Space Cap3d
Example

S. de Graaf

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

Report EWI-ENS 08-03
Feb 26, 2008

Last revision: Apr 16, 2008.

1. INTRODUCTION

This application note describes a small 3D capacitance extraction example. It describes some debug parameters of the *space3d* extractor, which can be used to get a more inside knowledge what is calculated.
This application note describes also the new "cap3d.sensitivity_data" parameter.

As a starting point the demo **poly5** example directory is used. See the shell script "script.sh". We must first change the directory into a project directory:

```
% mkpr -p scmos_n -l 0.05 .
```

We use a lambda of 0.05, that means that one design unit is 0.05 micron meter. We create a very simple layout cell "poly2" using the LDM language and create the layout cell in the project:

```
% cat poly2.ldm
ms poly2
term cpg  5 15 0 50 a
term cpg 25 35 0 50 b
me
% cldm poly2.ldm
```

With the *dbcat* program you can show how this data is stored in the project database (we don't list the empty streams):

```
% dbcat -v poly2
dbcat: -v: verbose mode
=> layout/poly2/info
cell/call/elmt  bxl,bxr,byb,byt:
    5    35     0    50
    0     0     0     0
    5    35     0    50
=> layout/poly2/term
term: lay: xl,xr,yb,yt: [bxl,bxr,byb,byt: dx,nx: dy,ny:]
a          cpg     5    15     0    50
b          cpg    25    35     0    50
```

Note that the *space3d* program internally multiplies all coordinate values by four. For we extract we must compile the technology file which contains:

```
unit vdimension  1e-6  # micron
conductors :
    resP : cpg : cpg : 0.0

vdimensions :
    dimP : cpg : cpg : 0.5 0.5

dielectrics :
    SiO2   3.9   0.0
    air    1.0   5.0
```

The first dielectric layer (SiO2) has an epsilon of 3.9 and is 5 micron thick. The bottom

of the conductor mask (cpg) is on 0.5 micron height and is 0.5 micron thick. We use the *tecc* program to compile "tech.s" into the "tech.t" format (we list not all output):

```
% tecc tech.s
   ...
   no. of elem.:  2 (2)
```

The used space parameter file "param.p" contains the following setup:

```
BEGIN cap3d
be_mode           0c
max_be_area       0.5
be_window         1
END cap3d
```

Now we can perform a cap3d extraction (we list not all output):

```
% space3d -v -C3 -E tech.t -P param.p poly2
Version 5.3.2, compiled on Tue, 26 Feb 2008 09:50:24
See http://www.space.tudelft.nl
parameter file: param.p
technology file: tech.t
preprocessing poly2 (phase 1 - flattening layout)
preprocessing poly2 (phase 2 - removing overlap)
extracting poly2
strip 5 35 (add)
Schur dimension 36, maxorder 17

extraction statistics for layout poly2:
        capacitances        : 3
        resistances         : 0
        nodes               : 3
        mos transistors     : 0

space3d: --- Finished ---
```

The following circuit is extracted:

```
% xspice -a poly2
poly2

* Generated by: xspice 2.44 30-Nov-2007
* Date: 27-Feb-08 11:42:50
* Path: /u/01/01/simon/cacd/share/demo/poly5
* Language: SPICE

* circuit poly2 a b
c1 b a 129.4118e-18
c2 b GND 358.6599e-18
c3 a GND 358.6599e-18
* end poly2
```

You see that there are three capacitances, two ground caps and one couple cap between the terminals "a" and "b".

2. MORE INSIDE INFORMATION

Ok, this is nice, but how can we get more inside information of what is done? First of all, see the "Space 3D Capacitance Extraction User's Manual".

The display tool *Xspace* can show the 3D boundary element mesh what is used and can show between which points a Green value is calculated. To do a cap3d extraction using *Xspace* give the following command (i left out the verbose option):

```
% Xspace -C3 -E tech.t -P param.p poly2
```

Note that *Xspace* runs *space3d* with some special options to get display data. Note that *Xspace* uses the technology file to get mask color info. See also the "Xspace User's Manual". When you have started *Xspace* (see above), you must set some display options before you extract. Touch the "Display" menu button and click on the "DrawBEMesh" item to display the mesh. To see a 3D mesh, click also on the "3 dimensional" item. To see the points between which the Green values are calculated, click on the "DrawGreen" item. To start the extraction, use hotkey 'e' (or click on "extract" in the "Extract" menu). What happened? The output was drawn very fast and you have probably misted the white Green lines! You can give *Xspace* a drawing speed limit by clicking in the right most (empty?) menu field. You don't need to extract again, but you want only show the display data again. Use hotkey 'a' to start the display again. You can also enter single step mode with hotkey 's' and step using the 's' key. You can go back to again mode using the 'a' key. You can quit the *Xspace* program by using the 'q' key. By the way, you can rotate the mesh with the arrow keys of the alternate keypad and you can switch from 3D to 2D view with the '*' key.

What do you see? You see a mesh (red lines). You see that the each conductor rail is split in four parts. Thus, each conductor rail has 18 faces. You see that the white Green lines are drawn between the center points of the faces of the mesh. You see that the be_window is to small, because the white lines are not drawn between all faces. Thus, if you want to calculate all Green values in one big matrix you need a bigger be_window. How big? I think that a cap3d.be_window of 3 micron is big enough. Note that this type mesh and the usage of center face points is only used by the piecewise constant methods (cap3d.be_mode 0c and 0g). The piecewise linear method uses a triangular mesh and uses the edges of the mesh to calculate the Green values. Ok, we use another be_window and see what happens. Use the **-S** option to change this parameter on the command line, type:

```
% Xspace -C3 -E tech.t -P param.p poly2 -Scap3d.be_window=3
```

Before extract, don't forget to set the display options again. Note that the extracted cap values are a little bit changed.

```
% xspice -a poly2
poly2
* circuit poly2 a b
c1 b a 132.2249e-18
c2 b GND 356.7857e-18
c3 a GND 356.7857e-18
* end poly2
```

The Schur dimension of the matrix was 36 (use the verbose space option to get this info). This dimension is for me a little to big to explain some special debug parameters. Thus, we are going the reduce the mesh till the minimum mesh. The area of one rail is a 0.5 by 2.5 micron (1.25 micronˆ2). Thus, if we change the cap3d.max_be_area to 2 micronˆ2, this must work. We shall make a new parameter file "param2.p" and put the new be_window and max_be_area in it. Do an extraction again and see the result:

```
% space3d -C3 -E tech.t -P param2.p poly2 -v
    ...
strip 5 35 (add)
Schur dimension 12, maxorder 11
    ...
```

Yes, it works! And how looks now the circuit?

```
% xspice -a poly2
poly2
* circuit poly2 a b
c1 b a 131.775e-18
c2 b GND 347.3718e-18
c3 a GND 347.3718e-18
* end poly2
```

Now we can show a number of debug parameters. To see all the mesh points (spiders) and edges type:

```
% space3d -C3 -E tech.t -P param2.p poly2 -Sdebug.print_spider
space3d: Debugging output to ‘spiderDebug’
% cat spiderDebug
stripB -220 -100, stripL 20 stripR 140 stripA 140
spider xyz 140 0 80 act 140 0 80
spider xyz 140 0 40 act 140 0 40
    ...
spideredge 140 0 80 <-> 140 200 80 f1: 137341992 f2:137346448
```

To see all green values, type:

```
% space3d -C3 -E tech.t -P param2.p poly2 -Sdebug.print_green
space3d: Debugging output to ‘spiderDebug’
% cat spiderDebug
stripB -220 -100, stripL 20 stripR 140 stripA 140
green 120 0 60 120 0 60 2.31072e+09
green 120 0 60 40 0 60 1.65192e+08
    ...
green 40 200 60 40 200 60 2.31072e+09
```

To see the Schur input and output matrix, type:

```
% space3d -C3 -E tech.t -P param2.p poly2 -Sdebug.print_matrix
space3d: Debugging output to ‘spiderDebug’
% cat spiderDebug
stripB -220 -100, stripL 20 stripR 140 stripA 140
row_in 0 11 2.310719e+09 1.651918e+08 1.643581e+08 1.814430e+08 1.420921e+08 ...
row_in 1 10 2.310719e+09 5.177703e+07 7.871098e+07 5.181586e+07 8.911910e+07 ...
row_in 2 9 7.371622e+08 3.798230e+08 3.185070e+08 2.564405e+08 1.140500e+08 ...
row_in 3 8 7.884100e+08 2.486012e+08 3.798230e+08 1.818322e+08 1.539436e+08 ...
row_in 4 7 6.566544e+08 3.185070e+08 1.347281e+08 9.385042e+07 7.605951e+07 ...
row_in 5 6 7.371622e+08 2.564405e+08 1.818322e+08 1.347281e+08 1.140500e+08 ...
row_in 6 5 7.371622e+08 3.798230e+08 3.185070e+08 2.564405e+08 8.911910e+07 ...
row_in 7 4 7.884100e+08 2.486012e+08 3.798230e+08 7.871098e+07 1.814430e+08
row_in 8 3 6.566544e+08 3.185070e+08 5.181586e+07 1.420921e+08
row_in 9 2 7.371622e+08 5.177703e+07 1.643581e+08
row_in 10 1 2.310719e+09 1.651918e+08
row_in 11 0 2.310719e+09
row_out 0 11 4.469416e-10 -2.587133e-11 -4.408452e-11 -4.803751e-11 -3.569790e-11 ...
row_out 1 10 4.469416e-10 -1.478727e-12 -8.484625e-12 -1.047996e-12 -5.844804e-12 ...
row_out 2 9 2.067049e-09 -7.507867e-10 -6.932227e-10 -1.537551e-11 1.111893e-11 ...
row_out 3 8 2.034983e-09 -1.124479e-11 -7.106845e-10 -7.567188e-11 -9.532756e-11 ...
row_out 4 7 2.218384e-09 -6.778380e-10 -4.553192e-11 1.324026e-11 -3.007495e-11 ...
row_out 5 6 2.203382e-09 -3.981970e-10 -7.567188e-11 -4.553192e-11 1.111893e-11 ...
row_out 6 5 2.203382e-09 -7.106845e-10 -6.778380e-10 -1.537551e-11 -5.844804e-12 ...
row_out 7 4 2.034983e-09 -1.124479e-11 -7.507867e-10 -8.484625e-12 -4.803751e-11
row_out 8 3 2.218384e-09 -6.932227e-10 -1.047996e-12 -3.569790e-11
row_out 9 2 2.067049e-09 -1.478727e-12 -4.408452e-11
row_out 10 1 4.469416e-10 -2.587133e-11
row_out 11 0 4.469416e-10
```

Other, maybe useful, space parameters are:

```
debug.server_matrix
print_coef
debug.print_cap3d_init
debug.print_green_init
debug.print_green_terms
```

The "debug.server_matrix" parameter shows only the "row_in" of "debug.print_matrix".
The "print_coef" parameter shows a matrix of the Schur module.
Use the "debug.print_schur" parameter to get more inside information about Schur calls.
Note that the output is going directly to "stderr".

The new "cap3d.sensitivity_data" parameter is added for Yu Bi. This parameter generates a Matlab file. The following *space3d* run shows the output of this data file.

```
% space3d -C3 -E tech.t -P param2.p poly2 -Scap3d.sensitivity_data
space3d: Sensitivity output to ‘poly2_sens.m’
% cat poly2_sens.m

% the schur_out matrix (dimension 12)
C = [ 4.469416e-10 -2.587133e-11 -4.408452e-11 -4.803751e-11 -3.569790e-11 ...
 0 4.469416e-10 -1.478727e-12 -8.484625e-12 -1.047996e-12 -5.844804e-12 ...
 0 0 2.067049e-09 -7.507867e-10 -6.932227e-10 -1.537551e-11 1.111893e-11 ...
 0 0 0 2.034983e-09 -1.124479e-11 -7.106845e-10 -7.567188e-11 -9.532756e-11 ...
 0 0 0 0 2.218384e-09 -6.778380e-10 -4.553192e-11 1.324026e-11 -3.007495e-11 ...
 0 0 0 0 0 2.203382e-09 -3.981970e-10 -7.567188e-11 -4.553192e-11 1.111893e-11 ...
 0 0 0 0 0 0 2.203382e-09 -7.106845e-10 -6.778380e-10 -1.537551e-11 -5.844804e-12 ...
 0 0 0 0 0 0 0 2.034983e-09 -1.124479e-11 -7.507867e-10 -8.484625e-12 -4.803751e-11;
 0 0 0 0 0 0 0 0 2.218384e-09 -6.932227e-10 -1.047996e-12 -3.569790e-11;
 0 0 0 0 0 0 0 0 0 2.067049e-09 -1.478727e-12 -4.408452e-11;
 0 0 0 0 0 0 0 0 0 0 4.469416e-10 -2.587133e-11;
 0 0 0 0 0 0 0 0 0 0 0 4.469416e-10];

% the incidence matrix (conductors 2)
I = [ 1 0;
      0 1;
      1 0;
      1 0;
      1 0;
      1 0;
      0 1;
      0 1;
      0 1;
      0 1;
      1 0;
      0 1];

% the incidence matrix sidewall flags
B = [ 1 1 1 0 0 1 1 0 0 1 1 1];

% the vector of the area of panels (microns^2)
A = [ 0.25 0.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 0.25 0.25];
```