UPDATE OF MAKESIZE PROGRAM

S. de Graaf

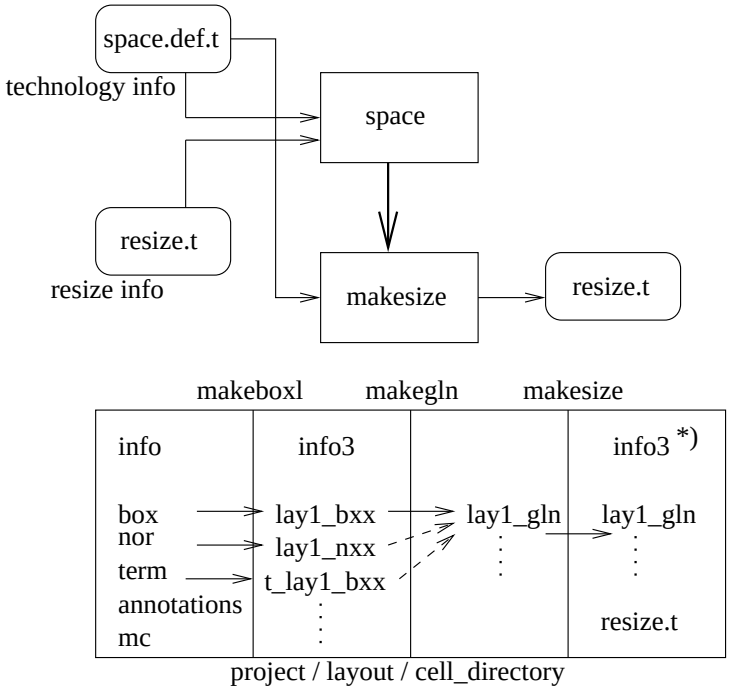
Circuits and Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
Delft University of Technology
The Netherlands

Report EWI-ENS 04-11
December 15, 2004

Last revision: April 20, 2005.

1. INTRODUCTION

The *makesize* program is the resize tool of the *space* program. The *space* program uses it, when resizes are needed. See figure below.



When preprocessing is not turned off, *space* shall look of cell expansion is needed. For example, there are no gln-files or the gln-files are out of date. In that case expand&resize is needed. However, when the gln-files look up to date, it can be that resizes are needed. When there is no "resize.t" file, only resize needs to be done. When there is a "resize.t" file, the resizes can be different, in that case expand&resize needs to be done. It is also possible, that there is a "resize.t" file, but no resizes are needed, in that case expand needs to be done again. Note that *makegln* shall remove an existing "resize.t" file.*

The *makesize* program is updated, because now users can grow and shrink new masks. Thus, more gln-files are created. And the gln-files may also contain negative masks. And besides that, it is now possible to specify conditional growing or shrinking.

---

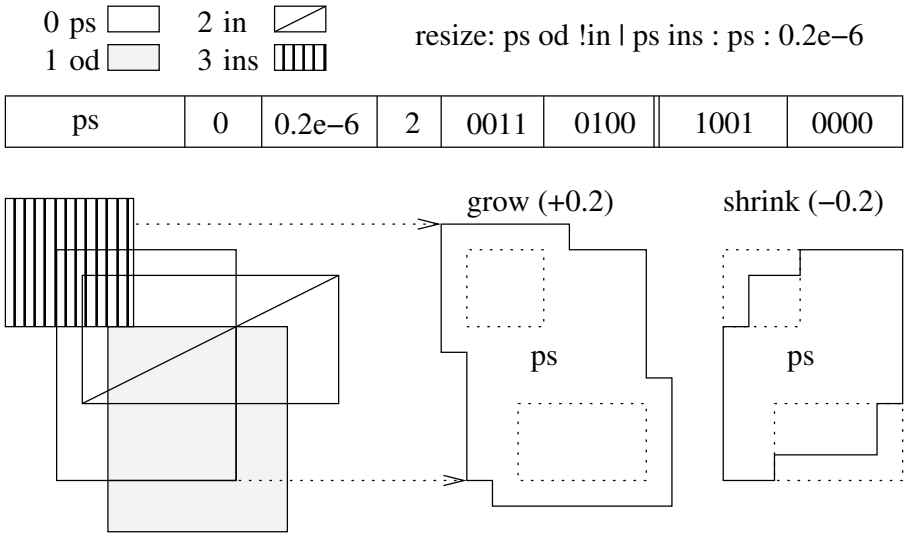
* Note that *makesize* shall update the cell expansion bounding box in stream "info3", when a grow is performed.

2. HOW MAKESIZE WORKS

A resize specification for a mask or new mask contains the following data:

maskname	id	value	cNr	cPresent $_1$	cAbsent $_1$	·····

The "maskname" is the name of the gln-file, which must be created. The "id" is the mask number (≥ 0) of this mask in the technology-file mask table. The "value" is the grow-value (if ≥ 0) or shrink-value (if < 0), which must be used. This value is given in meters. The "cNr" field specifies the number of "cPresent" and "cAbsent" conditions (must be ≥ 1). The "cPresent" and "cAbsent" fields are bit-masks. The first low order bit is used for mask id 0 and the second bit for mask id 1, etc. Thus, it is not equal to the color bit-masks of the technology-file. For example:



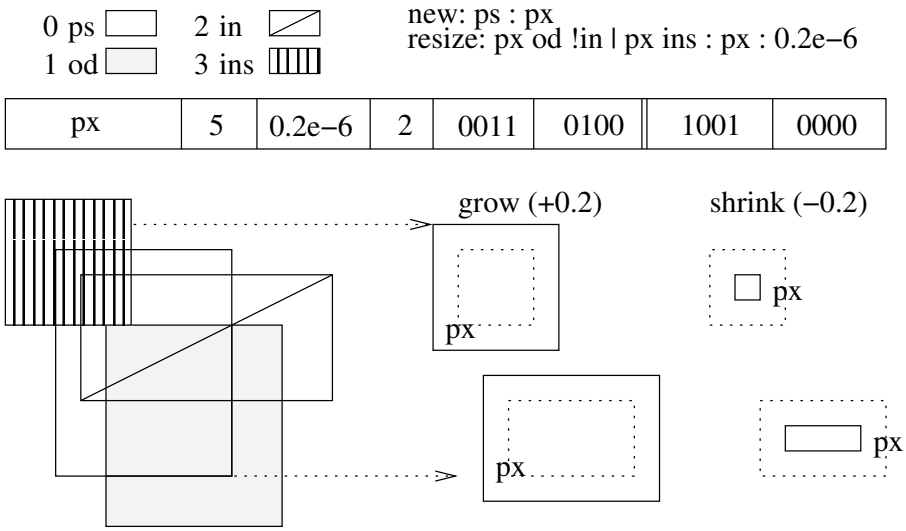
Normally, the "id" of the mask (which is created) is also present in all "cPresent" fields. When the "id" is not present, then it is the "id" of a completely new mask or of an existing mask which is overwritten.

Growing or shrinking conditional a present mask works a little bit different, than growing/shrinking conditional a new to create mask. In the first case, the parts of the present mask, which are not conditional growed/shrinked, are outputted unchanged.

Note that the old *makesize* program worked only with one "cPresent" bit-mask and expected a zero "cAbsent" bit-mask. And the "cPresent" bit-mask was only containing the mask "id". Thus, the complete mask was growed or shrinked. When other mask bits were present/absent, also these masks were completely growed/shrinked. Thus, conditional growing/shrinking was not possible.

Note that a shrink with extra conditions is a two pass operation. For the example on previous page, first the negative mask "ps" is growed, and the inverse of this is outputted. In second pass, the part "ps !(od !in | ins)" is merged with the output tiles of first pass.

For a new mask "px" is the result of the example on previous page as follows:

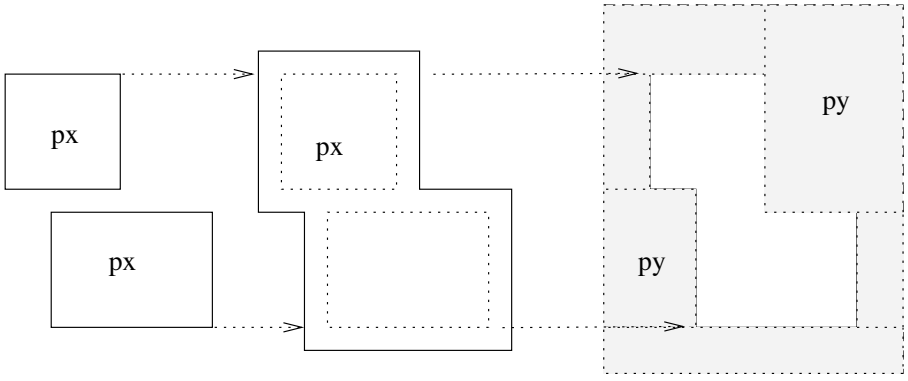


px	5	0.2e−6	2	0011	0100	1001	0000

Because "px" was not a mask before, "px" in the resize condition-list is changed into "ps". Thus, the "px" mask with id 5 (≥ procdata→maskno) is not present in the condition-list. A second resize of "px", however, is using the created "px_gln" file, and a third resize is using the negative mask of "px". For example:

resize: px : px : 0.2e−6 new: !px : py resize: py : py : 0.2e−6

px	5	0.2e−6	1	100000	000000
py	6	0.2e−6	1	000000	100000

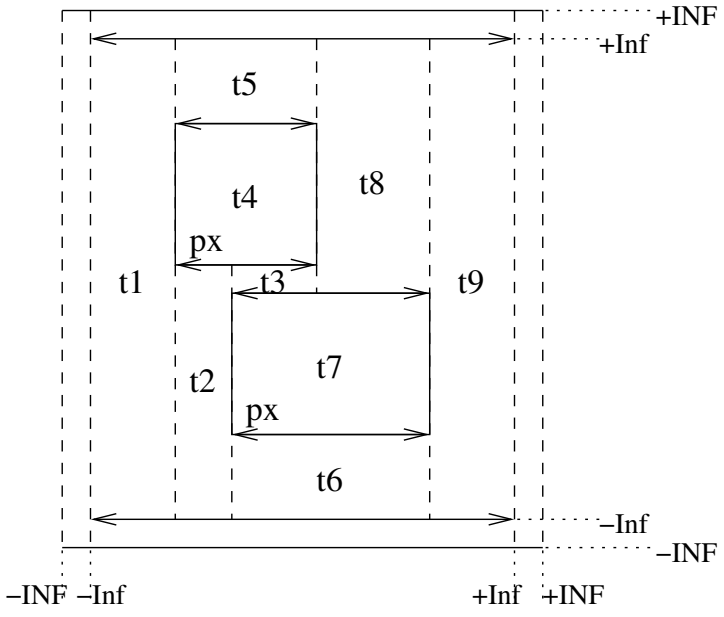


Thus, a gln-file can represent a negative mask. Such a gln-file starts with two infinity edges. The first edge starts on xl=-INF and yl=-INF, and the second edge on xl=-INF and yl=+INF. I have modified *space* and *makesize* and some other tools (*mplot* and *makefem*) to handle this special edges correctly. Here follows a dump of the "py_gln" file:

```
% dbcat -s py_gln t4
dbcat: -s: read only dbstream: py_gln
=> layout/t4/py_gln
-2147483647 2147483647 -2147483647 -2147483647
-2147483647 2147483647 2147483647 2147483647
   20    32    56    56
   20    56    92    92
   32    80    16    16
   56    80    52    52
```

The *tecc* program must also make an arangement for *space* in the technology file, that the hasSubmask flag is put on. Only in that case *space* is using two special edges internally and can it easy add the color bit of the negative gln-file mask to these special edges, and can it skip the two special edges of the gln-file.

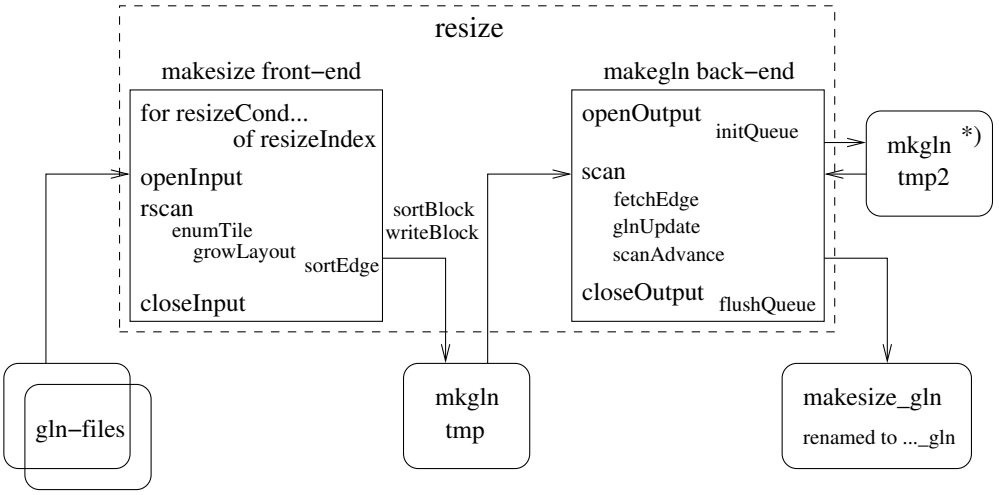
I have modified the *makesize* program code, thus it starts working internally always with two special edges. This makes it easier to work with negative masks, because these special edges get the opened negative gln-file bits. For example, the following edges and tiles are evaluated for a grow of mask "px":

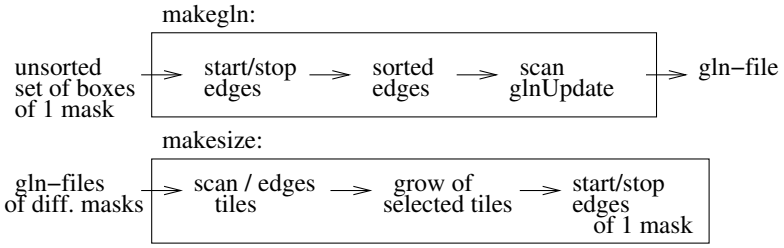


Only the tiles t4 and t7 have a color and are growed. The special edges laying on yl=-Inf and yl=+Inf are always removed in the output section. On the other hand, the *makesize* program adds two special edges (-INF/+INF), when a negative gln-file must be produced.

3. MAKESIZE WORKING SCHEME

The schematic picture below tries to explain how the *makesize* program works. It uses as back-end the sort edge and output gln part of the *makegln* program. Below the picture is given a schematic explanation for the working of the parts.





* Note that for the "mkgln" tmp2 file standard a compress/uncompress tool is used.