SPACE SHAPE ELEMENTS APPLICATION NOTE

S. de Graaf

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

Report ET-CAS 02-03
July 23, 2002

Last revision: December 9, 2003.

1. INTRODUCTION

To fully understand how to specify mask conditions for edge shape and cross-over shape elements in a technology file, i have written this note. See also the "Space 3D Capacitance Extraction User's Manual" [1].

1.1 Edge Shape Elements

Syntax:

```
eshapes:
  name : condition_list(s) : mask : dxb dxt
    .
    .
```

An edge shape element specification in the edge shape list, specifies for a conductor the extension in the x (or y) direction relative to the position of the original conductor edge in the layout.

The first value (*dxb*) specifies the extension of the bottom and the second value (*dxt*) specifies the extension of the top. Either extension may be negative.

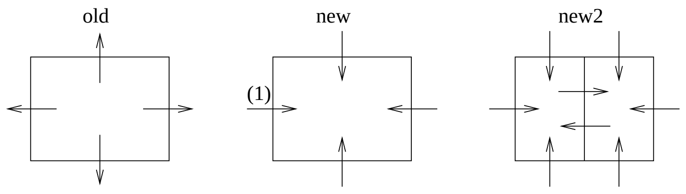
The following example uses three methods to specify the same metal1_eshape:

Example:

```
eshapes :
    metal1_eshape_old : in !-in : in : 0.2 0.1
    metal1_eshape_new : !in -in : in : 0.2 0.1
    metal1_eshape_new2:     -in : in : 0.2 0.1
```

Note that *space* can only use one specification for a conductor. Thus, you must make a choice which one you want to use. The "metal1_eshape_old" specification was used in the User's Manual. But it is not the best choice, because this edge element is also matched as a surface element with the condition "in".

In the new User's Manual is now used the "metal1_eshape_new" specification. This specifies really an edge element and is not matched as a surface element. Either, to use this specification, the *space* program had to be modified (see figure below).



By "metal1_eshape_new" the edge element was only found by transition (1), when the left tile is not an infinity tile. Besides that, the left tile is not the conductor tile. Thus, the assignment must in this case be done to the right tile.

Note that the figure also shows, what happens, when the edge shape is specified with "metal1_eshape_new2". The condition of that specification matches all metal1 edges (also internal edges). This is not really a problem for *space*, because *space* knows that on both sides the tile is a conductor. Thus, *space* can skip this two edge elements found.

Note that it is maybe elegant to use this short specification. But, because the condition not exactly specifies the edge shape element, the element is matched more times. I think that this gives no execution overhead. But it can also be optimized by *space* (or *tecc),* if needed.

Example:

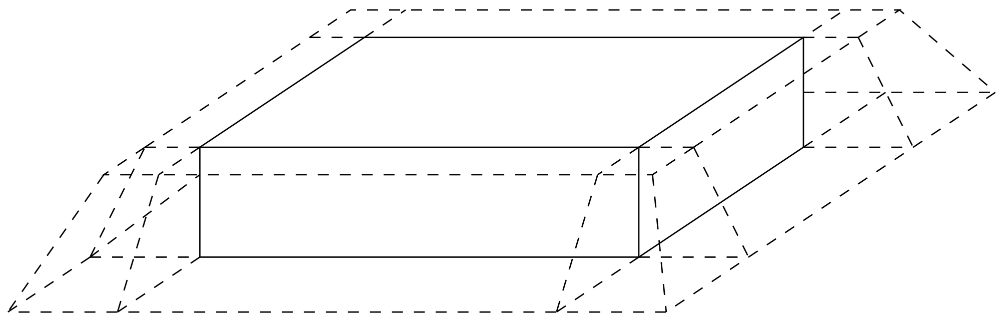
```
unit vdimension 1e-6
unit shape      1e-6

conductors:
    me1_cond : cmf : cmf : 0

vdimensions:
    me1_vdim : cmf : cmf : 25 0.02

eshapes:
    me1_esh  : !cmf -cmf : cmf : 0.02 0.01
```

Resulting metal1 layout mesh:



1.2 Cross-over Shape Elements

Syntax:

```
cshapes :
```
> *name : condition_list(s) : mask : xb1 xt1 xb2 xt2*
> .
> .

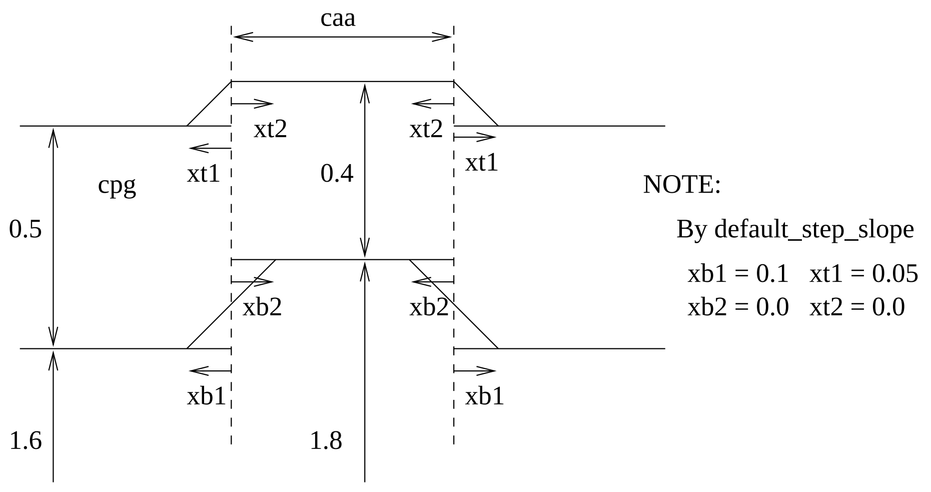
The cross-over shape list specifies for different conductors, the extensions of the bottom and top of each conductor in the x (or y) direction relative to the position of a transition edge in the layout. The transition edge, is an edge (caused by another mask), where a conductor goes from one bottom and thickness specification to another bottom and/or thickness specification. Thus, there must be two vdimension specifications for the same conductor. The cross-over shape specification overrules the *default_step_slope* method.

Note that the first and third value (*xb1* and *xb2*) specify the extension of the bottom of the conductor (at left and right side of the transition, when edge mask is on the right side) and the second and fourth value (*xt1* and *xt2*) specify the extension of the top of the conductor. Each extension value may be negative. The cross-over shape list should be present in the element definition file after the edge shape list.

Example:

```
vdimensions :
    ver_cpg_of_caa : cpg !caa  : cpg : 1.6 0.5
    ver_cpg_on_caa : cpg  caa  : cpg : 1.8 0.4

cshapes :
    cpg_cshape : cpg !caa -caa : cpg : 0.1 0.1 0.1 0.0
```



Warning: Some extension combinations can result in an incorrect mesh.

1.3 Cross-over Shape Example

Technology specification (in micron):

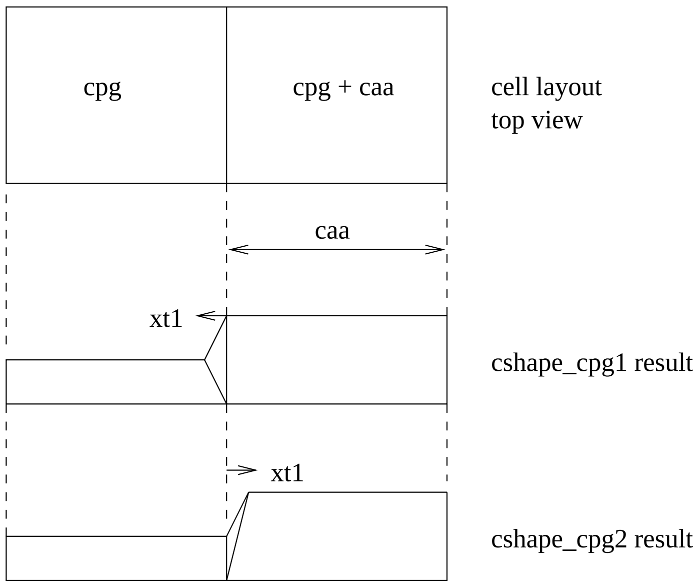
Example:

```
unit vdimension 1e-6
unit shape      1e-6

vdimensions:
    ver_cpg_of_caa : cpg !caa  : cpg : 25 0.01
    ver_cpg_on_caa : cpg  caa  : cpg : 25 0.02

cshapes:
    cshape_cpg1 : cpg !caa -caa : cpg : 0 0.005 0 0
    cshape_cpg2 : cpg caa !-caa : cpg : 0 0.005 0 0
```

Cell layout top view and mesh side views (lambda = 0.01 micron):



As can be seen is the result for the "cshape_cpg2" specification different, because for this not recommended specification everything is inverted. Thus, the value xt1 becomes xt2.

Note that the "cshape_cpg1" result is the default, which is get without any cshape specification (the "default_step_slope" parameter value is 2.0).

Note that the mesh coordinates can be printed with a ":debug" file with as contents the name of the "mpgreen.c" file.

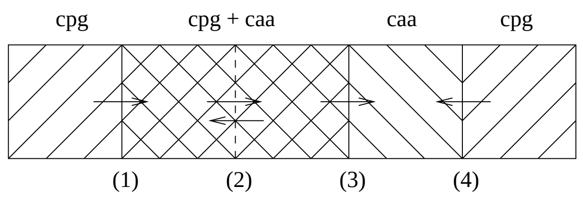
1.4 Warning Messages

A not fully specified condition for a cshape element can result in warning messages. It depends on the specification you choice:

Example:

```
cshapes :
    cpg_cshape2 : cpg      -caa      : cpg : 0.1 0.1 0.1 0.0
    cpg_cshape3 : cpg !caa -caa      : cpg : 0.1 0.1 0.1 0.0
    cpg_cshape4 : cpg !caa -caa -cpg : cpg : 0.1 0.1 0.1 0.0
```

The "cpg_cshape2" gives not so much warnings, because it is only matched with a RecogE(tile_l,tile_r) call. The tile_l must contain "cpg" and tile_r must contain "caa" (or vis a versa).



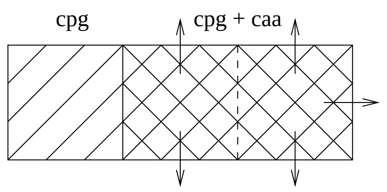
In the figure, transition (1) gives the correct match. Transition (2) gives a double match, but the cshapes are skipped, because on both sides is the same vdimension for "cpg". This situation is not matched for "cpg_cshape3" and "cpg_cshape4". Note that *space* does not more give a warning for this situation. Transitions (3) and (4) give also a match, but the cshape is skipped, because not on both sides is "cpg". Note that *space* gives a warning for these situations. These situations are not matched for "cpg_cshape4". Thus, only the mask conditions of "cpg_cshape4" specify fully the cshape for conductor "cpg".

On the other hand, the following (not recommended) cshape specification gives a lot of warnings:

Example:

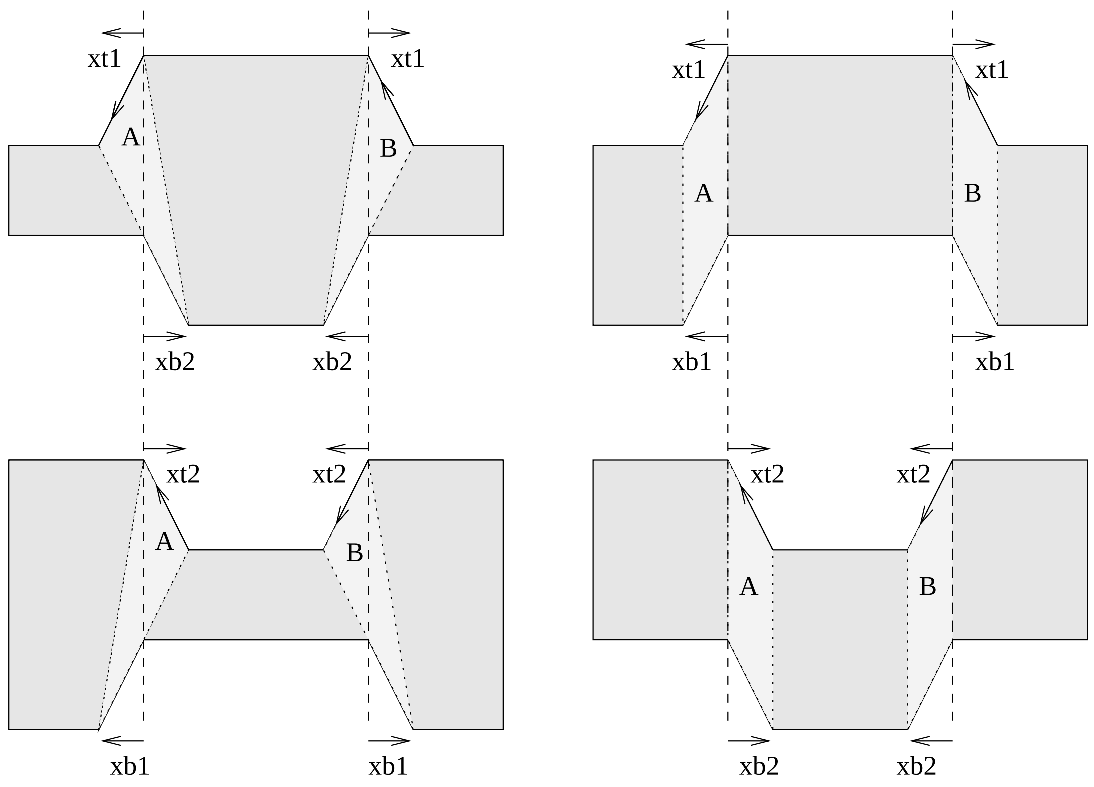
```
cshapes :
    cpg_cshape5 : cpg caa !-caa : cpg : 0.1 0.1 0.1 0.0
```

Note that *space* can be modified to skip these cshape matches and warnings.
The warnings ("not on both sides cpg") are given on the following transitions:



1.5 Cross-over Shape Defaults

The defaults of the bottom and top shape are given in the figure below. Note that this is maybe not what you have in mind. Therefor you can specify your own cshape.



1.6 Cross-over Shape Problems

Note that the above face polygons A and B are always created. Thus, when you define your own cshape or set a too small "default_step_slope" parameter, you can create incorrect face polygons. You can also get polygons which have no area.

References

1. A.J. van Genderen and N.P. van der Meijs, ‘‘Space 3D Capacitance Extraction User’s Manual,’’ Report ET-NT 94.37, Delft University of Technology, Network Theory Section, Delft, the Netherlands (July 2002).

CONTENTS