delayed nodes.
There are 11 remaining nodes.

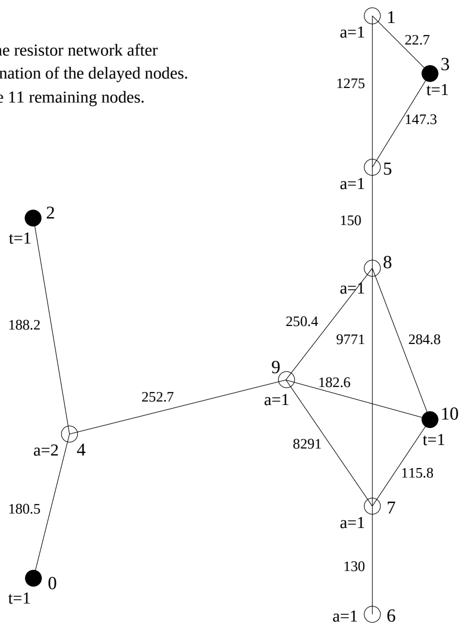


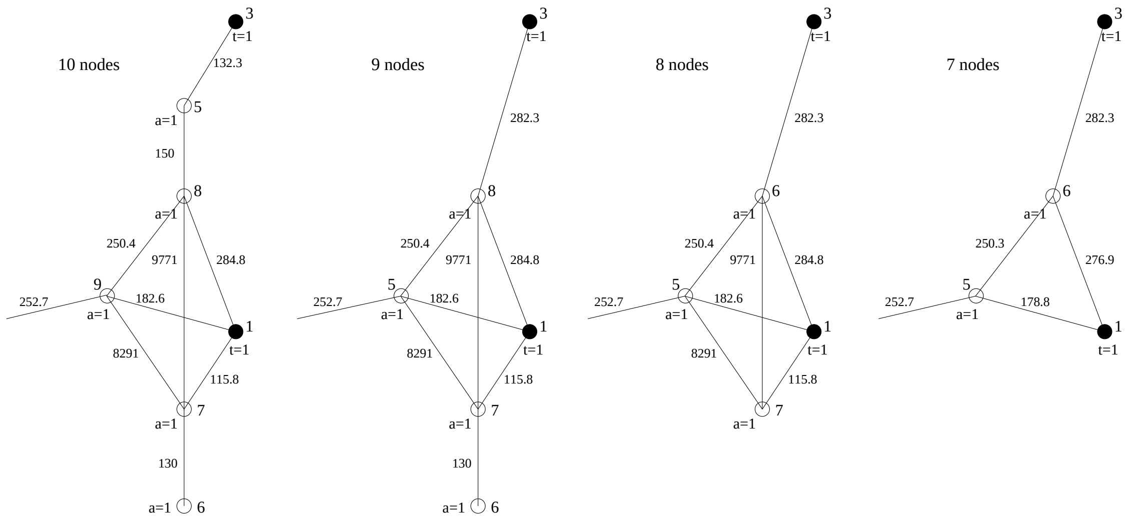
Figure 2

Note that the order of the nodes
in array qn is important.

The order is given by the array
index numbers 0 − 10.

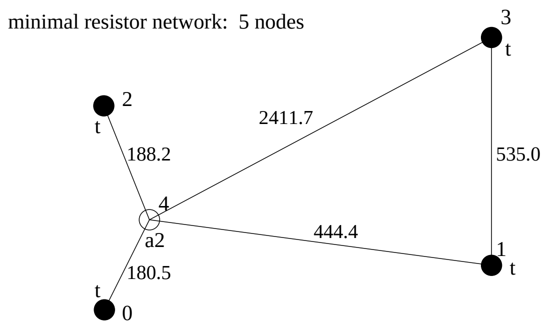
In function reducGroupI, we start with the remaining nodes of the resistor network (see figure 2 above). Some special nodes can not be eliminated because they are terminal nodes. The 4 terminal nodes (with t=1 flag), which are also a pin of a transistor. Note that for the 2 d/s-area nodes (on the left) the nodes touching the gate edge are joined together. And note, that the 2 gate terminal nodes (on the right) are in the top-right position of the gate area (see figure 1 above).
The 7 area nodes, however, can possibly be eliminated. This is depended of the chosen articulation reduction heuristic parameters. Parameter **min_art_degree=3** and parameter **min_degree=4** (are typical values) and are used by function reducArtDegree. Note that 6 line nodes (with a=1 flag) were created in the poly area and 1 area node (with a=2 flag) was created for the low resistive metal area. And note that this metal area connects the d/s areas with the poly area (see figure 1 above).

The first node found by function reducArtDegree, which may be eliminated is node 1. It has a degree=2 (res_cnt=2) which is less than parameter min_degree, thus is eliminated. The resistor between nodes 3 and 5 becomes now 132.3 ohm. The last node (10) becomes now node 1 in the qn array and is now checked by reducArtDegree. The next node which possibly can be eliminated is now node 4. But this node has a degree=3, which is >= min_art_degree, thus is not eliminated. Node 5, however, can be eliminated (degree=2 < min_degree). The resistor between node 3 and 8 becomes now 282.3 ohm. Node 9 becomes now node 5 in the qn array. The new node 5 is not eliminated, because degree=4 >= min_degree. But node 6 can be eliminated, because its degree=1 < min_degree. There are now 8 nodes left (see figures below).



Finally, node 7 can be eliminated, because its degree=3 < min_degree. This results in 7 nodes. Because nodes 5 and 6 are not more evaluated, they are not eliminated. But when another initial node order in qn was used, at least 1 more node could be eliminated.

When nodes 5 and 6 are eliminated, the following network can be the result:



Note that function reducArtDegree re-evaluates the keep status of the nodes (see code below). For "term=2" nodes the keep status is always set to 1. For other nodes the keep status is reset to 0 and evaluated by function testRCelim, which can set the keep status again. Nodes with "keep=1" are not futher evaluated by reducArtDegree. However, for nodes evaluated by reducArtDegree the keep status can be set. Because the node is an area node which may not be eliminated. Note that for "term=1" nodes also the keep flag can be set, but only when parameter min_art_degree <= 1.

Note that the node keep status after that only is evaluated by function reducMinRes. Function reducMinRes shall skip the nodes with a set keep status. Thus only "term=1" nodes with "keep=0" (which are not area nodes) are evaluated by function reducMinRes. When this heuristic is used (parameter min_res > 0) and the resistor value is < min_res the nodes are joined by a nodeRelJoin and the node is eliminated.

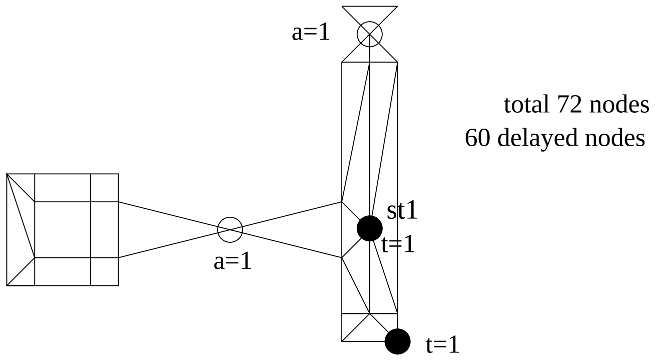
Note that nodeRelJoin does not set the "term" status of the remaining node. This is not directly a problem for the other reduction functions (because they don't test for "term=1" status anymore), but it is a problem when new code must test for a set "term" status of the node. To overcome this problem, new code must also test for node "names".

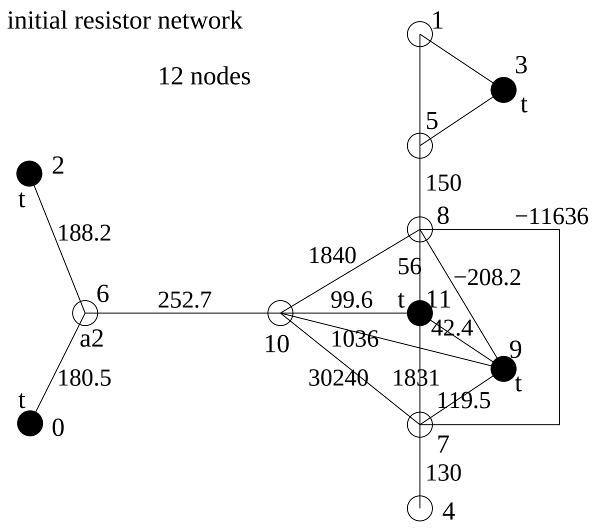
Note that "area=1" nodes are not eliminated when **min_art_degree <= 2**. And that "area=2" nodes are not eliminated when min_art_degree <= degree. Thus, a choice for a min_art_degree <= 2 can result in dangling non terminal nodes with only one resistor (for "area=2" nodes when **min_art_degree <= 1** because the degree=1). Also a too low value for **min_degree (< 2)** for "area=1" nodes can result in dangling non terminal nodes.
See the following part of the reducArtDegree code:

```
if (n->term < 2) { n->keep = 0; cx = testRCelim (n); } else { n->keep = 1; }
if (n->keep == 0 && artReduc && cx >= 0) {
   degree = 0;
        if (n->area == 1) degree = 2;
   else if (n->area == 2) { ... } /* compute art degree */
   else if (n->term == 1) degree = 1;
   if (degree >= min_art_degree || degree >= 2 && n->res_cnt >= min_degree) n->keep = 1;
}
if (n->keep == 0 && n->term == 0) { elim (n); ... } /* and remove n from qn */
```

2. DANGLING NODE PROBLEM INTRODUCTION

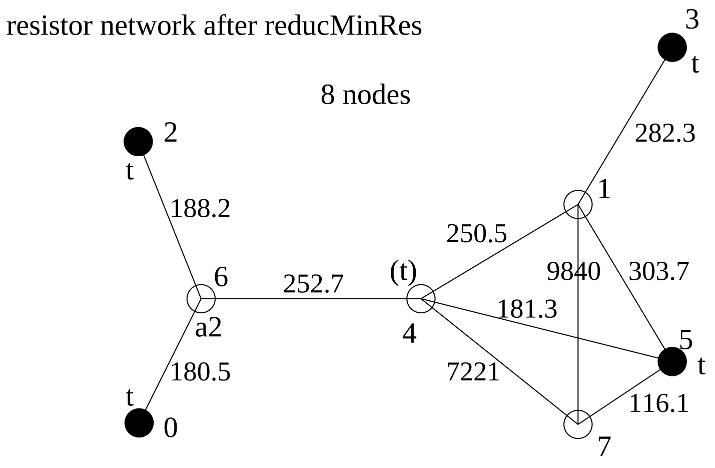
When on the poly area an extra terminal is placed (for example "st1"), then the resistivity mesh is locally a little changed. This also add 3 more nodes to the node group, see the figure below.





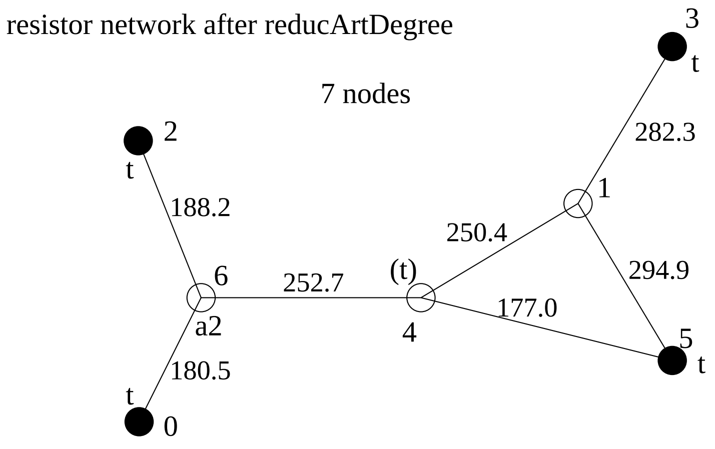
Function reducArtDegree can eliminate 3 nodes from the above initial resistor network. The following nodes: node 1 (node 11 -> 1), node 4 (node 10 -> 4) and node 5 (node 9 -> 5). This results in a 282.3 ohm resistor between node 3 and 8 and nine remaining nodes.

Function reducMinRes can eliminate 1 resistor (and 1 node) less than 100 ohm from the above resistor network. It has a choice between 3 resistors connected to node 1 (was 11). It takes the first one found and that is the 99.6 ohm resistor. That is not the best choice. Nodes 1 and 4 (was 10) are joined. Node 4 has the keep status, thus node 1 is eliminated. Note that node 4 should get the term status from node 1. Node 8 becomes node 1 and thus there are 8 remaining nodes. See the resistor network on the next page.



Function reducMaxParRes can now try to remove too large resistors. Candidates are the 7221 and 9840 ohm resistor. Function reducMaxParRes shall first start with a non area node and that is node 5. Nodes 5, 4, 1 and 7 (in this order) are placed in the QC array. For node 5 no candidate resistor is found. For node 4 the 7221 ohm resistor is found. Thus, this resistor element is removed. For node 1 the 9840 ohm resistor is found. Thus, this resistor element is removed. After that, node 7 becomes a dangling internal node, which is connected only to the 116.1 ohm resistor.

The existing problem for the above network can however easy be resolved by doing function reducArtDegree again after reducMinRes. Because node 7 can be eliminated (res_cnt < min_degree). See figure below.



Now, we can again use reducArtDegree to eliminate node 1. We get the following resistors: 772.4 (3 - 4), 909.7 (3 - 5) and 145.2 (4 - 5). Note that we can not eliminate node 4, because it has node "names". Note that when reducMinRes removes the 42.4 ohm resistor, then also node 4 can be eliminated.

3. NODE REDUCTION CONCLUSIONS

To forecome the problem to get a dangling internal node with only one resistor, we must invoke again reducArtDegree for each change in the resistor network. This needs to be done for all resistor heuristics used in function reducGroupI. The elimination of a dangling internal node is only possible when the node does not have the "term" status. For an area node, also parameter min_degree and min_art_degree must not be too small.

Note that for the SNE method, internal nodes can get the "term" status and are therefor not eliminated. When such a node becomes dangling we can not easy eliminate that node. We can only possible lower the "term" status when the node does not have "names". Besides that, such a node must have capacitors. Possible the removal of couple capacitors can also lead to a dangling node in a ready group.

A dangling node with only one resistor can also not be eliminated if it gets the "keep" status. In that case it is not really dangling, because it must have capacitors.

When reducArtDegree rescanning is used, more nodes are possible eliminated in an early stage of the reduction proces. After that some other reduction functions are possible not more needed and that gives maybe some reduction speed-up.

When a resistor is removed from the resistor network, we know that the articulation degree is possibly changed. In some cases we don’t need to remove the resistor, but we can eliminate the node instead. And this can give a more accurate reduction.

When you add a terminal or label, its position must be chosen with care, because it can easy introduce an extra vertical tile split. And this give another resistivity mesh and after reduction other resistor values.

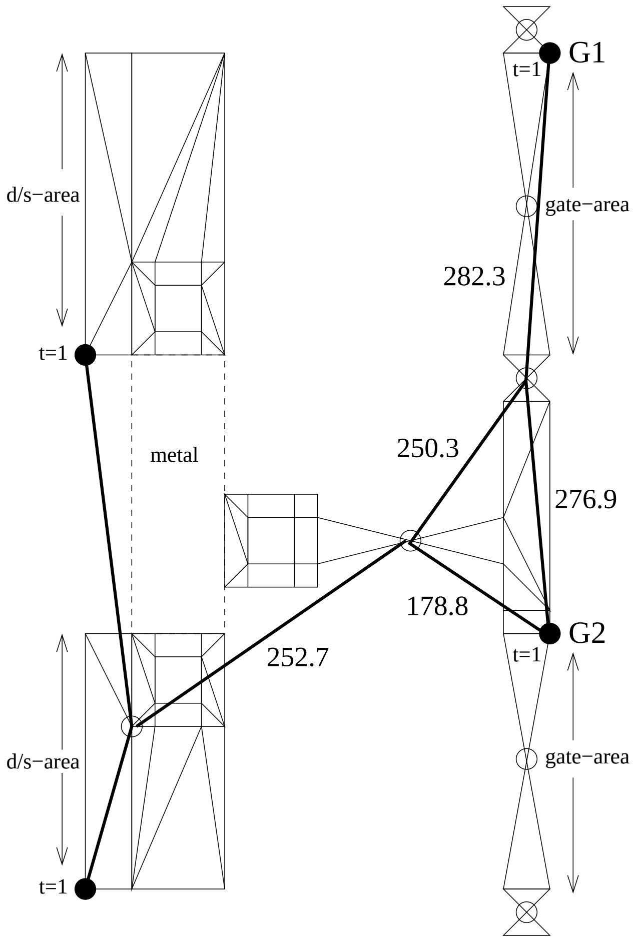
Depended of the choice by reducMinRes the final resistor network can be different. Because of the use of the "keep" flag (i don’t know why this is necessary) a small resistor between two "keep" nodes can not be removed. The min_sep_res heuristic does not use this "keep" status and can be used to remove this small resistor (reducMinSepRes).

NOTE

I have added parameter **delete_dangling** to eliminate the dangling node problem.
This parameter is default "off".

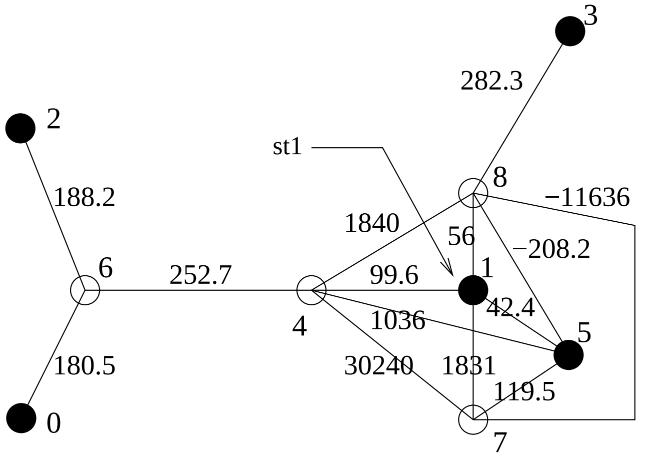
4. ABOUT GATE AREA NODE ASSIGNMENT

Always the top-right node of a transistor gate area is flagged as "term=1". In the figure below you see that this leads to an asymmetric resistor mesh. Thus there is much more resistance to G1 and too less to G2. Better (if possible) the center position of the gate-area should be taken for the gate node.

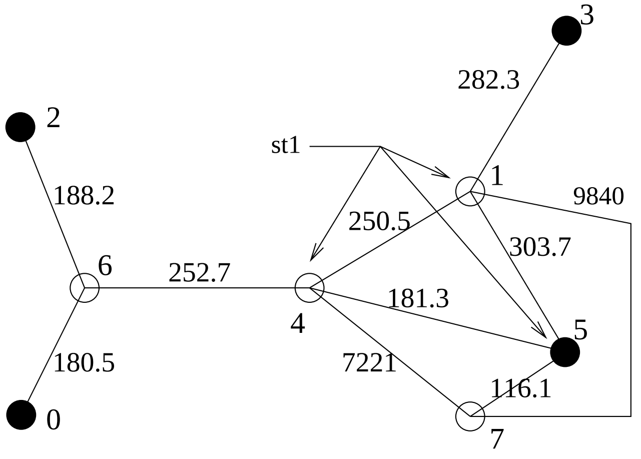


5. reducMinRes vs. reducMinSepRes

We compare functions reducMinRes and reducMinSepRes starting with following figure:



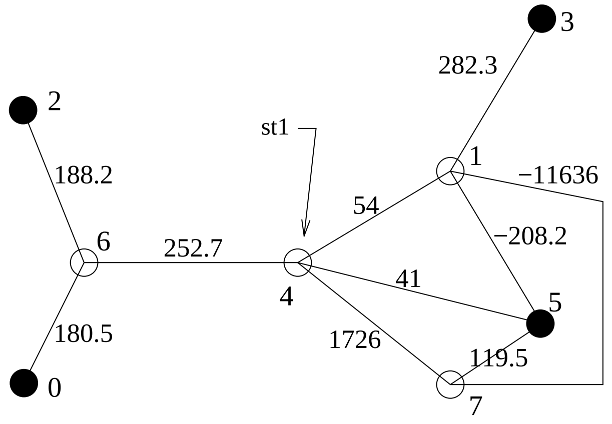
When parameter "min_res" is used, function reducMinRes shall try to reduce too small resistors connected to nodes with non-keep status (that are only the terminal nodes 0, 1, 2, 3 and 5). For node 1 (terminal st1) three resistors less than 100 Ohm are found. Thus, node 1 can possibly be joined with one of the nodes 4, 5 or 8. After that node 1 is eliminated. The resulting network (see below) is in all cases the same, no matter which choice is made. The only difference is, that the node name "st1" can be given to node 4, 5 or 8.



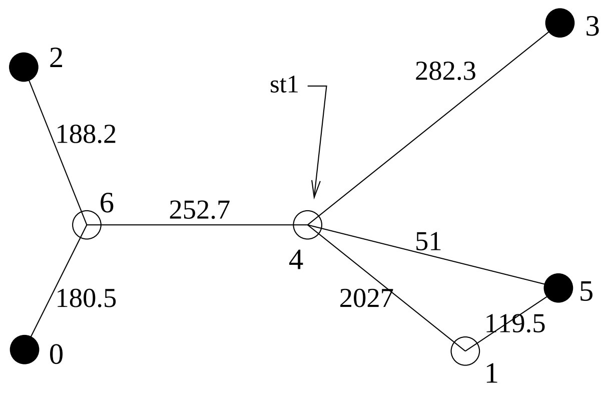
When we want, that st1 is joined with node 5, then we must change the code of function reducMinRes. Because the node with lowest resistor value is the best candidate.
Note: In the result above we see that function reducMaxParRes needs to be done.

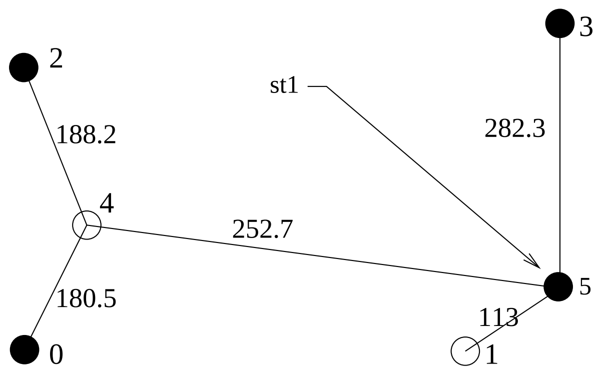
When we use function reducMinSepRes in place of function reducMinRes, what is the difference? Set parameter "min_sep_res" to 100 and reset "min_res" to zero.

Function reducMinSepRes works different, it shall examine all nodes (not only the non-keep ones). Besides that it shall join two nodes using function nodeJoin, which does not do a Gaussian node elimination of the "st1" node. Thus the network results are different, we start with nodeJoin of nodes 1 and 4, see result:



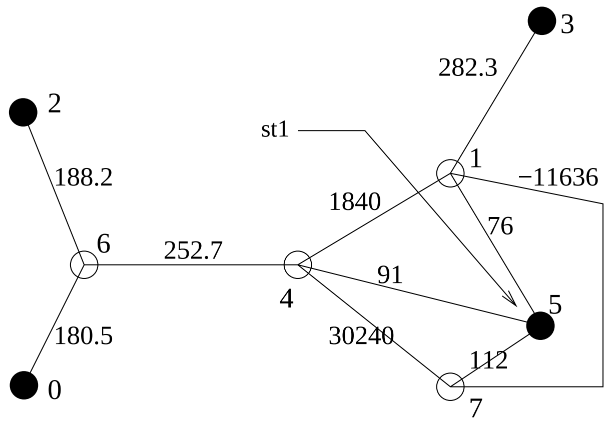
Because of the nodeJoin reduction, the other 2 low resistors are still in the network. Thus next nodeJoin shall join nodes 1 and 4 and following nodeJoin shall join nodes 4 and 5. This gives a dangling node 1 as result:

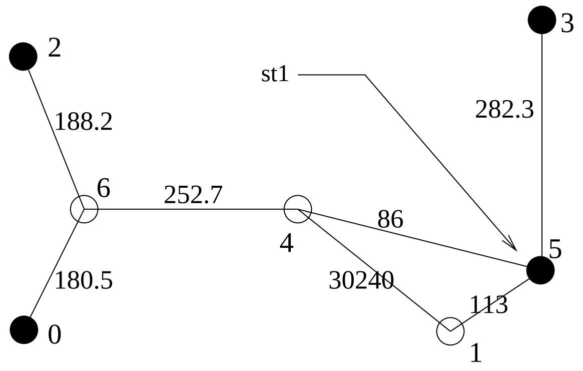


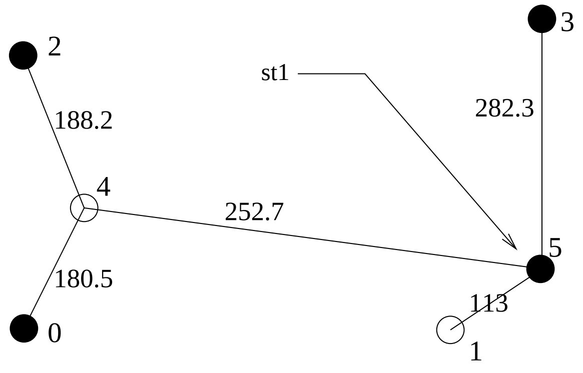


In the result above we see that function reducMaxParRes needs **not** to be done anymore.

If we start the reduction with the lowest resistor by function reducMinSepRes, we see what happens below:







We see that the same result is achieved. We get a dangling node, which can be removed, because it is a local node (set parameter **delete_dangling** to "on").

Note that the combination reducMinRes and reducMaxParRes can give almost the same result as function reducMinSepRes and that reducMaxParRes can give the same dangling node. When you want to use the lowest resistor by function reducMinRes, set parameter **lowest_min_res** to "on" (default: off).