SPACE EXAMPLES

Nick van der Meijs, Simon de Graaf

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

Report ET-ENS 2006.01

Last revision: June 29, 2006

Contents

1 Introduction

This document explains something about the space demo examples. Trying these demo's can be a good starting point to learn more about the Space System.

Before you can use the demo's, you must know the space software installation path. When the space software is installed in your home directory, then the installation path (or `ICDPATH`) is `$HOME/cacd`. But you can also rename 'cacd' into something else, for example 'cacd_new'. The only thing you need to do, is to set your `PATH` environment variable to the `ICDPATH`. See also the file `$ICDPATH/Installation.txt` for more information. Give for example the following commands to set your `PATH`:

```
% setenv ICDPATH $HOME/cacd
% setenv PATH $ICDPATH/share/bin:$PATH
```

Go now to the demo directory, for example type:

```
% cd $ICDPATH/share/demo
% ls
attenua  invert      ny9t    README    sub3term   switchbox
crand    multiplier  poly5   sram      suboscil
```

Each demo contains a `README` file, which explains what you must do. When you don't have a private software installation or don't have write permission, you must copy the demo files to a private directory to do the demo. For example:

```
% mkdir mydemo ; cd mydemo
% cp $ICDPATH/share/demo/attenua/* .
% cat README
```

When you don't want to type the space commands yourself, you can use a shell script to do it for you. In that case, type:

```
    % ./script.sh
or
    % sh script.sh
```

Read the following chapters, to know more about the demo examples (if included). We wish you success. We hope that you appreciate our software and want to use it. If you have any questions or comments, they are welcome.

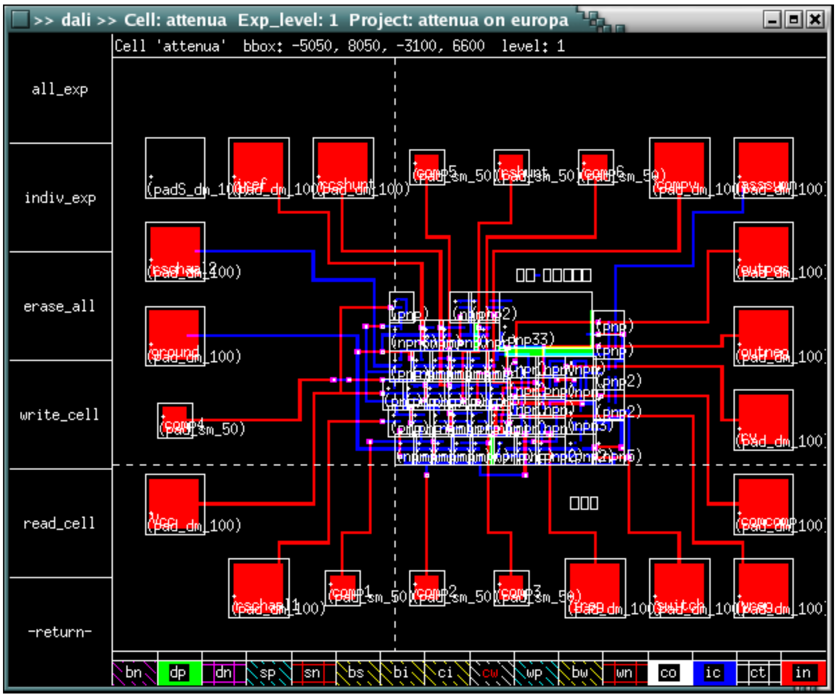
```
The SPACE team.
```

2 attenua Example of Bipolar Extraction

2.1 Introduction

This example gives a demonstration of an attenuator circuit in bipolar DIMES-01 process. See also: A. van Staveren and A.H.M. van Roermund, ”Low-voltage Low-power Controlled Attenuator for Hearing Aids”, Electronic Letters, 22 July 1993, Vol. 29, No. 15, pp. 1355-1356.

The layout looks as follows, using the layout editor `dali` (see `icdman dali`):



2.2 Files

This tutorial is located in the directory `$ICDPATH/share/demo/attenua`. Initially, it contains the following files:

`README:`	A file containing information about the demo.
`attenua.cmd:`	Command file for a PSPICE simulation.
`attenua.gds:`	GDS2 file of the layout of the design.
`att_ref.spc:`	SPICE file of the reference circuit.
`npnBW.dev:`	Device file containing npnBW device.

`pnpWP.dev:`	Device file containing pnpWP device.
`script.sh:`	Batch file for running all commands of the demo in sequence.
`spice3f3.lib:`	Library file for circuit listing.
`xspicerc:`	Init file for circuit listing.

2.3 Running the Extractor

First, use the following command to change the current working directory '.' into a project directory:

```
% mkpr -p dimes01 -l 0.1 .
```

The command specifies the `dimes01` process from the technology library and a lambda (design unit) of $0.1\mu m$. We use the mask names as defined in the `maskdata` file of the library. And we use the default technology file `space.def.s` and parameter file `space.def.p` of the library.

Second, the layout description is put into the project database. The layout is supplied in a GDS2 file, which is converted into an internal database format with the `cgi` program:

```
% cgi attenua.gds
```

Now, using verbose mode, perform a flat extraction of cell "attenua".

```
% space -vF attenua
```

```
Version 5.3.1, compiled on Fri Feb 03 12:45:53 GMT 2006
See http://www.space.tudelft.nl
parameter file: $ICDPATH/share/lib/process/dimes01/space.def.p
technology file: $ICDPATH/share/lib/process/dimes01/space.def.t
preprocessing attenua (phase 1 - flattening layout)
preprocessing attenua (phase 2 - removing overlap)
extracting attenua

extraction statistics for layout attenua:
        capacitances        : 0
        resistances         : 0
        nodes               : 31
        mos transistors     : 0
        bipolar vertical    : 39
        bipolar lateral     : 59
        substrate nodes     : 0

overall resource utilization:
        memory allocation  : 0.408 Mbyte
        user time          :         0.0
        system time        :         0.0
        real time          :         2.0   1%

space: --- Finished ---
```

You can try out the options `-c`, `-C`, `-r` and `-z` for capacitance and resistance extraction.

The extracted circuit can be inspected using the `xspice` program. We use the `-a` option to get alpha-numeric node names, type:

```
% xspice -a attenua
```

This gives the following output (for an extraction w/o cap and res):

```
attenua

* circuit attenua iref comp4 comp1 comp2 comp3 comp6 rshunt comp5 ireg vreg
*                 switch asssymm rschaal2 rschaal1 rcshunt rv ground outneg
*                 Vcc outpos compv comcomp
q1 comp4 1 Vcc wp102c
q2 comp2 1 rschaal1 wp102c
q3 comp5 1 rschaal2 wp102c
q4 1 1 Vcc wp102c
q5 comp2 comp1 ground bw101a
q6 1 iref ground bw101a
   ...
q9 2 2 Vcc wp102c
q10 comp5 comp4 ground bw101a
q11 comp1 1 Vcc wp102c
q12 comp1 comp1 ground bw101a
q13 iref iref ground bw101a
q14 rshunt 2 Vcc wp102c
q15 comp4 comp4 ground bw101a
q16 2 3 ground bw101a
q17 comp1 comp1 ground bw101a
q18 rcshunt rshunt Vcc wp102c
q19 comp4 comp4 ground bw101a
q20 3 3 ground bw101a
q21 switch iref ground bw101a
q22 comp1 comp3 ground bw101a
q23 rcshunt iref ground bw101a
    ...
q26 comp4 comp6 ground bw101a
q27 comp6 rcshunt Vcc wp102c
q28 comp3 comp2 ground bw101a
q29 3 4 Vcc wp102c
q30 rshunt iref ground bw101a
q31 compv switch Vcc wp102c
q32 compv switch Vcc wp102c
q33 comp6 comp5 ground bw101a
q34 4 comcomp ground bw101a
q35 comp3 rcshunt Vcc wp102c
q36 8 8 Vcc wp102c
q37 ireg 8 Vcc wp102c
q38 outneg comp2 vreg bw101a
q39 comcomp comp2 vreg bw101a
q40 comcomp comp5 vreg bw101a
q41 outpos comp5 vreg bw101a
q42 5 switch Vcc wp102c
q43 5 switch Vcc wp102c
q44 5 5 ireg bw101a
q45 compv 5 rv bw101a
q46 6 compv Vcc wp102c
q47 Vcc 6 rv bw101a
q48 switch switch Vcc wp102c
```

```
     ...
q81 7 switch Vcc wp102c
q82 7 switch Vcc wp102c
q83 7 7 ground bw101a
     ...
q92 8 7 ground bw101a
q93 comcomp switch Vcc wp102c
q94 comcomp switch Vcc wp102c
q95 9 switch Vcc wp102c
q96 9 switch Vcc wp102c
q97 outneg asssymm Vcc wp102c
q98 outpos asssymm Vcc wp102c
* end attenua

.model wp102c pnp( ... )
.model bw101a npn( ... )
```

If you have PSPICE, you can perform a circuit simulation (after customizing the script `nspice`) as follows (see `icdman nspice`):

```
% nspice attenua attenua.cmd
```

2.4 Running a Circuit Comparison

You can compare the extracted circuit against a reference circuit with the circuit comparison program `match`.

First, add device descriptions for the bipolar transistors to the database so that the reference circuit can be stored into the database:

```
% putdevmod pnpWP.dev npnBW.dev
% xcontrol -device pnpWP npnBW
```

Second, add the reference circuit description for the ”attenua” circuit to the database using the program `cspice`:

```
% cspice att_ref.spc
```

Now, compare this reference circuit with the extracted circuit, type:

```
% match att_ref attenua
```

```
match: Succeeded.
```

The above result shows that the circuits are identical. Note that you can use the `-bindings` option to get more information.

You can also try to change the layout of the ”attenua”. For example, use the layout editor `dali` to make an error by deleting some connection. Before you run the compare tool `match` again, extract the circuit of cell ”attenua” again, type:

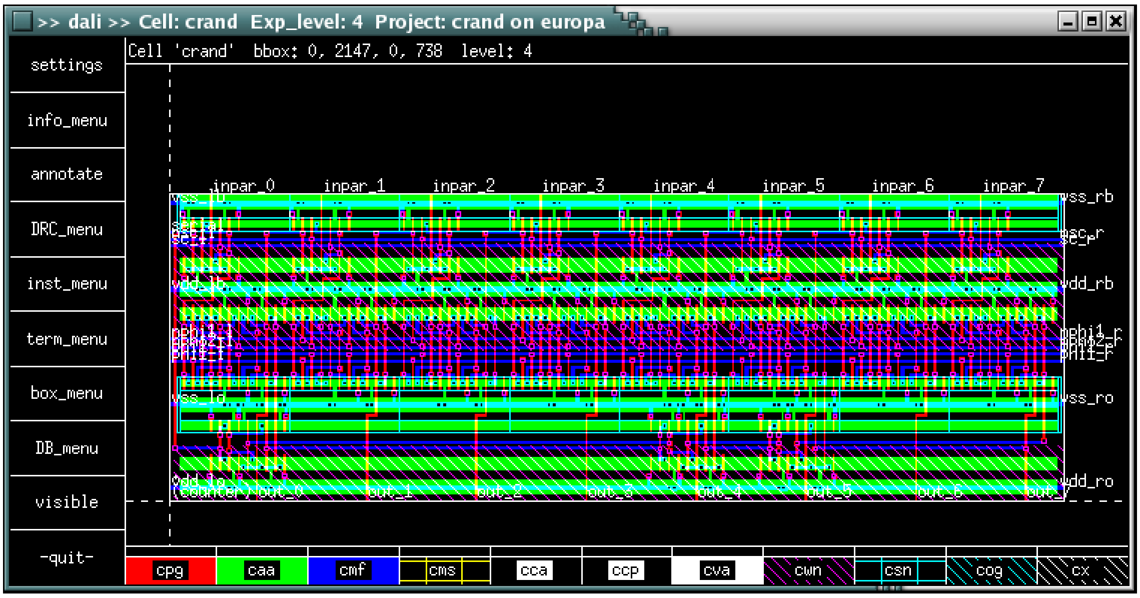
```
% space -F attenua
% match att_ref attenua
```

3 crand Example of Extraction/Switch-Level Simulation

3.1 Introduction

In this example, we will be studying a random counter circuit. We will see how Space is used for circuit extraction. And how you can do a switch-level simulation of the circuit.

The layout looks as follows, using the layout editor `dali` (see `icdman dali`):



3.2 Files

This tutorial is located in the directory `$ICDPATH/share/demo/crand`. Initially, it contains the following files:

`README:`	A file containing information about the demo.
`crand.cmd:`	Command file for circuit simulation.
`crand.gds:`	GDS2 file of the layout of the crand design.
`script.sh:`	Batch file for running all commands of the demo in sequence.

3.3 Running the Extractor

First, use the following command to change the current working directory '.' into a project directory:

```
% mkpr -p scmos_n -l 0.2 .
```

The command specifies the `scmos_n` process from the technology library and a lambda (design unit) of $0.2\mu m$. We use the mask names as defined in the `maskdata` file of the library. And we are using the default technology file `space.def.s` and parameter file `space.def.p` of the library.

```
% cgi crand.gds
```

Now, we can extract a circuit description for the layout of the `crand` cell, as follows:

```
% space -vFc crand
```

```
Version 5.3.1, compiled on Fri Feb 03 12:45:53 GMT 2006
See http://www.space.tudelft.nl
parameter file: $ICDPATH/share/lib/process/scmos_n/space.def.p
technology file: $ICDPATH/share/lib/process/scmos_n/space.def.t
preprocessing crand (phase 1 - flattening layout)
preprocessing crand (phase 2 - removing overlap)
extracting crand

extraction statistics for layout crand:
        capacitances        : 221
        resistances         : 0
        nodes               : 222
        mos transistors     : 419
        bipolar vertical    : 0
        bipolar lateral     : 0
        substrate nodes     : 0

overall resource utilization:
        memory allocation  : 0.287 Mbyte
        user time          :         0.0
        system time        :         0.0
        real time          :         1.5   5%

space: --- Finished ---
```

You can show the resulting circuit with one of the circuit listing tools. For example, to list the circuit in a SLS description, use `xsls` (see `icdman`).

```
% xsls crand
```

The output is default going to ”stdout”, a part is shown below:

```
   ...
network crand (terminal out_7, out_6, out_5, out_4, out_3, out_2, out_1, out_0,
               inpar_7, inpar_6, inpar_5, inpar_4, inpar_3, inpar_2, inpar_1,
               inpar_0, serial, vss_lb, vss_lo, sc_l, nsc_l, vdd_lb, vdd_lo,
               nphi1_l, phi1_l, nphi2_l, phi2_l, phi1_r, phi2_r, nphi2_r,
               nphi1_r, vss_ro, vss_rb, vdd_rb, nsc_r, sc_r, vdd_ro)
{
    net {vdd_lo, vdd_ro};
    net {phi1_l, phi1_r};
    net {phi2_l, phi2_r};
    net {nphi2_l, nphi2_r};
    net {nphi1_l, nphi1_r};
```

```
    net {sc_l, sc_r};
    net {vdd_lb, vdd_rb};
    net {nsc_l, nsc_r};
    net {SUBSTR, vss_lb};
    net {SUBSTR, vss_rb};
    net {SUBSTR, vss_ro};
    net {SUBSTR, vss_lo};
    cap 2.8f (1, GND);
    nenh w=4u l=1.2u (14, 1, 14);
    cap 11.44f (2, GND);
    penh w=6.8u l=1.2u (14, 2, 14);
    cap 11.44f (3, GND);
    penh w=6.8u l=1.2u (vdd_lb, 3, 12);
    cap 2.8f (4, GND);
    nenh w=4u l=1.2u (SUBSTR, 4, SUBSTR);
    nenh w=4u l=1.2u (phi1_l, 13, 14);
    penh w=6.8u l=1.2u (nphi1_l, 13, 14);
    penh w=6.8u l=1.2u (serial, 12, vdd_lb);
    nenh w=4u l=1.2u (serial, 5, SUBSTR);
    nenh w=4u l=1.2u (phi2_l, 10, 13);
    penh w=6.8u l=1.2u (nphi2_l, 10, 13);
    penh w=6.8u l=1.2u (nsc_l, 12, vdd_lb);
    cap 3.2f (5, GND);
    nenh w=4u l=1.2u (nsc_l, 5, 14);
    cap 72.16f (6, GND);
    penh w=6.8u l=1.2u (vdd_lo, 6, 8);
    cap 16f (7, GND);
    nenh w=4u l=1.2u (SUBSTR, 7, 9);
    nenh w=4u l=1.2u (10, 10, 10);
    penh w=6.8u l=1.2u (10, 10, 10);
    penh w=6.8u l=1.2u (inpar_0, 12, 14);
    nenh w=4u l=1.2u (inpar_0, 11, 14);
    nenh w=7.2u l=1.2u (inpar_0, SUBSTR, SUBSTR);
    ...
    ...
}
```

3.4 Running the Switch-Level Simulation

For this simulation you are using the switch-level simulator `sls`. See the ”SLS: Switch-Level Simulator User’s Manual” and for the manual page `icdman sls`. This simulator is started from the simulation GUI `simeye` and the results are shown in the output window (see `icdman simeye`).

First, start the simulation GUI `simeye`.

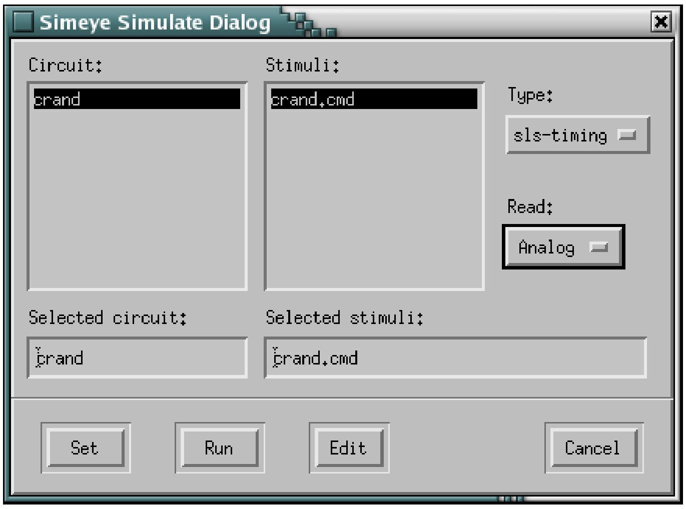
```
% simeye
```

Second, prepare the simulation:

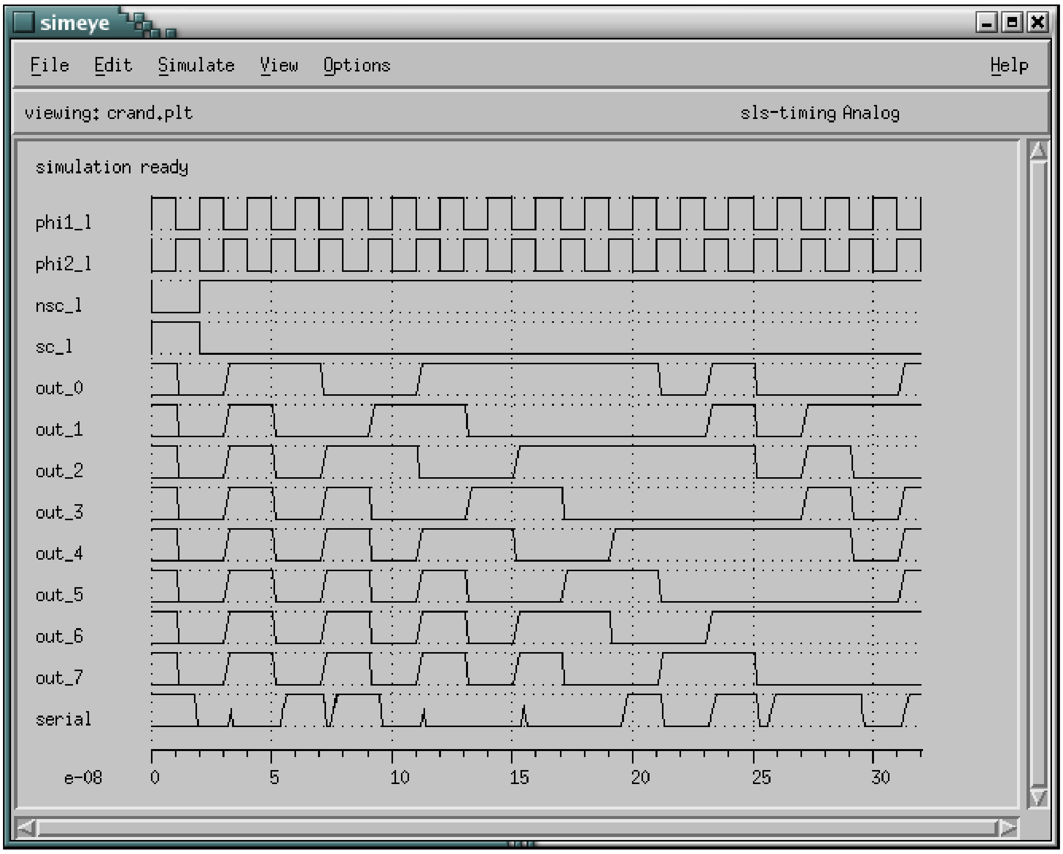
Click on the ”Simulate” menu and choice the ”Prepare” item. Select in the ”Circuit:” field cell name ”crand” and in the ”Stimuli:” field file name ”crand.cmd” (click on it). To inspect or edit the input signals, click on the ”Edit” button.

Third, start the switch-level simulation:

Go back to the "Simulate" menu and choice the "Prepare" dialog item again:



In the dialog window, choice simulation "Type: sls-timing" and for "Read: Analog". Now, start the switch-level timing simulation by clicking on the "Run" button and wait for simulation results. Below, you see the output waveforms.



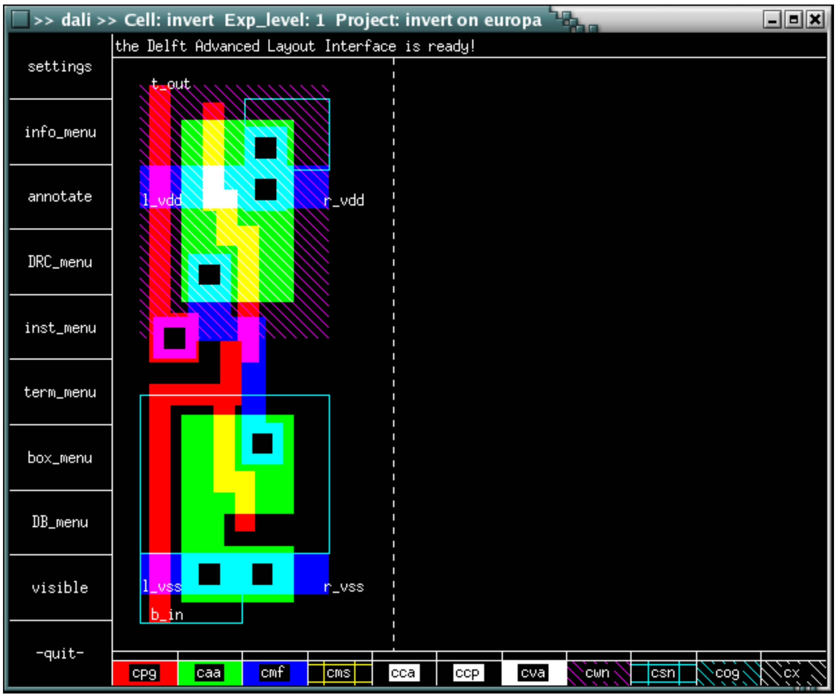
Note, to exit `simeye`, go to the "File" menu and click on "Exit" and "Yes".

4 invert Example of Junction Capacitance Extraction

4.1 Introduction

In this example, we demonstrate the extraction of an inverter with transistor bulk connections and drain/source regions. The latter are extracted as either non-linear junction capacitances, or as drain/source area and perimeter information attached to the MOS transistors. See also the ”Space Tutorial” sections ”3.8”, ”4” and ”5”.

The layout looks as follows, using the layout editor `dali` (see `icdman dali`):



4.2 Files

This tutorial is located in the directory `$ICDPATH/share/demo/invert`. Initially, it contains the following files:

`README:`	A file containing information about the demo.
`invert.cmd:`	Command file for circuit simulation.
`invert.gds:`	GDS2 file of the layout of the design.

`jun.lib`: Model library file for circuit listing.

`script.sh`: Batch file for running all commands of the demo in sequence.

`xspicerc`: Init file for circuit listing.

4.3 Running the Extractor

First, we need a project directory. To change the current working directory '.' into a project directory, type:

```
% mkpr -p scmos_n -l 0.2 .
```

We use the `scmos_n` process from the technology library and a lambda of $0.2\mu m$. We are using the default technology file `space.def.s` and parameter file `space.def.p` of the library.

Second, the layout description is put into the project database. The layout is supplied in a GDS2 file, which can be converted to internal database format with the `cgi` program:

```
% cgi invert.gds
```

Now, we can perform an extraction of the "invert" cell. Use option `-C` to include coupling capacitances. The drain/source regions are extracted as non-linear junction capacitances that have parameters 'area' and 'perimeter' (using option `-S`). The following command is used:

```
% space -C -Sjun_caps=area-perimeter invert
```

Look to the listing of file "xspicerc" to see how the junction capacitances are printed in the netlist output as diodes with parameters '`area`' and '`pj`':

```
listing of file xspicerc
include_library spice3f3.lib
include_library jun.lib

model nenh_0 nenh nmos ()
model penh_0 penh pmos ()
model ndif  ndif  d ()
model nwell nwell d ()
model pdif  pdif  d ()

bulk nmos 0.0
bulk pmos 5.0

params ndif  { area=$area pj=$perim }
params nwell { area=$area pj=$perim }
params pdif  { area=$area pj=$perim }
```

Next, retrieve a SPICE circuit description. Use the `xspice` command with options `-a` and `-u` to exstract SPICE from the database:

```
% xspice -au invert
```

```
invert

* circuit invert b_in t_out l_vdd r_vdd l_vss r_vss
vnet1 l_vdd r_vdd 0
vnet2 l_vss r_vss 0
```

```
m1 t_out b_in l_vss l_vss nenh_0 w=6.8u l=1.2u
m2 l_vdd b_in t_out l_vdd penh_0 w=12.4u l=1.2u
c1 l_vdd b_in 795.12e-18
c2 l_vdd t_out 3.04892f
d1 t_out l_vdd pdif area=20.48p pj=14.8u
c3 l_vdd GND 249.6e-18
d2 GND l_vdd nwell area=155.52p pj=50.4u
c4 b_in t_out 495.12e-18
c5 b_in GND 3.21264f
c6 b_in l_vss 282.72e-18
c7 t_out GND 987.68e-18
d3 GND t_out ndif area=16.16p pj=11.2u
c8 l_vss GND 774.4e-18
d4 GND l_vss ndif area=27.92p pj=22.6u
* end invert

.model nenh_0 nmos(level=2 ...)
.model penh_0 pmos(level=2 ...)
.model pdif d(is=10u cjo=500u vj=800m m=500m)
.model nwell d(is=2u cjo=100u vj=800m m=500m)
.model ndif d(is=2u cjo=100u vj=800m m=500m)
```

An alternative is to extract the drain/source regions as ’area’ and ’perimeter’ information that is attached to the MOS transistors. This is achieved by modifying the transistor definitions in the element definition file. Therefore, first copy the element definition file from the process directory to the local file ”elem.s”:

```
% cp $ICDPATH/share/lib/process/scmos_n/space.def.s elem.s
```

```
listing of file elem.s
    ...
    conductors :
    # name    : condition     : mask : resistivity : type
      cond_mf : cmf           : cmf  : 0.045       : m   # first metal
      cond_ms : cms           : cms  : 0.030       : m   # second metal
      cond_pg : cpg           : cpg  : 40          : m   # poly interconnect
      cond_pa : caa !cpg !csn : caa  : 70          : p   # p+ active area
      cond_na : caa !cpg  csn : caa  : 50          : n   # n+ active area
      cond_well : cwn         : cwn  : 0           : n   # n well

    fets :
    # name : condition    : gate d/s : bulk
      nenh : cpg caa  csn : cpg  caa : @sub  # nenh MOS
      penh : cpg caa !csn : cpg  caa : cwn   # penh MOS
    ...
    junction capacitances ndif :
    # name    :  condition                 : mask1 mask2 : capacitivity
      acap_na :  caa       !cpg  csn !cwn  : @gnd  caa : 100  # n+ bottom
      ecap_na : !caa -caa !-cpg -csn !-cwn : @gnd -caa : 300  # n+ sidewall

    junction capacitances nwell :
      acap_cw :  cwn                  : @gnd  cwn : 100  # bottom
      ecap_cw : !cwn -cwn             : @gnd -cwn : 800  # sidewall
```

```
74
75  junction capacitances pdif :
76    acap_pa :  caa       !cpg  !csn cwn      :  caa cwn : 500 # p+ bottom
77    ecap_pa : !caa -caa !-cpg !-csn cwn -cwn : -caa cwn : 600 # p+ sidewall
    ...
```

Change in line 46 and in line 47 the resistance value to zero (conductor elements `cond_pa` and `cond_na`). This, to avoid complaints by `tecc` about the fact the resistance for drain/source regions should be zero (the resistance is already modeled in the transistor model).

Change line 52 and line 53 of the fets definitions, into:

```
52    nenh : cpg caa  csn : cpg caa (!cpg caa csn) : @sub # nenh MOS
53    penh : cpg caa !csn : cpg caa (!cpg caa !csn): cwn  # penh MOS
```

This, to extract area/perimeter information of the drain/source regions. And remove the `ndif` and `pdif` junction capacitance lists, lines (68,69) and lines (76,77).

Next, run the technology compiler `tecc` to compile the new element definition file:

```
% tecc elem.s
```

Now, extract again, using the compiled element definition file:

```
% space -C -Sjun_caps=area-perimeter -E elem.t invert
```

And watch the resulting SPICE output again:

```
% xspice -au invert
```

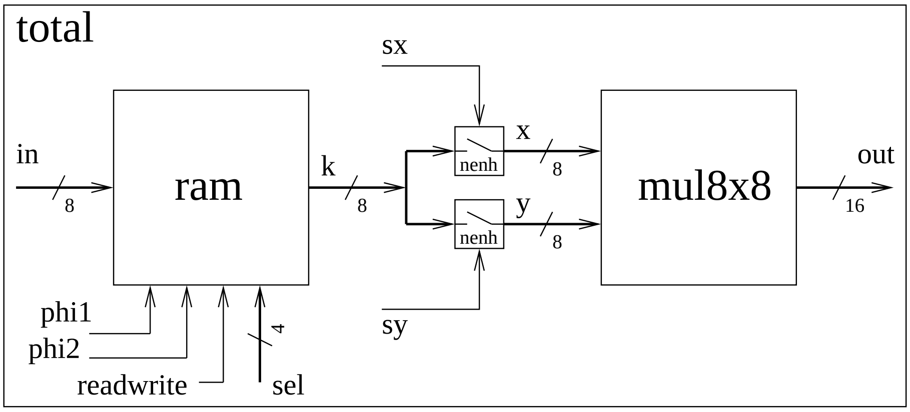
```
invert

* circuit invert b_in t_out l_vdd r_vdd l_vss r_vss
vnet1 l_vdd r_vdd 0
vnet2 l_vss r_vss 0
m1 t_out b_in l_vss l_vss nenh_0 w=6.8u l=1.2u ad=16.16p as=27.92p pd=11.2u
+  ps=22.6u nrs=0.603806 nrd=0.349481
m2 l_vdd b_in t_out l_vdd penh_0 w=12.4u l=1.2u ad=23.36p as=20.48p pd=10.8u
+  ps=14.8u nrs=0.133195 nrd=0.151925
c1 l_vdd b_in 795.12e-18
c2 l_vdd t_out 3.04892f
c3 l_vdd GND 249.6e-18
d1 GND l_vdd nwell area=155.52p pj=50.4u
c4 b_in t_out 495.12e-18
c5 b_in GND 3.21264f
c6 b_in l_vss 282.72e-18
c7 t_out GND 987.68e-18
c8 l_vss GND 774.4e-18
* end invert

.model nenh_0 nmos(level=2 ...)
.model penh_0 pmos(level=2 ...)
.model nwell d(is=2u cjo=100u vj=800m m=500m)
```

5 multiplier Example of Functional/Switch-Level Simulation

5.1 Introduction

This example demonstrates how you can do a combined functional and switch-level simulation. Read also the following documents: ”SLS: Switch-Level Simulator User’s Manual” and ”Functional Simulation User’s Manual”.

In this example a network ’total’ is simulated that has a ram and a multiplier that are described at the functional level. Pass transistors are used to demultiplex the output signals of the ram and to multiply two subsequent words that come out of the ram. The circuit looks as follows:



5.2 Files

This tutorial is located in the directory `$ICDPATH/share/demo/multiplier`. Initially, it contains the following files:

`README:`	A file containing information about the demo.
`mul8x8.fun:`	Functional description of the mul8x8 block.
`ram.fun:`	Functional description of the ram block.
`total.cmd:`	Command file for the circuit simulation.
`total.sls:`	SLS description of the total network.
`script.sh:`	Batch file for running all commands of the demo in sequence.

5.3 Compiling the Descriptions

To compile the descriptions and to simulate, we need to work in a project directory. First, we change the current working directory '.' into a project directory:

```
% mkpr -p scmos_n .
```

We use the "scmos_n" technology and a default lambda value.

To compile the function blocks, we use the `cfun` program. To add the functional descriptions to the project database, type the following commands:

```
% cfun ram.fun
% cfun mul8x8.fun
```

To compile the SLS network description, we use the `csls` program. Type the following command to add network `total` to the database:

```
% csls total.sls
```

To get a hierarchical cell listing (of the circuit tree) of the project database, use the `dblist` command:

```
% dblist -h
```

```
circuit:
1 - total           (18)
    2 - ram               1 (function)
    2 - mul8x8            1 (function)
```

You see 2 hierarchical levels. The top cell "total" has 2 sub cells (function blocks).

5.4 Running the Switch-Level Simulation

For this simulation you are using the switch-level simulator `sls`. See the "SLS: Switch-Level Simulator User's Manual" and for the manual page `icdman sls`. This simulator is started from the simulation GUI `simeye` and the results are shown in the output window (see `icdman simeye`).

First, start the simulation GUI `simeye`.

```
% simeye
```

Second, prepare the simulation: Click on the "Simulate" menu and choice the "Prepare" item. Select in the "Circuit:" field cell name "total" and in the "Stimuli:" field file name "total.cmd" (click on it).

Third, start the switch-level simulation: Click on the "Run" button and wait for the simulation results.

After that, watch the results: Zoom-in onto the lowest 3 signals, choice the "ZoomIn" item of the "View" menu. Click in the results window and move the mouse to specify a zooming area and click again. Now, choice "Measure" from the "View" menu, and click on the right mouse button to watch the integer values of the multiplied signals.

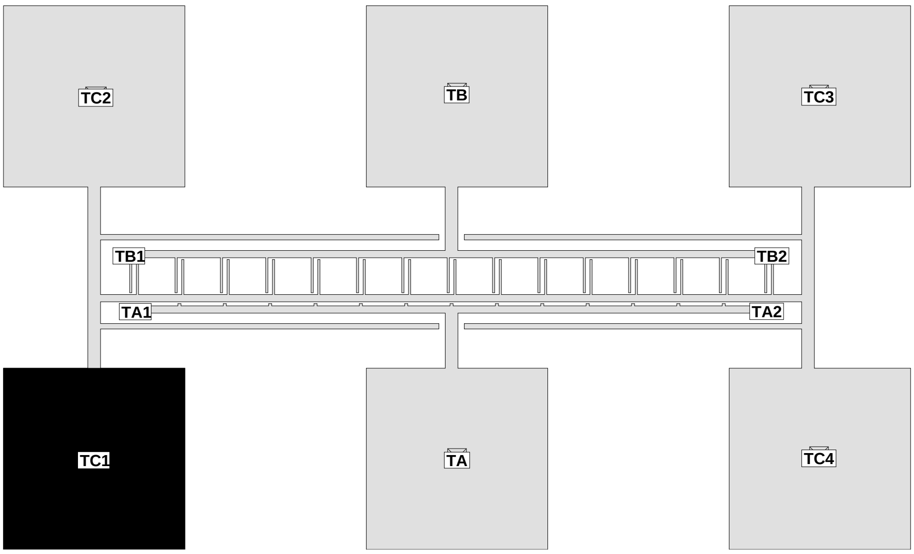
At last, to exit `simeye` program, go to the "File" menu and click on "Exit" and "Yes".

6 ny9t Example for Capacitive Coupling via the Substrate

6.1 Introduction

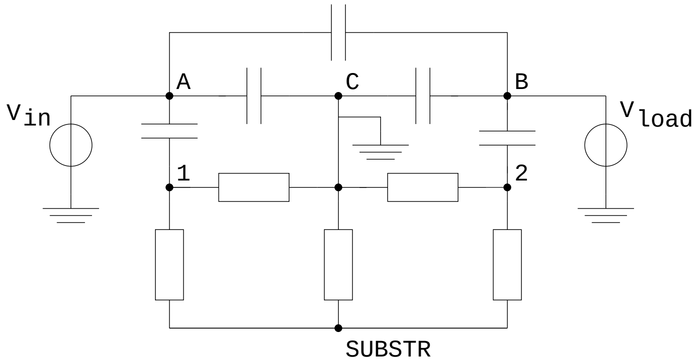
In this tutorial, we will be studying the ny9t example. We will see that the capacitive coupling from metal through the substrate to other metal can be a relevant coupling effect, and we will see how this effect can be modelled using Space.

The example is a calibration/de-embedding structure for MOSFET device characterization. It is accompanied by measurements. We will see that we can match the measurements fairly accurately if the extraction takes the combined effect of capacitive coupling of interconnect into the substrate and resistive coupling through the substrate into account. The layout looks as follows (such a figure can be printed using `getepslay`, see `icdman getepslay`):



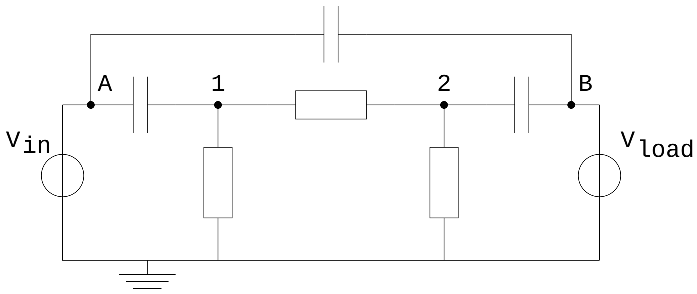
We will study the Y_{12} between port `TA` and `TB`. The ports `TCx`, where x is a number, impose a ground-signal-ground structure. The black pad with port (terminal) `TC1` is connected to the substrate, the others are galvanically connected to `TC1` but are floating above the substrate. Ports `TA1, TA2, TB1, TB2` are connected to `TA` or `TB` respectively.

The basic circuit topology is shown below.



Nodes A, B and C are the terminals (ports) in the layout. In the layout there are multiple terminals with a number attached to the name, they are connected. Nodes A and B have a capacitance to nodes 1 and 2 respectively, these nodes are physically located on top of the substrate underneath the metal traces. If distributed effects are ignored, the coupling areas from metal to substrate become single nodes. Of course, that is only an approximation, the extractor can extract the distributed network. With a single node for 1 and 2, the substrate resistance network is connected to nodes 1, 2 and C, while C is also grounded. There are interconnect capacitances between A and B, between A and C and between B and C. In the simulation we will connect voltage sources to nodes A and B. V_{in} is the input, and we will measure the current through V_{load}. The ratio between load current and input voltage is the Y_{ab} parameter.

Since the $SUBSTR$ node is floating, it can be eliminated from the circuit w/o any accuracy penalty. Since C is grounded, we can also remove the capacitors from A and B to C. Then, the resulting equivalent circuit can be redrawn as follows:



In this circuit, C_{A1} and C_{B2} will have values around $10fF$, C_{AB} will be much smaller, around $30aF$. The resistors to ground will have values around 50Ω and R_{12} will have a value around 200Ω.

6.2 Files

This tutorial is located in the directory `$ICDPATH/share/demo/ny9t`. Initially, it contains the following files:

`meas.dat:`	Contains the measurement data.
`ny9t.gds:`	The layout of the ny9t design.
`stim.spc:`	Contains the sources and control commands for Spice. See Section 6.5.
`commands.gnp:`	Contains the commands for gnuplot to generate the plots.

`tech.s:` The technology file for Space. See Section 6.3.

`param.p:` The parameter settings file for Space. See Section 6.4.

`script.sh:` A file containing the commands for executing all steps of the demo in sequence. See Section 6.5.

6.3 Technology File

The technology file that we will use is the file `tech.s`, see below.

```
listing of file tech.s
        colors:
            cmf      blue
            cca     black

        unit vdimension    1e-6     # --> um
        unit c_resistance  1e-12    # --> ohm um^2

        conductors:
          # name     : boolean : conductor : sheet resistance
            cond_cmf : cmf     : cmf       : 0.060

        contacts:
          # name    : boolean condition : pins     : contact resistance
            con_sub : cca cmf !cwn !csn : cmf @sub : 100

        capacitances:
          # name         : boolean cond.  :  pins     : values
            pcap_cmf_sub : cmf            :  cmf @sub : 3.6348e-05
            ecap_cmf_sub : !cmf -cmf      : -cmf @sub : 5e-07 9.3033e-12
                                                        1e-06 1.6637e-11
            lcap_cmf_sub : !cmf -cmf =cmf : -cmf =cmf : 5e-07 8.2548e-11
                                                        1e-06 4.7931e-11
                                                        2e-06 2.556e-11

        vdimensions:
          # name    : boolean : conductor : zbot thickness
            metal1v : cmf     :    cmf    : 0.95 0.6

        dielectrics:
          # name       permitivity    zbot
            SiO2         3.9           0
            air          1.0           3.15

        sublayers:
          # name       conductivity   ztop
            epi          6.87          0.0
            substrate    10000        -1.6
```

Please note that the masks to be used in this file have already been defined in the `maskdata` file in the process library `$ICDPATH/share/lib/process`. In this example, we will only use the metal-1 (`cmf`) and contact-to-substrate (`cca`) masks. We will now dissect this file, section by section. The first section is the colors section:

```
 1    colors:
 2        cmf     blue
 3        cca    black
```

It specifies that the metal-1 mask `cmf` is to be shown in blue and that the cca mask is to be shown in black. Then comes the units section.

```
 5    unit vdimension    1e-6     # --> um
 6    unit c_resistance  1e-12    # --> ohm um^2
```

This section specifies the units to be used in the rest of the file. It specifies that vertical dimensions are in μm and contact resistances in $\Omega/\mu m^2$. The next section is the conductors section:

```
 8    conductors:
 9      # name     : boolean : conductor : sheet resistance
10        cond_cmf : cmf     : cmf       : 0.060
```

Line 10 specifies that the conductor name `cond_mf` is defined by the boolean condition `cmf` and that mask `cmf` represents the actual conductor. The metal-1 sheet resistance is $60m\Omega/\Box$. The subsequent section defines the contacts:

```
12    contacts:
13      # name    : boolean condition : pins     : contact resistance
14        con_sub : cca cmf !cwn !csn : cmf @sub : 100
```

On line 14 the contact of metal-1 to the substrate is defined. It is a normal contact definition, but the `@sub` indicates that the contact is to substrate. The contact resistance is $100\Omega/\mu m^2$, see line 6. Subsequently, the `capacitances` section starts the interconnect capacitances.

```
16    capacitances:
17      # name         : boolean cond.  :  pins     : values
18        pcap_cmf_sub : cmf            :  cmf @sub : 3.6348e-05
19        ecap_cmf_sub : !cmf -cmf      : -cmf @sub : 5e-07 9.3033e-12
20                                                  1e-06 1.6637e-11
21        lcap_cmf_sub : !cmf -cmf =cmf : -cmf =cmf : 5e-07 8.2548e-11
22                                                  1e-06 4.7931e-11
23                                                  2e-06 2.556e-11
```

It specifies the capacitance values (as a function of the geometry) for 2.5D capacitance extraction, but these values are not used in our example, since we will use the 3D BEM method for capacitance extraction. Nevertheless, the `@sub` in these lines defines that the metal-1 capacitance couples to the substrate, instead of ideal ground (as would be the case when `gnd` would have been used instead of `@sub`). Thus, although the specified values are not used, we can not remove these lines (although we could remove the metal sidewall to metal sidewall capacitance on lines 21 ... 23). Subsequently, the 3D data for the interconnect is started on line 25.

```
25    vdimensions:
26      # name    : boolean : conductor : zbot thickness
27        metal1v : cmf     :   cmf     : 0.95 0.6
```

Line 27 section specifies a very simple boolean condition for the metal-1 height and thickness, being $0.95\mu m$ and $0.6\mu m$, respectively. The `vdimension` units were specified on line 6. Subsequently, the `dielectrics` section specifies the dielectric properties for the interconnect capacitance extraction.

```
29    dielectrics:
30      # name       permitivity    zbot
31        SiO2         3.9           0
32        air          1.0           3.15
```

Line 31 states that a layer with $\epsilon_r = 3.9$ begins on $z = 0$, i.e. on the silicon surface. The next line states that a layer with $\epsilon_r = 1.0$ begins on $z = 3.5\mu m$, implying that the SiO2 actually ends here. Air extends to infinity. The strings `SiO2` and `air` are just arbitrary names only used for documentation and messages. The final section is the sublayers section.

```
sublayers:
  # name       conductivity   ztop
    epi          6.87           0.0
    substrate    10000         -1.6
```

It works similar to the dielectrics section, but here the conductivity in S/m or $1/\Omega m$ of the substrate layers is specified, where the top of each layer is given with the silicon surface again at $z = 0$, and here z being negative.

6.4 Parameter File

The next file controlling the extraction is the parameter file `param.p`. We will only be looking at the following part of this file:

```
 9    cap3d.be_mode          0g
10    cap3d.max_be_area      5
11    cap3d.be_window        10
12
13    sub3d.max_be_area      10
14    sub3d.be_window        10
```

Here, line 9 specifies the mode for 3D boundary element extraction. It is the `0g` mode, meaning $0th$ order Galerkin mode, see the Space User's Manual. The next line specifies the maximum area in μm^2 for the BEM panels (which is chosen relatively large in order to reduce the computation time), and the window in mum over which coupling capacitances should be computed. The value shown is reasonable, given the fact that the metal-1 is very close to the substrate. Hence, the electric field lines die out very rapidly. Lines 13 and 14 specify the maximum panel area and influence window for the substrate resistance extraction. On the type of substrate, the $10\mu m$ is actually far greater then necessary, since over such distances the current will be flowing through the deep substrate anyway.

6.5 Running the Extractor

The file `script.sh` is a batch file for running all the commands for this example, see below. The script can be executed by the following command at the Unix prompt:

```
# sh script.sh
```

Alternatively, commands from this file can be cut and paste individually in a terminal window.

```
listing of file script.sh
#!/bin/sh

```

```
 3    if test -f .dmrc; then rmpr -fs .; fi
 4
 5    mkpr -p scmos_n -l 0.01 .
 6    cgi ny9t.gds
 7    tecc tech.s
 8
 9    space3d -vFC3r -E tech.t -P param.p -S name_extension=_nosub ny9t
10    xspice -ax ny9t_nosub -S stim.spc > ny9t_nosub.spc
11    spice3 < ny9t_nosub.spc > ny9t_nosub.dat
12
13    space3d -BvFC3r -E tech.t -P param.p -Sname_extension=_sub ny9t
14    xspice -ax ny9t_sub -S stim.spc > ny9t_sub.spc
15    spice3 < ny9t_sub.spc > ny9t_sub.dat
16
17    gnuplot < commands.gnp
```

Line 3 tests if a database already exists, if so, the database is removed to start completely fresh. A database will not exist if this script is run for the first time but should exist on each subsequent occasion of running this script. The test is to see if the file `.dmrc` exists, which is one of the files and directories that are created for a database. The command for removing the database is `rmpr`, it takes as argument a pathname to the database. In this case, the path is `.`, meaning the current working directory. See `icdman rmpr`. If there are other non-project files and/or directories, they are preserved. Options:

- `-f`: Force removal of all cells in project before project is removed. Without this option, `rmpr` requires the project to be empty before removal.

To remove all or part of the contents of a project, use command `rmdb` (see `icdman rmdb`).

The next command, on line 5, creates a new database. It takes the following options:

- `-p scmos_n`: Use the `scmos_n` process from the technology library. The process data is normally located in the directory `$ICDPATH/share/lib/process`, and any name or number of the process can be specified at the `-p` option.
- `-l` *`lambda`*: Specify the internal grid of the database in μm. Here we have used a value of $0.01\mu m$. This value should be fine enough so that the co-ordinates of the items in the GDS2 file to be imported (see the next line) fit on the grid. If they don't fit, they are rounded (and warning messages are printed).
- `.`: The last item is the pathname to the database created. It can be a relative or absolute pathname. Here it is `.` for the current working directory.

The next command on line 6 is to import the GDS2 file, see `icdman cgi`. No options are needed in this case. Now, the layout can be seen using the command (see Dali User's Manual):

```
# dali ny9t
```

Subsequently, on line 7 the technology file (see Section 6.3) is compiled. The technology compiler turns a `.s` file into a `.t` file. Space can only read a `.t` file, not a `.s` file directly.

Now, the actual extractions and simulations can be run. Two different extractions will be run, one without and one with substrate resistance extraction. Line 9 and line 13 are the two command lines for `space3d`. See `icdman space` for a full list of all the possible options. The options that are specified, beside `-v` for verbose, have the following meaning:

- `-F:` Flat extraction. Note that this is actually the default behavior.
- `-C:` Capacitance extraction. It is combined with the -3 option.
- `-3:` Use the 3-dimensional BEM for capacitance extraction.
- `-r:` Extract interconnect resistances.
- `-B:` Extract substrate resistances using a boundary-element technique. This option is used for the second invocation of `space` on line 13 but not on line 9.
- `-E tech.t:`
  Use the file `tech.t` as technology file.
- `-P param.p:`
  Use the file `param.p` as parameter setting file.
- `-S name_extension=`*`ext`*`:`
  Use *ext* as an extension for the name of the circuit to be extracted, relative to the name of the layout. For example, on line 9 the layout of `ny9t` is to be extracted as circuit `ny9t_nosub`.

The final argument of the `space3d` command is the name of the layout (here: `ny9t`) to be extracted.

The next command in this file is the `xspice` command, see lines 10 and 14. This command retrieves a spice listing from the database, which is redirected into a file (`ny9t_nosub.spc` and `ny9t_sub.spc` respectively). See `icdman xspice` for a full list of options. Options used:

- `-a:` Produce alpha-numeric node names. W/o this option, only node numbers are produced. The node-names are inherited from the terminal names (port names) in the layout.
- `-x:` Use node number = 0 for nodes whose name start with ”gnd” or ”GND”.
- `-S `*`stimfile`*`:`
  append the file *`stimfile`* to the output file. The stimfile can contain input stimuli and simulation control statements (such as print and plot commands and the .tran statement). This file is just copied, no checking or interpretation of its contents is done.

For the example, the contents of `stim.spc` look as follows:

```
listing of file stim.spc
   1    * input source and output and GND groundings
   2    vin   TA TC1 DC 0 AC 1
   3    vload TB TC2 DC 0
   4    vgnd  TC1  0 DC 0
   5
   6    * first line after .control can only contain title
   7    .control
   8    ny9t example
   9    ac dec 100 100Meg 50G
  10    * plot abs(re(i(vload))),abs(im(i(vload))) loglog
  11    print abs(re(i(vload))),abs(im(i(vload)))
  12    .endc
```

It contains three sources, starting on line 2, and a control block starting on line 7. The `plot` command on line 10 is commented out in favor of the `print` statement on the next line. We will use `gnuplot` to produce a graph from the output produced by the `print` statement. Alternatively, a program such as `matlab` or most any other program that can produce graphs could be used. If the `plot` command is used, the signals could be viewed using `nutmeg` or any other `spice` viewer. The analysis to be done by `spice` is an AC sweep, see line 9, and the output is the real and imaginary parts of the output current (which was short-circuited to ground on line 3).

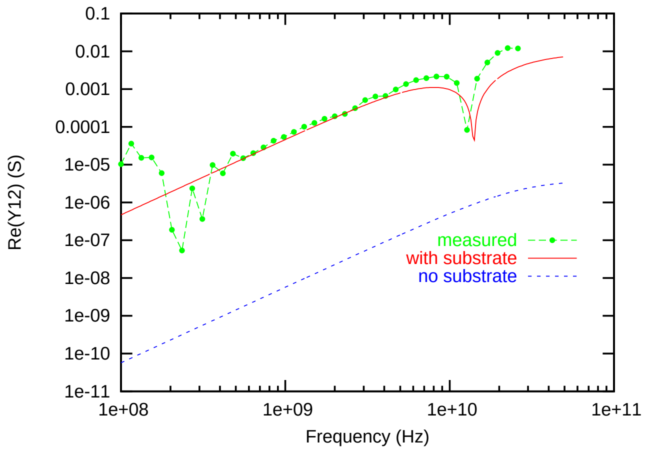
We now continue with the file `script.sh`, see lines 11 and 15:

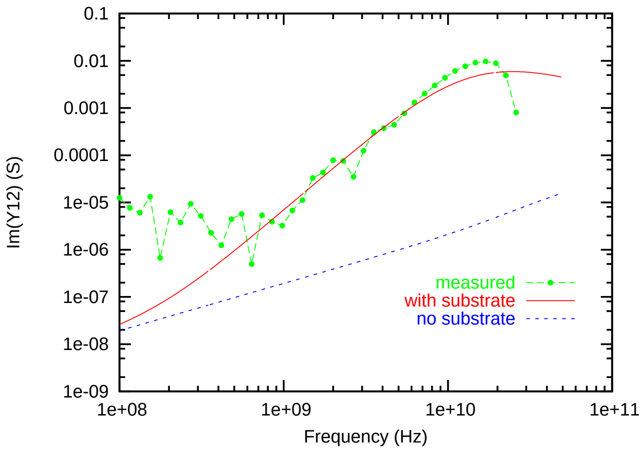
```
11    spice3 < ny9t_nosub.spc > ny9t_nosub.dat
```

```
15    spice3 < ny9t_sub.spc > ny9t_sub.dat
```

These lines run the spice simulations for the two circuits, named `ny9t_nosub` and `ny9t_sub` for the versions with and without the substrate resistances extracted respectively, and store the results in the two `.dat` files.

If the simulations have been run, the results are plotted using `gnuplot` on line 17. Then, two .eps files are produced, which can be inspected using e.g. `ghostscript`. Alternatively, the `commands.gnp` file can be edited to instruct `gnuplot` to produce one of many other possible output formats or to produce the graphs directly on the screen. The graphs should look as follows.





These graphs compare the simulations to the measurements, for the real and imaginary parts of Y_{12} and for the two networks with and w/o the substrate, respectively. A reasonable match is demonstrated. Please note that the negative peak in the real part of Y_{12} actually corresponds to a 90-degree phase shift. In the simulation, it can be made almost arbitrarily sharp by increasing the number of simulation points per decade in the file `stim.spc`, line 9.

6.6 What Next

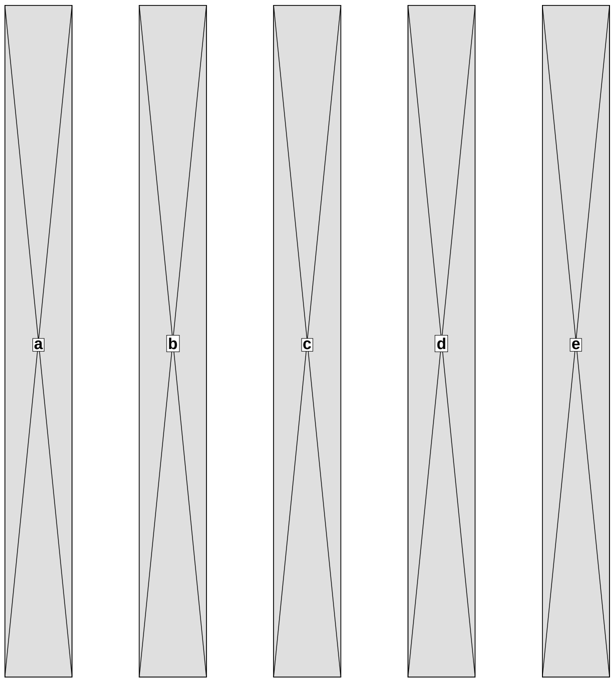
This example is good for experimenting with the different parameters and settings of Space. For example, by reducing the parameter `cap3d.be_window` in `param.p` to about 4, the capacitor between nodes A and B (see the schematics in Section 6.1) will not be found anymore since the distance between A and B in the layout exceeds this value. Then, the curve for the case of 'no substrate' in the graph for $Im(Y_{12})$ looks completely different, although the curve for the case of 'with substrate' is hardly changed. Also, instead of using 3D interconnect capacitance extraction, 2.5D extraction could be used by specifying the `-l` option of Space instead of the `-3` option, giving similar results as with a small window but much faster.

7 poly5 Example of 3D Capacitance Extraction

7.1 Introduction

In this tutorial, we will be studying the poly5 example. We will see how Space is used to compute 3D capacitances. See also the ”Space 3D Capacitance Extraction User’s Manual”.

The example consists out of a configuration of 5 parallel conductors. The conductors have a length of 5 micron, a width of 0.5 micron, a height of 0.5 micron and their separation is also 0.5 micron. Each conductor is assigned a terminal (port). Thus, each conductor can easy be found in the extracted netlist. The layout looks as follows (such a figure can be printed using `getepslay`, see `icdman getepslay`):



7.2 Files

This tutorial is located in the directory `$ICDPATH/share/demo/poly5`. Initially, it contains the following files:

`README:`	A file containing information about the demo.
`poly5.gds:`	The layout of the poly5 design.

`tech.s`: The technology file for Space. See Section 7.3.

`param.p`: The parameter settings file for Space. See Section 7.4.

`script.sh`: A file containing the commands for executing all steps of the demo in sequence.

7.3 Technology File

The technology file that we will use is the file `tech.s`, see listing below.

```
listing of file tech.s
        ...
   3
   4    unit vdimension  1e-6  # micron
   5
   6    colors :
   7        cpg  red
   8
   9    conductors :
  10        resP : cpg : cpg : 0.0
  11
  12    vdimensions :
  13        dimP : cpg : cpg : 0.5 0.5
  14
  15    dielectrics :
  16        # Dielectric consists of 5 micron thick SiO2
  17        # (epsilon = 3.9) on a conducting plane.
  18        SiO2   3.9   0.0
  19        air    1.0   5.0
```

Please note that the masks to be used in this file have already been defined in the `maskdata` file in the process library `$ICDPATH/share/lib/process`. In this example, we will only use the poly-silicon mask (`cpg`).

When we look to the file, we first see on line `4` an `unit` specification for the `vdimensions` definitions starting on line `12`. An unit of `1e-6` m is chosen for the values. Thus, the values must be specified in μm.

Starting on line `6` we find the `colors` specification. The color of the `cpg` mask is chosen to be `red`. Note that this info is only used by the visualization program `Xspace`.

Starting on line `9` we find the `conductors` specification. This is an important section, because without this we can not specify element pins and even a vdimension. The conductor value is for this example unimportant, because we don't extract resistances.

For 3D capacitance extraction, we need only two additional specifications. First, the `vdimensions` definitions for building the 3D conductor mesh. This section specifies respectivily the bottom (against the ground plane) and the thickness of each conductor. Second, the `dielectrics` definitions specify the dielectric properties of the media around the conductors, respectivily the relative permitivity ϵ_r and the bottom in μm. Note that the thickness of the last medium (`air`) extends to infinity.

7.4 Parameter File

The parameter file `param.p` is controlling the extraction, see listing below.

```
listing of file param.p
        ...
   3
   4    BEGIN cap3d
   5    be_mode          0c
   6    max_be_area      0.5
   7    be_window        1
   8    END cap3d
```

In the above listing, we see a `cap3d` block from line 4 to line 8. These parameters are only used for controlling capacitance 3D extraction. The `max_be_area` (in μm^2) and `be_window` (width in μm) parameters needs always to be specified. On line 5, you find also the `be_mode` parameter. However, this boundary element mode is the default (see 3D Cap. User's Manual).

7.5 Running the Extractor

The file `script.sh` is a batch file for running all the commands for this example. First, it changes the current working directory '.' into a project directory:

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

It uses the `scmos_n` process from the technology library. We use the mask names as defined in the `maskdata` file of the library. But, see before, we are not using the default technology file `space.def.s` and parameter file `space.def.p` of the library. Thus, the local technology file `tech.s` needs to be compiled with `tecc` to the format (a `.t` file) which is used by the extractor:

```
% tecc tech.s
```

Second, the layout description is put into the project database. The layout is supplied in a GDS2 file, which can be converted to internal database format with the `cgi` program:

```
% cgi poly5.gds
```

Now, we can extract a circuit description for the layout of the `poly5` cell, as follows:

```
% space3d -C3 -E tech.t -P param.p poly5
```

In this case, the capacitances are calculated, using a 3D mesh method (BEM). The method calculates couple capacitances between the conductors and between the conductors and a ground plane.

The extracted circuit can be inspected using the `xspice` program. We use also the `-f` option to get a file `poly5.spc`:

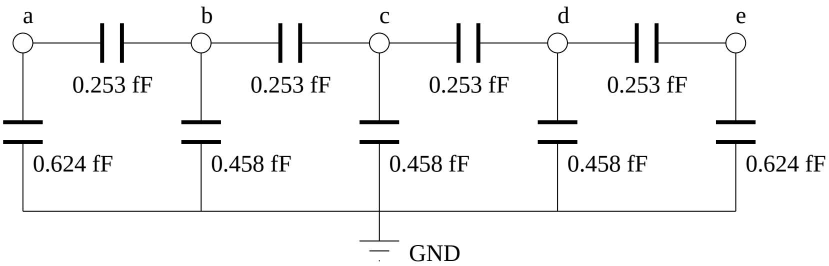
```
% xspice -af poly5
```

The content of the file is listed below.

```
listing of file poly5.spc
poly5

* Generated by: xspice 2.39 25-Jan-2006
* Date: 19-Jun-06 16:03:04 GMT
* Path: /users/simon/poly5
```

```
* Language: SPICE

* circuit poly5 e d c b a
c1 a b 253.3136e-18
c2 a GND 624.1936e-18
c3 b c 253.3136e-18
c4 b GND 457.9544e-18
c5 c d 253.3136e-18
c6 c GND 457.9544e-18
c7 e d 253.3136e-18
c8 e GND 624.1936e-18
c9 d GND 457.9544e-18
* end poly5
```

We can make a drawing of this listing of the extracted circuit. For this topology see the figure below.



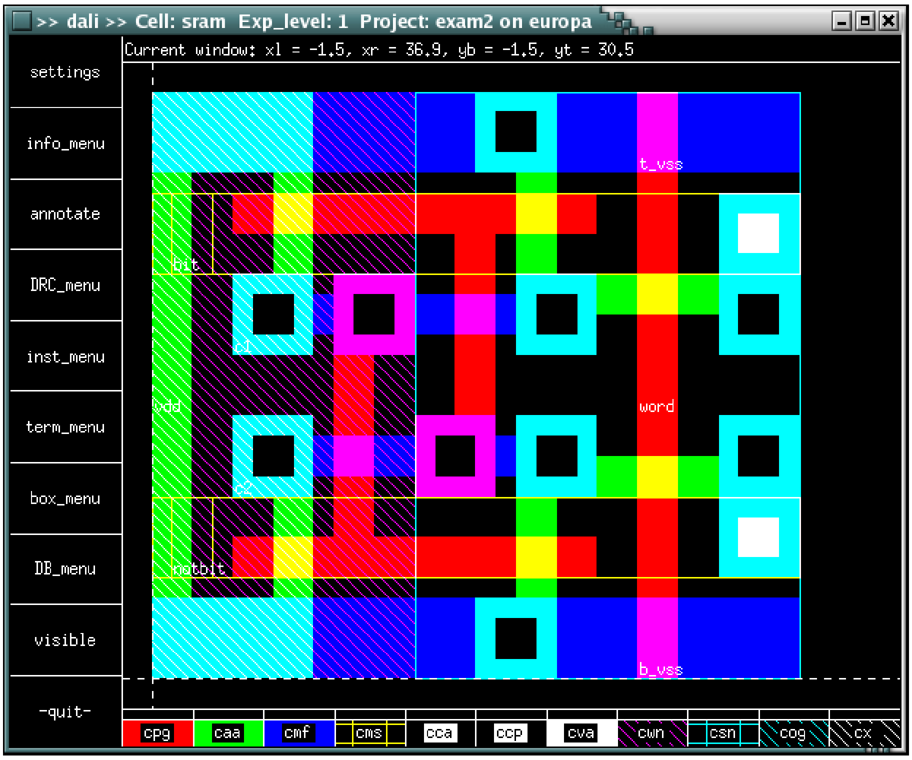
Note that there are no couple capacitances between conductors (a, c), (b, d) and (c, e). This happens, because the chosen `be_window` is $1\mu m$ width, and the distance between these conductors is more than $1\mu m$. Note that the "Space 3D Capacitance User's Manual" shows also the capacitance values in a table for other values of the `be_window`.

8 sram Example of 3D Capacitance Extraction

8.1 Introduction

In this tutorial, we will be studying a CMOS static RAM example. We will see how Space is used to compute 3D capacitances. See also the "Space 3D Capacitance Extraction User's Manual".

This is a more serious example than the 5 parallel conductors example `poly5`. The layout looks as follows, using the layout editor `dali` (see `icdman dali`):



8.2 Files

This tutorial is located in the directory `$ICDPATH/share/demo/sram`. Initially, it contains the following files:

`README:` A file containing information about the demo.

`sram.gds:` The layout of the sram design.

`sram.s:`	The technology file for Space. See Section 8.3.
`sram.p:`	The parameter settings file for Space. See Section 8.4.
`sram.cmd:`	A command file with stimuli to simulate the circuit.
`script.sh:`	A file containing the commands for executing all steps of the demo in sequence.

8.3 Technology File

The technology file that we will use is the file `sram.s`, of which a number of parts are listed below.

```
listing of file sram.s
  ...
unit vdimension    1e-6  # um
unit shape         1e-6  # um
  ...
colors :
   cpg   red
   caa   green
   cmf   blue
   cms   gold
   cca   black
   ccp   black
   cva   black
   cwn   glass
   csn   glass
   cog   glass

conductors :
# name    : condition     : mask : resistivity : type
  cond_mf : cmf           : cmf  : 0.045       : m   # first metal
  cond_ms : cms           : cms  : 0.030       : m   # second metal
  cond_pg : cpg           : cpg  : 40          : m   # poly interc.
  cond_pa : caa !cpg !csn : caa  : 70          : p   # p+ active area
  cond_na : caa !cpg  csn : caa  : 50          : n   # n+ active area

fets :
# name : condition    : gate d/s
  nenh : cpg caa  csn : cpg  caa     # nenh MOS
  penh : cpg caa !csn : cpg  caa     # penh MOS

contacts :
# name   : condition        : lay1 lay2 : resistivity
  cont_s : cva cms cmf      : cms  cmf  :   1     # metal to metal2
  cont_p : ccp cmf cpg      : cmf  cpg  : 100     # metal to poly
  cont_a : cca cmf caa !cpg : cmf  caa  : 100     # metal to act. area

capacitances :
  ...
  ...
```

```
 97  vdimensions :
 98     ver_caa_on_all : caa !cpg   : caa : 0.30 0.00
 99     ver_cpg_of_caa : cpg !caa   : cpg : 0.60 0.50
100     ver_cpg_on_caa : cpg caa    : cpg : 0.35 0.70
101     ver_cmf        : cmf        : cmf : 1.70 0.70
102     ver_cms        : cms        : cms : 2.80 0.70
103
104  dielectrics :
105     # Dielectric consists of 5 micron thick SiO2
106     # (epsilon = 3.9) on a conducting plane.
107     SiO2   3.9   0.0
108     air    1.0   5.0
109
110  #EOF
```

Please note that the masks to be used in this file have already been defined in the `maskdata` file in the process library `$ICDPATH/share/lib/process`. In this example, we will only use the `vdimensions` of 4 masks.

When we look to the file, we first see on line `22` an `unit` specification for the `vdimensions` definitions starting on line `97`. An unit of `1e-6` m is chosen for the values. Thus, the values must be specified in μm. On line `23` is an unit specified for possible `eshapes` definitions. For shape definitions, see sections 3.4 and 3.5 of the ”Space 3D Capacitance User's Manual”.

Starting on line `27` we find the `colors` specification. Note that this information is only used by the visualization program `Xspace`. For a picture, see section 5.2 of the User's Manual. When the color is `glass`, then the mask is invisible.

Starting on line `39` we find the `conductors` definitions. This is an important section, because without this we can not specify element pins and even a vdimension. The conductor values are not important, because we don't extract resistances in this example. On line `44` and `45` are two different conductors specified for the `caa` mask. These are the drain/source regions, also called diffused conductors. These conductors have a conductor type, respectivily `p` and `n`.

We skip all 2D `capacitances` definitions, starting on line `58`. Because, for 3D capacitance extraction, we need only the following two sections. First, the `vdimensions` definitions for building the 3D conductor mesh. This section specifies respectivily the bottom (against the ground plane) and the thickness of each conductor. You find, on line `98`, only one specification for the diffused conductors. And these conductors are modeled with a zero thickness (see section 3.8 of the User's Manual). On lines `99` and `100`, you see, that the gate conductor `cpg` is modeled with two different heights and thicknesses.

Second, the `dielectrics` definitions specify the dielectric properties of the media around the conductors, respectivily the relative permitivity ϵ_r and the bottom in μm. Note that the thickness of the last medium (`air`) extends to infinity.

8.4 Parameter File

The parameter file `sram.p` is controlling the extraction, see listing below.

```
listing of file sram.p
        ...
```

```
4    BEGIN cap3d
5    be_mode                0c
6    be_window              2.0
7    max_be_area            1.0
8    omit_gate_ds_cap       on
9    END cap3d
```

In the above listing, we see a `cap3d` block from line `4` to line `9`. These parameters are only used for controlling capacitance 3D extraction. The `max_be_area` (in μm^2) and `be_window` (width in μm) parameters needs always to be specified. On line `5`, you find also the `be_mode` parameter. However, this boundary element mode is the default (see the 3D Cap. User's Manual). On line `8` you see the `omit_gate_ds_cap` parameter. With this parameter, you can skip the couple capacitances between gate and drain/source regions. Possibly, these capacitances are already present in the used SPICE device model (see section 3.9 of the User's Manual).

8.5 Running the Extractor

For example, you have copied this demo files to a directory `exam2`:

```
% cd exam2
% cp $ICDPATH/share/demo/sram/* .
```

The file `script.sh` is a batch file for running all the commands for this example. First, it changes the current working directory '.' into a project directory:

```
% mkpr -p scmos_n -l 0.25 .
```

The static RAM cell is in 0.5μ technology. Thus, we use a lambda value (option `-l`) of $0.25\mu m$. For the CMOS technology, we use the `scmos_n` process from the technology library. We use the mask names as defined in the `maskdata` file of the library. But, we are not using the default technology file `space.def.s` and parameter file `space.def.p` of the library. Thus, the local technology file `sram.s` needs to be compiled with `tecc` to the format (a `.t` file) which is used by the extractor:

```
% tecc sram.s
```

Second, the layout description is put into the project database. The layout is supplied in a GDS2 file, which can be converted to internal database format with the `cgi` program:

```
% cgi sram.gds
```

Now, we can extract a circuit description for the layout of the `sram` cell, as follows:

```
% space3d -C3 -E sram.t -P sram.p sram
```

In this case, the capacitances are calculated, using a 3D mesh method (BEM). The method calculates couple capacitances between the conductors and between the conductors and a ground plane.

The extracted circuit can be inspected using the `xspice` program. We use option `-a` to get alphanumeric node names.

```
% xspice -a sram
```

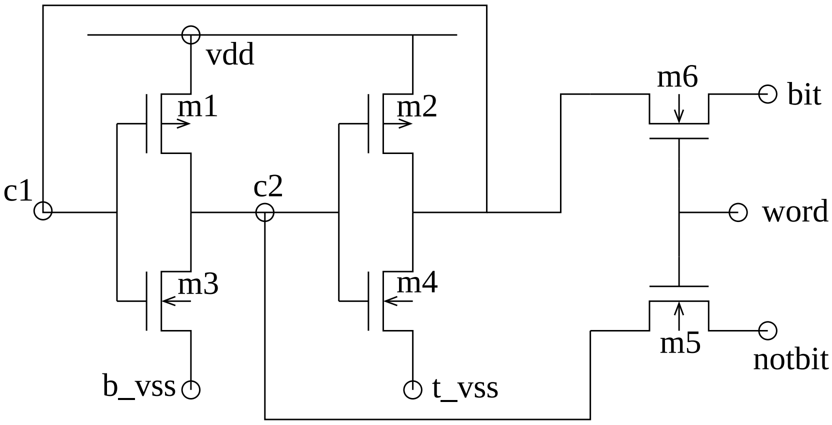
The output of the SPICE circuit is listed below.

```
sram

* Generated by: xspice 2.39 25-Jan-2006
```

```
* Date: 20-Jun-06 10:40:34 GMT
* Path: /users/simon/exam2
* Language: SPICE

* circuit sram pbulk nbulk word vdd b_vss t_vss c1 c2 bit notbit
m1 vdd c1 c2 pbulk penh_0 w=500n l=500n
m2 vdd c2 c1 pbulk penh_0 w=500n l=500n
m3 b_vss c1 c2 nbulk nenh_0 w=500n l=500n
m4 t_vss c2 c1 nbulk nenh_0 w=500n l=500n
m5 notbit word c2 nbulk nenh_0 w=500n l=500n
m6 bit word c1 nbulk nenh_0 w=500n l=500n
c1 b_vss word 114.8418e-18
c2 b_vss vdd 43.84559e-18
c3 b_vss c2 74.079e-18
c4 b_vss c1 212.2091e-18
c5 b_vss notbit 610.2115e-18
c6 b_vss GND 1.80899f
c7 notbit bit 107.3522e-18
c8 notbit word 115.7189e-18
c9 notbit vdd 6.609528e-18
c10 notbit c2 360.549e-18
c11 notbit c1 66.39728e-18
c12 notbit GND 2.597814f
c13 t_vss word 114.8457e-18
c14 t_vss vdd 43.82952e-18
c15 t_vss bit 610.3344e-18
c16 t_vss c2 212.2316e-18
c17 t_vss c1 73.78773e-18
c18 t_vss GND 1.808788f
c19 word bit 115.8338e-18
c20 word c2 113.7943e-18
c21 word c1 111.6566e-18
c22 word GND 344.449e-18
c23 vdd bit 6.60235e-18
c24 vdd c2 56.77325e-18
c25 vdd c1 58.65703e-18
c26 vdd GND 9.5875f
c27 bit c1 360.2213e-18
c28 bit c2 67.05254e-18
c29 bit GND 2.597536f
c30 c2 c1 998.1198e-18
c31 c2 GND 5.426256f
c32 c1 GND 5.425883f
* end sram

.model penh_0 pmos(level=2 ld=0 tox=25n nsub=50e15 vto=-1.1 uo=200 uexp=100m
+ ucrit=10k delta=200m xj=500n vmax=50k neff=1 rsh=0 nfs=0 js=10u cj=500u
+ cjsw=600p mj=500m mjsw=300m pb=800m cgdo=300p cgso=300p)
.model nenh_0 nmos(level=2 ld=0 tox=25n nsub=20e15 vto=700m uo=600 uexp=100m
+ ucrit=10k delta=200m xj=500n vmax=50k neff=1 rsh=0 nfs=0 js=2u cj=100u
+ cjsw=300p mj=500m mjsw=300m pb=800m cgdo=300p cgso=300p)
```

```
57   vpbulk pbulk 0 5
58   rpbulk pbulk 0 100meg
59
60   vnbulk nbulk 0 0
61   rnbulk nbulk 0 100meg
```

On lines `49` to `54`, you find the device models for `penh_0` and `nenh_0` fets. Lines `57` to `61` are automatically added, and are the `pbulk` and `nbulk` voltage sources and resistors. Below, you find a drawing of the listing of the extracted circuit.



Note that there are couple capacitances between almost all nodes of the circuit. This happens, because the chosen `be_window` is $2\mu m$ width. Each node has also a couple capacitance to the ground plane (node `GND`).

8.6 Running a Circuit Simulation

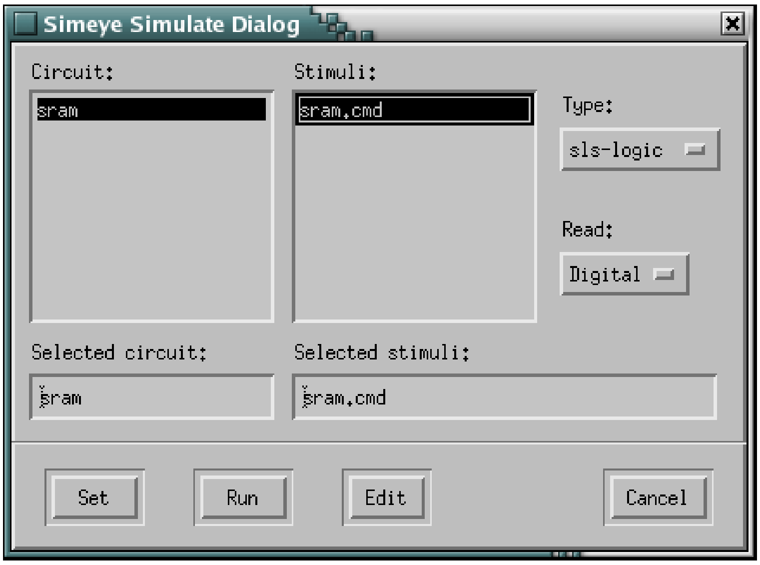
With the simulation GUI `simeye` you can run a SLS or SPICE circuit simulation. With SLS, you can choice between a logic and timing simulation. The timing simulation is also using the extracted couple capacitances. An accurate analog simulation is performed with the SPICE simulator. The following command file with stimuli is used for both simulators:

```
listing of file sram.cmd
     ...
     /*
     /* Command file for simulating the sram circuit with simeye/nspice.
     */

     print vdd t_vss b_vss word bit notbit c1 c2
     plot  vdd t_vss b_vss word bit notbit c1 c2
     option simperiod = 30
     option sigunit = 0.75e-10
     option outacc  = 10p
     option level = 3

     /*  spice commands:

```

```
16   *%
17   trise 0.075n
18   tfall 0.075n
19   tstep 0.0125n
20   *%
21   .options nomod
22   .options limpts=601
23   .options cptime=30
24   *%
25
26   */
27   set vdd = h*25
28   set t_vss = l*25
29   set b_vss = l*25
30   set word = l*5 h*5 l*5 h*5 l*5 h
31   set notbit = l*10 h*10 l*5
32   set bit = h*10 l*10 h*5
```

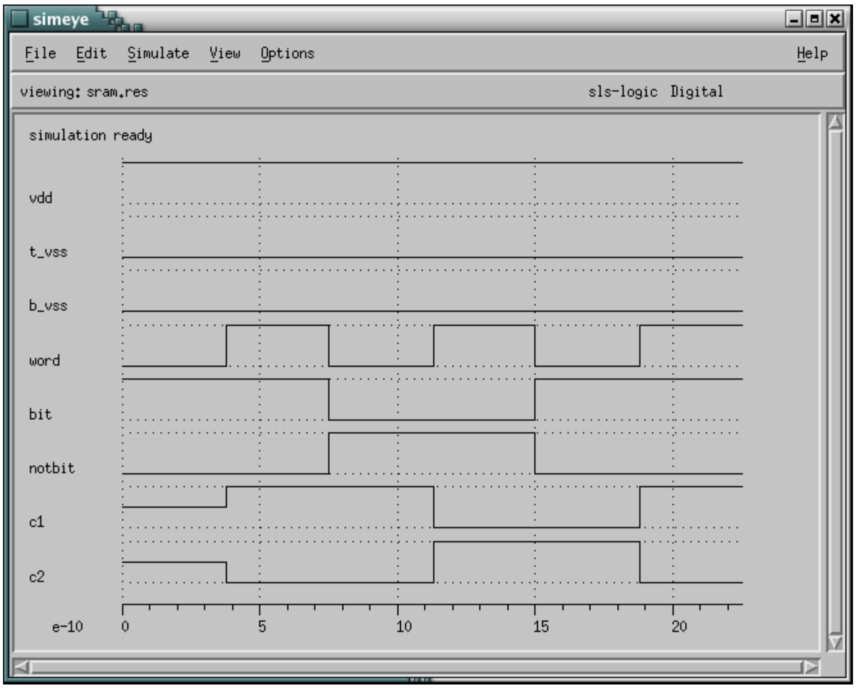
For a SPICE simulation, `simeye` is using the script `nspice`. First, the script contains commands to prepare the circuit and to translate the stimuli to pwl voltage sources. Note that these stimuli are connected between the nodes and the ground node `0`. On line `27` you see node `vdd`, the positive power supply node, which gets a high voltage (default $5V$). On lines `28` and `29` you see the `vss` nodes, which get a low voltage of $0V$ and are connected to the ground node. Second, the script tries to run the Berkeley `spice3` simulator. For more information, see `icdman nspice`.

Below, you see the `simeye` Simulate Prepare Dialog.

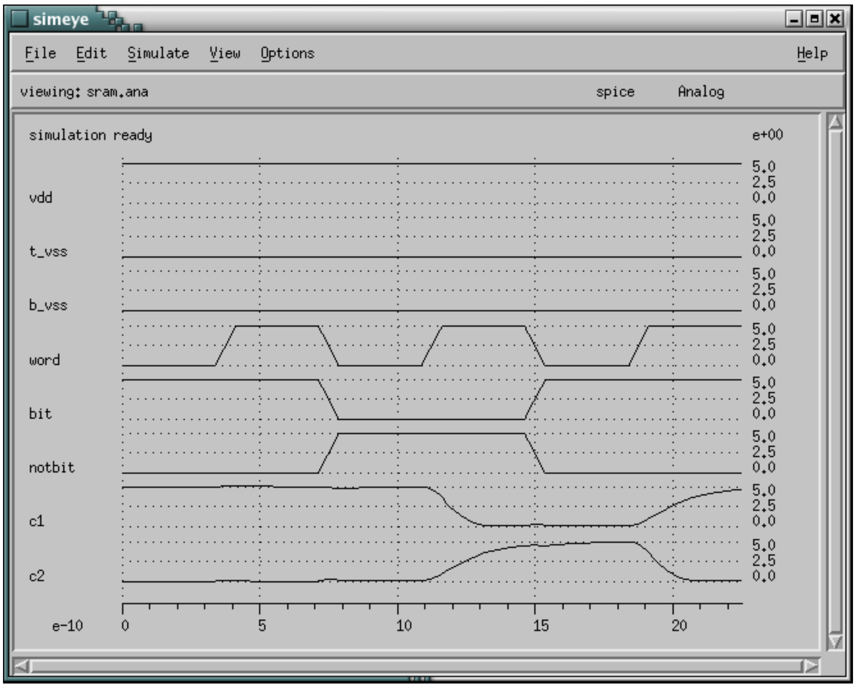


On the next page, you find the results of a logic SLS and analog SPICE simulation.

Below, you see the result of a logic SLS simulation.



Below, you see the result of a analog SPICE simulation.

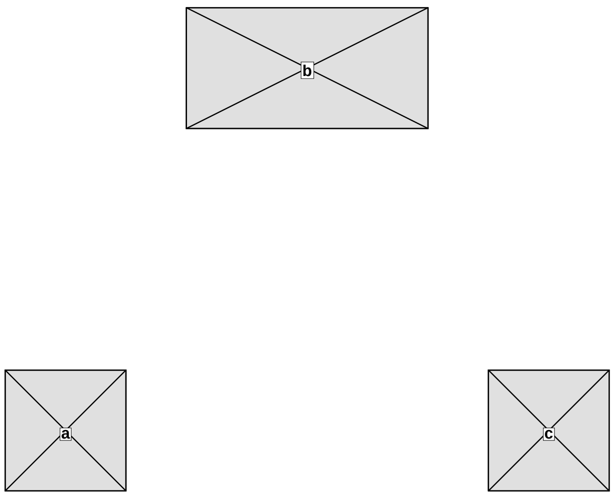


9 sub3term Example of Substrate Resistance Extraction

9.1 Introduction

In this example, we will be studying a configuration of 3 substrate terminals. We will see how Space is used to compute the 3D substrate resistances. See also the ”Space Substrate Resistance Extraction User’s Manual”.

The layout looks as follows (such a figure can be printed using `getepslay`, see `icdman getepslay`):



9.2 Files

This tutorial is located in the directory `$ICDPATH/share/demo/sub3term`. Initially, it contains the following files:

`README:` A file containing information about the demo.

`sub3term.gds:` The layout of the sub3term design.

`elem.s:` The technology file for Space. See Section 9.3.

`param.p:` The parameter settings file for Space. See Section 9.4.

`script.sh:` A file containing the commands for executing all steps of the demo in sequence.

9.3 Technology File

The technology file that we will use is the file `elem.s`, see listing below.

```
listing of file elem.s
   3    #
   4    # space element definition file for metal substrate terminals
   5    #
   6
   7    colors :
   8        cmf   blue
   9        @sub  pink
  10
  11    conductors :
  12      # name     : condition : mask : resistivity : type
  13        cond_cmf : cmf       : cmf  : 0.0         : m
  14
  15    contacts :
  16      # name     : condition : lay1 lay2 : resistivity
  17        cont_cmf : cmf       : cmf  @sub : 0.0
  18
  19    sublayers :
  20      # name       conductivity  top
  21        epi        6.7            0.0
  22        substrate  2.0e3         -7.0
  23
  24    #EOF
```

Please note that the masks to be used in this file have already been defined in the `maskdata` file in the process library `$ICDPATH/share/lib/process`. In this example, we will only use the first metal mask (`cmf`).

When we look to the file, we first see on line `7` the `colors` specification. The color of the `cmf` mask is chosen to be `blue`. The color for the `@sub` mesh is chosen to be `pink`. Note that this info is only used by the visualization tool `Xspace`.

Starting on line `11` we find the `conductors` definitions. This is an important section, because without this we can not specify element pins and contacts. The conductor value is for this example unimportant, because we don't extract interconnect resistances.

Starting on line `15` we find the `contacts` definitions. This is also an important section, because here is the connection between first metal and the substrate specified. These nodes are also called substrate terminals. However, substrate terminals can also be bulk regions or pins of capacitances.

For 3D substrate resistance extraction, we need only one additional section. Starting on line `19` we find the `sublayers` definitions. These definitions specify for each substrate layer the conductivity (in S/m) and the starting position (in μm). Note that the last layer (on line `22`) extends to minus infinity.

9.4 Parameter File

The parameter file `param.p` is controlling the extraction, see listing below.

```
listing of file param.p
  3
  4  BEGIN sub3d
  5  be_mode         0c   # piecewise constant collocation
  6  max_be_area     1.0  # max. size of interior elements in sq. microns
  7  edge_be_ratio   0.05 # max. size edge elem. / max size inter. elem.
  8  edge_be_split   0.2  # split fraction for edge elements
  9  be_window     inf    # infinite window, all resistances
 10  END sub3d
 11
 12  disp.save_prepass_image   on
```

In the above listing, we see a sub3d block from line 4 to line 10. These parameters are only used for controlling the substrate 3D extraction. The max_be_area (in μm^2) and be_window (width in μm) parameter needs always to be specified. On line 5, you find also the be_mode parameter. However, this boundary element mode is the default. On line 9, you see that the be_window parameter is chosen to be infinity. That means, that all substrate terminal nodes are in the same strip, and are all connected with each other. See for an explanation of the parameters also the ”Space Substrate Extraction User’s Manual”.

On line 12, you find a parameter for the visualization tool Xspace. It means, that the image of the prepass may not be cleared when starting drawing the image of the last pass. See for an explanation also the ”Xspace User’s Manual”.

9.5 Running the Extractor

The file script.sh is a batch file for running all the commands for this example. First, it changes the current working directory ’.’ into a project directory:

```
% mkpr -p scmos_n -l 0.1 .
```

We use a lambda value (option -l) of $0.1\mu m$. For the CMOS technology, we use the scmos_n process from the technology library. We use the mask names as defined in the maskdata file of the library. But, we are not using the default technology file space.def.s and parameter file space.def.p of the library. Thus, the local technology file elem.s needs to be compiled with tecc to the format (a .t file) which is used by the extractor:

```
% tecc elem.s
```

Second, the layout description is put into the project database. The layout is supplied in a GDS2 file, which can be converted to internal database format with the cgi program:

```
% cgi sub3term.gds
```

Now, we can extract a circuit description for the layout of the sub3term cell, as follows:

```
% space3d -vB -E elem.t -P param.p sub3term
```

We use the verbose option (-v) to see what the extractor program is doing. Second, we use option -B to perform a 3D substrate resistance extraction. The substrate resistances are calculated, using a 3D mesh method (BEM). The calculated resistance values are assigned between substrate terminals and between the substrate terminals and the substrate plane (node SUBSTR).

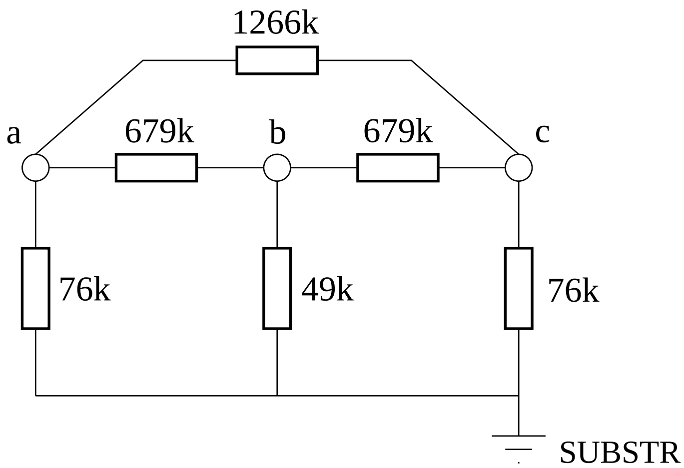
The extracted circuit can be inspected using the xspice program. We use option -a to get alphanumeric node names.

```
% xspice -a sub3term
```

The output of the SPICE circuit is listed below.

```
sub3term

* Generated by: xspice 2.39 25-Jan-2006
* Date: 21-Jun-06 14:56:15 GMT
* Path: /users/simon/sub3term
* Language: SPICE

* circuit sub3term c b a
r1 c a 1.265872meg
r2 c b 679.1891k
r3 c SUBSTR 75.86724k
r4 a b 679.1891k
r5 a SUBSTR 75.86724k
r6 b SUBSTR 49.44378k
* end sub3term
```

Below, you find a drawing of the listing of the extracted circuit.



Note that there are substrate resistances between all nodes of the circuit. This happens, because the chosen `be_window` has a width of infinity. Each node has also a substrate resistance to the substrate plane (node `SUBSTR`).

9.6 Running the Visualization Tool

When you want to see the boundary element mesh, which is used for the substrate terminal regions, you must use the visualization tool `Xspace`. It runs the `space3d` extractor in the background and requests for graphical output. The extractor writes this graphical output to a "display.out" file, which `Xspace` is reading. The advantage of the output file is, that you can playback it again without running the extractor. And it makes zooming and panning on the image very easy. For additional information, consult the "Xspace User's Manual".

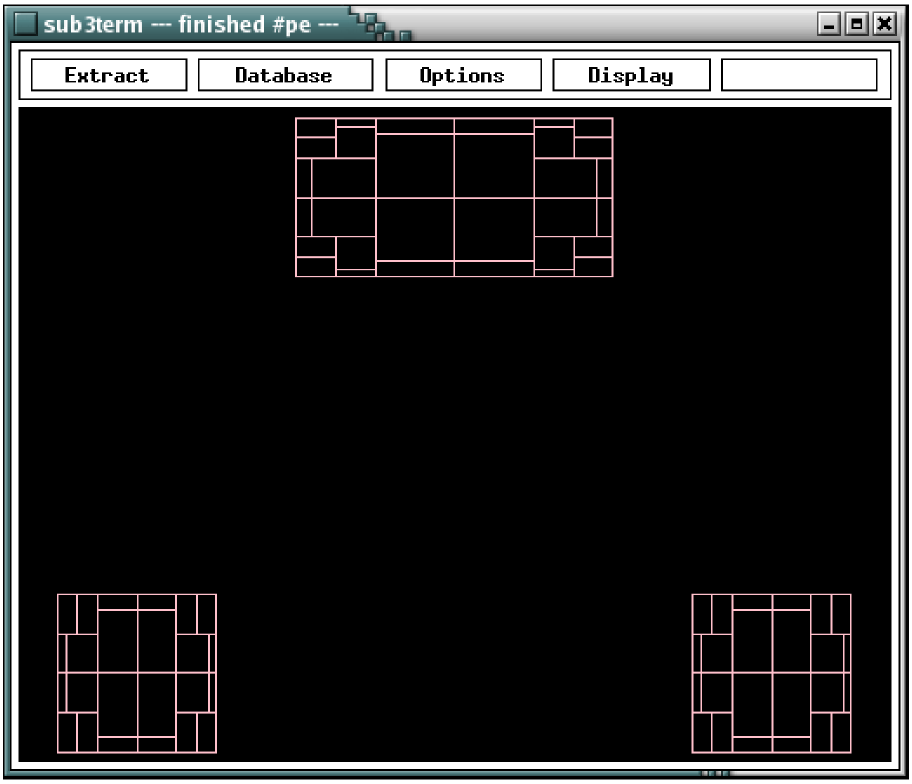
Note that you can playback only the file "display.out" and that `Xspace` needs the color information of the used technology file. To show something, you need to set items in the "Display" menu.

Note that you can only show items, which are written to the file ”display.out”. To start a playback, use hotkey ’a’ (or click on ”extract again” in the ”Extract” menu).

Now, we shall extract a circuit description for the layout of the `sub3term` cell, using the visualization tool as follows:

```
% Xspace -vB -E elem.t -P param.p sub3term
```

You don’t need to set the cell name in the ”Database” menu and also don’t need to set the extract options in the ”Options” menu, because that is already specified on the command line.
You need only to set some display options. Go to the ”Display” menu and click on button ”DrawBEMesh”. To run the extractor, go to the ”Extract” menu and click on ”extract” (or use hotkey ’e’).
You get the following picture:

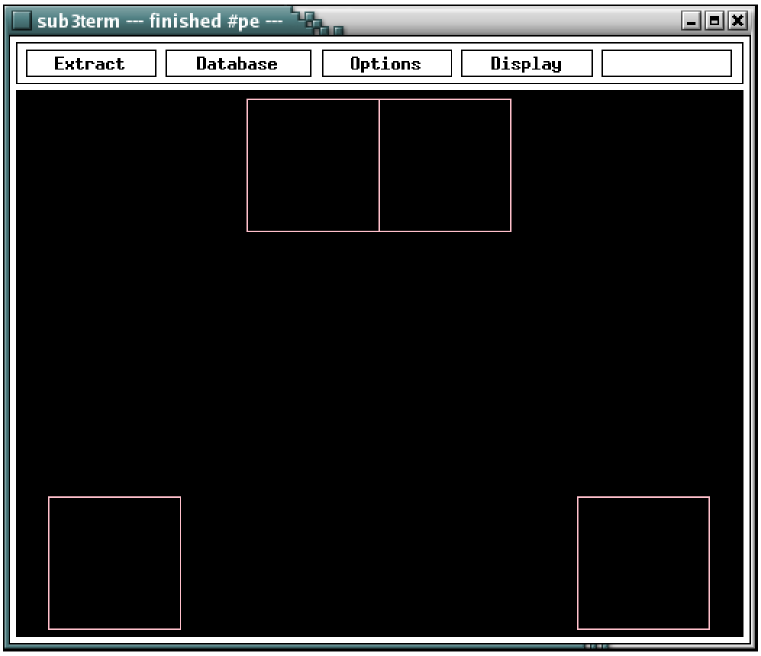


You see three substrate terminals, which are meshed into a number of area’s. The complete area of terminal ”a” and ”c” is $1.0 \mu m^2$. The area of terminal ”b” is $2.0 \mu m^2$.

When you look to the space parameter file ”param.p”, you see that the maximum size of an interior element may be $1.0 \mu m^2$ (parameter `max_be_area`). The maximum size of an edge element, however, may be $0.05 * 1.0 \mu m^2$ (parameter `edge_be_ratio`). When we edit the file ”param.p” and change the `edge_be_ratio` to its default value of 1.0 (or make the line to be a comment line by placing a ’#’ before it), then we shall see what happens.

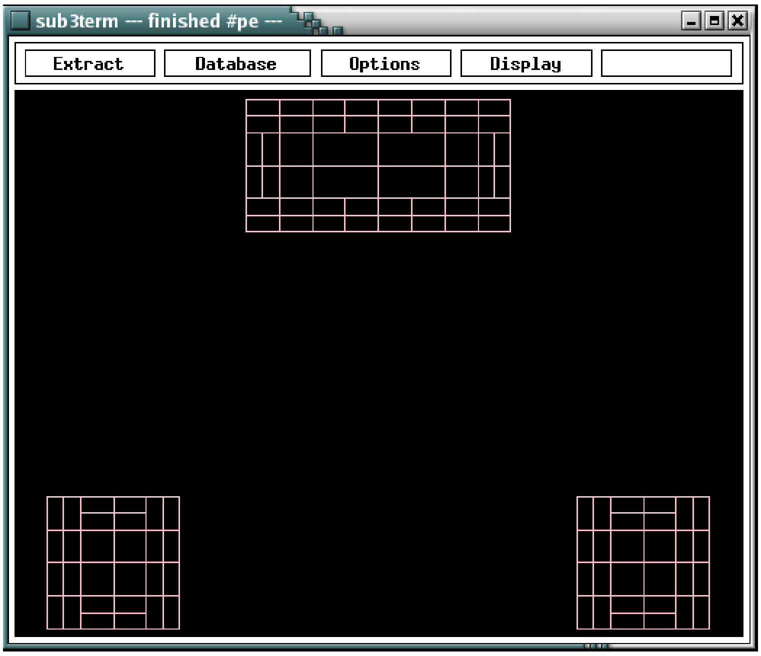
We can now run the visualization tool again. Or better, when you have put it in the background, hit the extract key ’e’ again.

You get the following picture:



And indeed, we get another mesh for the 3 substrate terminals.

Below you see what happens, if we don't change the `edge_be_ratio`, but the `edge_be_split` parameter instead (set this fraction to its default value of 0.5). You get the following picture:



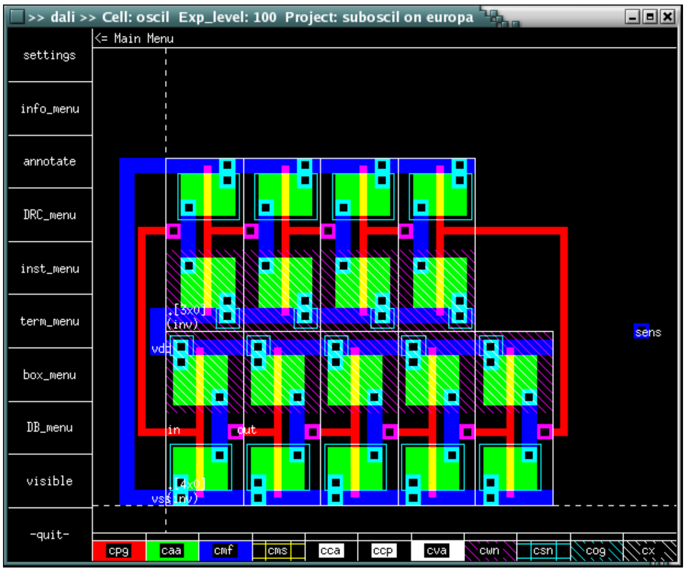
The accuracy of the extraction result is depended of the mesh you chose. And, how finer the mesh, how more computation time is needed to calculate the result.

10 suboscil Example of Substrate Resistance Extraction

10.1 Introduction

In this example, we will be studying a CMOS ring oscillator. We will see how Space is used to compute substrate resistances. First using the fast interpolation method and second using the more accurate 3D BE-method. See also section 5.5 of the ”Space Substrate Resistance Extraction User’s Manual”.

The layout looks as follows, using the layout editor `dali` (see `icdman dali`):



10.2 Files

This tutorial is located in the directory `$ICDPATH/share/demo/suboscil`. Initially, it contains the following files:

`README:`	A file containing information about the demo.
`oscil.gds:`	The layout of the oscillator design.
`oscil.cmd:`	Command file for circuit simulation.
`tech.s:`	The technology file for Space. See Section 10.3.
`tech2.s:`	Alternate technology file for more substrate terminals.
`tech3.s:`	Alternate technology file for using BEM/FEM.

`param.p:` The parameter settings file for Space. See Section 10.4.

`param2.p:` The parameter settings file for Space.

`jun.lib:` Local model library for xspice.

`xspicerc:` Init file for xspice.

`script.sh:` A file containing the commands for executing all steps of the demo in sequence.

10.3 Technology File

The technology file that we will use is the file `tech.s`, see listing below.

```
listing of file tech.s
    ...

unit resistance    1     # ohm
unit c_resistance  1e-12 # ohm um^2
unit a_capacitance 1e-6  # aF/um^2
unit e_capacitance 1e-12 # aF/um
unit capacitance   1e-15 # fF
    ...

conductors :
  # name     : condition      : mask : resistivity : type
    cond_mf : cmf            : cmf  : 0.045       : m     # first metal
    cond_ms : cms            : cms  : 0.030       : m     # second metal
    cond_pg : cpg            : cpg  : 40          : m     # poly interconnect
    cond_pa : caa !cpg !csn : caa  : 70          : p     # p+ active area
    cond_na : caa !cpg  csn : caa  : 50          : n     # n+ active area
    cond_well : cwn          : cwn  : 0           : n     # n well

fets :
  # name : condition    : gate d/s : bulk
    nenh : cpg caa  csn : cpg  caa : @sub  # nenh MOS
    penh : cpg caa !csn : cpg  caa : cwn   # penh MOS

contacts :
  # name    : condition          : lay1 lay2 : resistivity
    cont_s : cva cmf cms        : cmf  cms  :   1   # metal to metal2
    cont_p : ccp cmf cpg        : cmf  cpg  : 100   # metal to poly
    cont_a : cca cmf caa !cpg cwn !csn
           | cca cmf caa !cpg !cwn csn
                                : cmf  caa  : 100   # metal to active area
    cont_w : cca cmf cwn csn    : cmf  cwn  :  80   # metal to well
    cont_b : cca cmf !cwn !csn : cmf  @sub :  80   # metal to subs

junction capacitances ndif :
  # name    :  condition                 : mask1 mask2 : capacitivity
    acap_na :  caa        !cpg  csn !cwn  :  caa @gnd : 100  # n+ bottom
    ecap_na : !caa -caa !-cpg -csn !-cwn : -caa @gnd : 300  # n+ sidewall

```

```
junction capacitances nwell :
    acap_cw :  cwn                  :  cwn @gnd : 100  # bottom
    ecap_cw : !cwn -cwn             : -cwn @gnd : 800  # sidewall

junction capacitances pdif :
    acap_pa :  caa       !cpg  !csn cwn      :  caa cwn : 500 # p+ bottom
    ecap_pa : !caa -caa !-cpg !-csn cwn -cwn : -caa cwn : 600 # p+ sidewall

capacitances :
  # polysilicon capacitances
    acap_cpg_sub :  cpg                !caa !cwn :  cpg @gnd : 49
    acap_cpg_cwn :  cpg                !caa  cwn :  cpg cwn  : 49
    ecap_cpg_sub : !cpg -cpg !cmf !cms !caa !cwn : -cpg @gnd : 52
    ecap_cpg_cwn : !cpg -cpg !cmf !cms !caa  cwn : -cpg cwn  : 52

  # first metal capacitances
    acap_cmf_sub :  cmf           !cpg !caa !cwn :  cmf @gnd : 25
    acap_cmf_cwn :  cmf           !cpg !caa  cwn :  cmf cwn  : 25
    ecap_cmf_sub : !cmf -cmf !cms !cpg !caa !cwn : -cmf @gnd : 52
    ecap_cmf_cwn : !cmf -cmf !cms !cpg !caa  cwn : -cmf cwn  : 52

    acap_cmf_caa :  cmf      caa !cpg !cca :  cmf  caa : 49
    ecap_cmf_caa : !cmf -cmf caa !cms !cpg : -cmf  caa : 59

    acap_cmf_cpg :  cmf      cpg !ccp :  cmf  cpg : 49
    ecap_cmf_cpg : !cmf -cmf cpg !cms : -cmf  cpg : 59
...

sublayers :
  # name       conductivity  top
    substrate  6.7           0.0

selfsubres :
#  Generated by subresgen on 10:56:15 13-5-2003
#    area    perim          r   rest
     0.64      3.2    81286.8   0.01  # w=0.8 l=0.8
     0.64        4   73678.39   0.01  # w=0.4 l=1.6
     0.48      3.2   88205.12   0.01  # w=0.4 l=1.2
     2.56      6.4    40643.4   0.01  # w=1.6 l=1.6
     2.56        8    36839.2   0.01  # w=0.8 l=3.2
     1.92      6.4   44102.56   0.01  # w=0.8 l=2.4
    10.24     12.8    20321.7   0.01  # w=3.2 l=3.2
    10.24       16    18419.9   0.01  # w=1.6 l=6.4
     7.68     12.8   22051.22   0.01  # w=1.6 l=4.8
    40.96     25.6   10160.85   0.01  # w=6.4 l=6.4
    40.96       32   9209.799   0.01  # w=3.2 l=12.8
    30.72     25.6   11025.61   0.01  # w=3.2 l=9.6
   163.84     51.2   5080.426   0.01  # w=12.8 l=12.8
   163.84       64   4604.902   0.01  # w=6.4 l=25.6
   122.88     51.2   5512.804   0.01  # w=6.4 l=19.2
   655.36    102.4   2540.212   0.01  # w=25.6 l=25.6
   655.36      128   2302.451   0.01  # w=12.8 l=51.2
   491.52    102.4   2756.403   0.01  # w=12.8 l=38.4
```

```
150    2621.44    204.8   1270.106   0.01  # w=51.2 l=51.2
151    2621.44      256   1151.225   0.01  # w=25.6 l=102.4
152    1966.08    204.8   1378.201   0.01  # w=25.6 l=76.8
153
154  coupsubres :
155  #  Generated by subresgen on 10:56:15 13-5-2003
156  #   area1     area2     dist          r    decr
157        0.64      0.64      1.6     648598  0.873512  # w=0.8 d=1.6
158        0.64      0.64      3.2    1101504  0.925946  # w=0.8 d=3.2
159        0.64      0.64      6.4    1996617  0.959256  # w=0.8 d=6.4
160        0.64      0.64     25.6    7341756  0.988935  # w=0.8 d=25.6
161        2.56      2.56      3.2   324299.1  0.873515  # w=1.6 d=3.2
162        2.56      2.56      6.4   550752.1  0.925953  # w=1.6 d=6.4
163        2.56      2.56     12.8   998307.9  0.959253  # w=1.6 d=12.8
164        2.56      2.56     51.2    3670877  0.988967  # w=1.6 d=51.2
165       10.24     10.24      6.4   162149.6  0.873515  # w=3.2 d=6.4
166       10.24     10.24     12.8     275376  0.925950  # w=3.2 d=12.8
167       10.24     10.24     25.6   499154.2  0.959259  # w=3.2 d=25.6
168       10.24     10.24    102.4    1835439  0.988967  # w=3.2 d=102.4
169      655.36    655.36     51.2   20268.69  0.873515  # w=25.6 d=51.2
170      655.36    655.36    102.4      34422  0.925953  # w=25.6 d=102.4
171      655.36    655.36    204.8   62394.28  0.959257  # w=25.6 d=204.8
172      655.36    655.36    819.2   229429.8  0.988985  # w=25.6 d=819.2
173
```

When we look to the file, we first see starting from line `18` the `unit` specifications. The 2D capacitance units on line `20` and `21` are used for the capacitances definitions starting from line `63`. Note that the `colors` specification (not shown) is only used by visualization tool `Xspace`.

Starting on line `39` we find the `conductors` definitions. This is an important section, because without this we can not specify element pins for the capacitances and contacts.

Starting on line `48` we find the transistor (`fets`) definitions. Here, we see, that the bulk region of the `nenh` is connected with the substrate.

Starting on line `53` we find the `contacts` definitions. This is also an important section, because here we find a contact definition between first metal `cmf` and the substrate `@sub`. Note that the 2D capacitances or not connected to the substrate pin `@sub`, but to the ground node pin `@gnd`.

For 3D substrate resistance extraction, we need only one additional section. Starting on line `125` we find the `sublayers` definitions. These definitions specify for each substrate layer the conductivity (in S/m) and the starting position (in μm). Note that there is only one layer defined, which extends to minus infinity.

For 2D substrate resistance extraction (using the interpolation method), we need two additional sections. Starting on line `129` we find the `selfsubres` table and on line `154` we find the `coupsubres` table. By the way, these tables are generated with tool `subresgen`, by performing a number of 3D substrate resistance extractions.

10.4 Parameter File

The parameter file `param.p` is controlling the extraction, see listing below.

```
listing of file param.p
   3
   4    BEGIN sub3d            # Data for the boundary-element method
   5    be_mode           0g
   6    max_be_area       1     # micron^2
   7    edge_be_ratio     0.01
   8    edge_be_split     0.2
   9    be_window         10    # micron
  10    END sub3d
  11
  12    min_art_degree          3      # Data for network reduction
  13    min_degree              4
  14    min_res               100      # ohm
  15    max_par_res            20
  16    no_neg_res             on
  17    min_coup_cap            0.05
  18    lat_cap_window          6.0    # micron
  19    max_obtuse            110.0    # degrees
  20    equi_line_ratio         1.0
  21
  22    disp.save_prepass_image  on    # Data for Xspace
```

In the above listing, we see a `sub3d` block from line `4` to line `10`. These parameters are only used for controlling the substrate 3D extraction. The `max_be_area` (in μm^2) and `be_window` (width in μm) parameter needs always to be specified. See for an explanation of the parameters also the ”Space Substrate Extraction User’s Manual”.

On line `22`, you find a parameter for the visualization tool `Xspace`. It means, that the image of the prepass may not be cleared when starting drawing the image of the last pass. See for an explanation also the ”Xspace User’s Manual”.

10.5 Running the Extractor

The file `script.sh` is a batch file for running all the commands for this example. First, it changes the current working directory ’.’ into a project directory:

```
% mkpr -p scmos_n -l 0.1 .
```

We use a lambda value (option `-l`) of $0.1\mu m$. For the CMOS technology, we use the `scmos_n` process from the technology library. We use the mask names as defined in the `maskdata` file of the library. But, we are not using the default technology file `space.def.s` and parameter file `space.def.p` of the library. Note that the local technology file `tech.s` is a copy of the file `space.def.s`. To use the local file, it needs to be compiled with `tecc` to the format (a `.t` file) which is used by the extractor:

```
% tecc tech.s
```

Second, the layout description is put into the project database. The layout is supplied in a GDS2 file, which can be converted to internal database format with the `cgi` program:

```
% cgi oscil.gds
```

Now, we can extract a circuit description for the layout of the `oscil` cell, as follows:

```
% space3d -vF -bC -E tech.t -P param.p oscil
```

We use the verbose option (`-v`) to see what the extractor program is doing and option `-F` to perform a flat extraction. Second, we use option `-b` to perform fast interpolated substrate res. extraction, and we use option `-C` to extract also the 2D couple caps. The calculated substrate resistors are assigned between the substrate terminals and between the substrate terminals and the substrate plane (node SUBSTR).

The extracted circuit can be inspected using the `xspice` program. We use option `-a` to get alpha-numeric node names, and we use option `-u` to omit the `pbulk` and `nbulk` nodes.

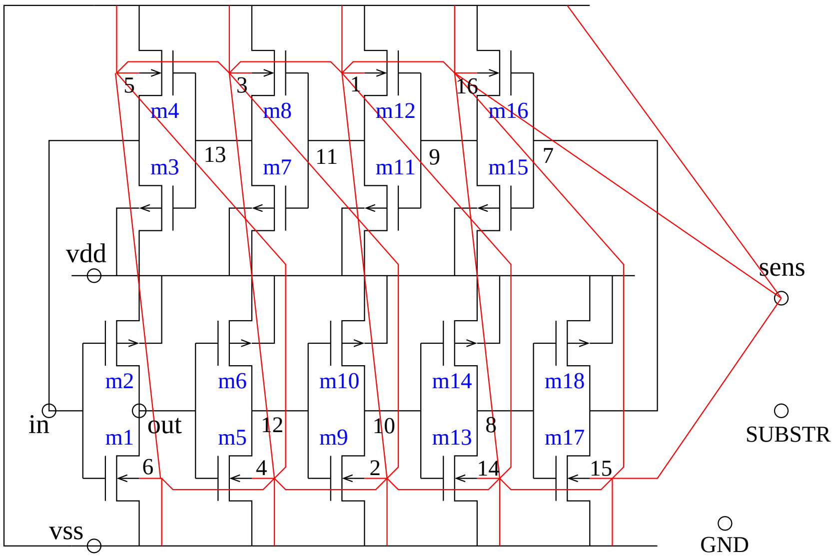
```
% xspice -au oscil
```

The output of the SPICE circuit is listed below.

```
oscil

* Generated by: xspice 2.39 25-Jan-2006
* Date: 23-Jun-06 9:09:24 GMT
* Path: /users/simon/suboscil
* Language: SPICE

* circuit oscil in out vss vdd sens
m1 out in vss 6 nenh_0 w=4.4u l=800n
m2 vdd in out vdd penh_0 w=5.2u l=800n
m3 vdd 13 in vdd penh_0 w=5.2u l=800n
m4 in 13 vss 5 nenh_0 w=4.4u l=800n
m5 12 out vss 4 nenh_0 w=4.4u l=800n
m6 vdd out 12 vdd penh_0 w=5.2u l=800n
m7 vdd 11 13 vdd penh_0 w=5.2u l=800n
m8 13 11 vss 3 nenh_0 w=4.4u l=800n
m9 10 12 vss 2 nenh_0 w=4.4u l=800n
m10 vdd 12 10 vdd penh_0 w=5.2u l=800n
m11 vdd 9 11 vdd penh_0 w=5.2u l=800n
m12 11 9 vss 1 nenh_0 w=4.4u l=800n
m13 8 10 vss 14 nenh_0 w=4.4u l=800n
m14 vdd 10 8 vdd penh_0 w=5.2u l=800n
m15 vdd 7 9 vdd penh_0 w=5.2u l=800n
m16 9 7 vss 16 nenh_0 w=4.4u l=800n
m17 7 8 vss 15 nenh_0 w=4.4u l=800n
m18 vdd 8 7 vdd penh_0 w=5.2u l=800n
c1 vdd 8 12.80448f
c2 vdd 7 12.80448f
c3 vdd 10 12.80448f
c4 vdd 9 12.80448f
c5 vdd 12 12.80448f
c6 vdd 11 12.80448f
c7 vdd out 12.80448f
c8 vdd 13 12.80448f
c9 vdd in 12.80448f
c10 vdd GND 152.1344f
r1 1 16 513.9126k
r2 1 14 2.95793meg
r3 1 2 2.773095meg
```

```
r4 1 vss 229.6561k
r5 1 3 513.9126k
r6 1 SUBSTR 39.66184k
r7 2 14 513.9126k
r8 2 vss 229.6561k
r9 2 3 2.95793meg
r10 2 4 513.9126k
r11 2 SUBSTR 39.66184k
r12 3 4 2.773095meg
r13 3 vss 229.6561k
r14 3 5 513.9126k
r15 3 SUBSTR 39.66184k
r16 4 vss 229.6561k
r17 4 5 2.95793meg
r18 4 6 513.9126k
r19 4 SUBSTR 39.66184k
r20 5 6 2.773095meg
r21 5 vss 362.2433k
r22 5 SUBSTR 35.28052k
r23 6 vss 229.6561k
r24 6 SUBSTR 36.68792k
c11 7 GND 10.52699f
c12 8 GND 6.05915f
c13 9 GND 6.05915f
c14 10 GND 6.05915f
c15 11 GND 6.05915f
c16 12 GND 6.05915f
c17 13 GND 6.05915f
c18 out GND 6.05915f
c19 in GND 9.72507f
r25 14 15 513.9126k
r26 14 16 2.773095meg
r27 14 vss 229.6561k
r28 14 SUBSTR 39.66184k
r29 sens 15 2.111757meg
r30 sens vss 6.648565meg
r31 sens 16 3.610183meg
c20 sens GND 396.8e-18
r32 sens SUBSTR 87.51464k
r33 15 vss 387.7774k
r34 15 16 2.95793meg
r35 15 SUBSTR 35.20636k
r36 vss 16 229.6561k
c21 vss GND 46.05344f
r37 vss SUBSTR 13.67904k
r38 16 SUBSTR 37.50382k
* end oscil
...
```

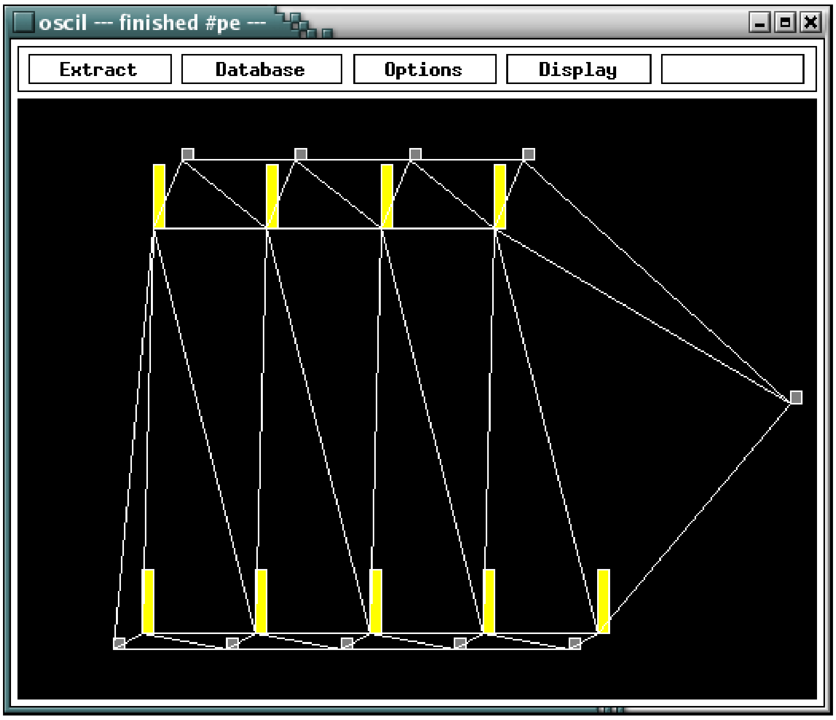
Below, you find a drawing of the listing of the extracted circuit (the capacitors are not shown).



Note that the red lines are substrate resistors. The transistor bulks and substrate contact regions are substrate terminals. There are not substrate resistors between all substrate terminals, because of the interpolation method and the used Delaunay triangulation. However, each substrate terminal has a substrate resistor to the substrate node `SUBSTR`. When you use the visualization tool `Xspace`, you can draw an image with the substrate resistors (see below). Type:

```
% Xspace -bC -E tech.t -P param.p oscil
```

Go to the ”Display” menu and click on buttons ”DrawSubTerm”, ”FillSubTerm” and ”DrawSubResistor”. To start the extraction use hotkey ’e’. To exit the visualization tool use hotkey ’q’.



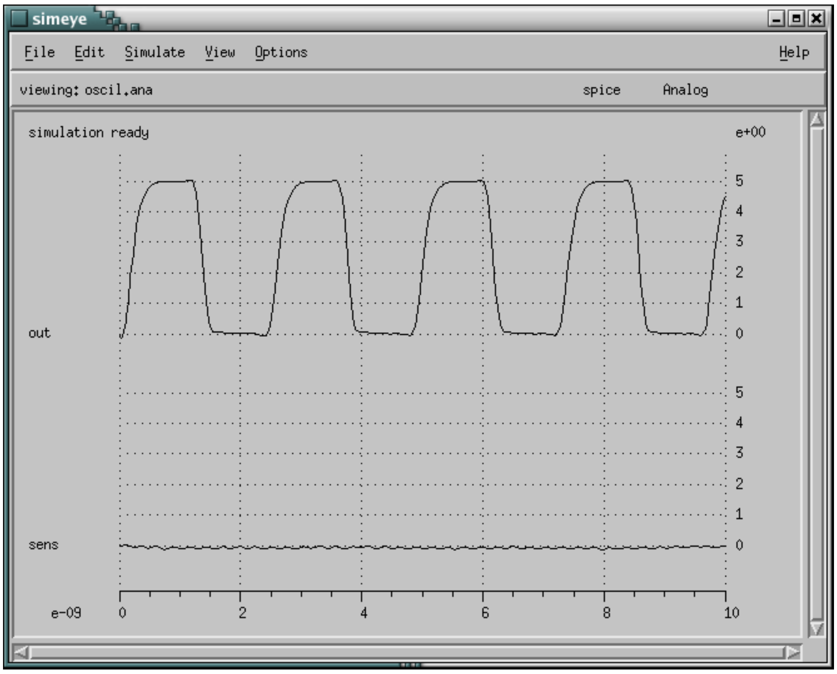
10.6 Running a Circuit Simulation

If you have a `spice3` simulator available, you can perform a spice3 simulation to inspect the noise on the ”sens” terminal node. The noise is caused by the substrate coupling effects.

You can use the simulation GUI `simeye`, to start the analog simulation. First, check the shell script `nspice` in `$ICDPATH/share/bin` to see if `spice3` is called correctly. To start `simeye`, use the following command:

```
% simeye
```

Now, click on the ”Simulate” menu and choice item ”Prepare”. Select in the ”Circuit:” field cell name ”oscil” and in the ”Stimuli:” field file name ”oscil.cmd”. Choice simulation ”Type: spice” and click on button ”Run”. You must get the following spice output result:



You can zoom-in on the ”sens” waveform. Choice the ”ZoomIn” item out of the ”View” menu. Click with the mouse pointer in the output window, lower left by the ”sens” waveform. Draw a rubber box with the mouse around the waveform and click again. To zoom-in even more, set ”Detail-ZoomON” item in the ”Options” menu. To see the individual waveform points, choice ”Values” in the ”View” menu.

To exit the program, choice item ”Exit” in the ”File” menu.

10.7 Running a 3D Substrate Extraction

By a 3D extraction, the boundary element mesh (BEM) method is used to calculate the resistances. For more details consult the ”Substrate Extraction User’s Manual”. To perform the 3D substrate resistance extraction of cell ”oscil” in batch mode, type the following command:

```
% space3d -v -BC -E tech.t -P param.p oscil
```

The `-B` option is now used in place of the `-b` option. Note that you don’t need to give the `-F` option, because substrate extractions are always done in flat mode. The output of the verbose mode is listed below.

```
Version 5.3.1, compiled on Mon Feb 20 11:51:10 GMT 2006
See http://www.space.tudelft.nl
parameter file: param.p
Flat extraction mode turned on!
technology file: tech.t
preprocessing oscil (phase 1 - flattening layout)
preprocessing oscil (phase 2 - removing overlap)
prepassing oscil for substrate resistance prepare
strip 52 152 (subtract)
strip -48 152 (add)
strip 152 252 (subtract)
strip 52 252 (add)
strip 252 352 (subtract)
strip 152 352 (add)
strip 352 452 (subtract)
strip 252 452 (add)
strip 352 500 (add)
computing substrate effects for oscil
running: makesubres -Etech.t -Pparam.p -v oscil
makesubres: See http://www.space.tudelft.nl
makesubres: makesubres -Etech.t -Pparam.p -v oscil
makesubres: parameter file: param.p
makesubres: technology file: tech.t
makesubres: prepassing oscil for substrate resistance calculate
makesubres: strip 52 152 (subtract)
makesubres: Schur dimension 460, maxorder 247
makesubres: strip -48 152 (add)
makesubres: Schur dimension 848, maxorder 423
makesubres: strip 152 252 (subtract)
makesubres: Schur dimension 424, maxorder 211
makesubres: strip 52 252 (add)
makesubres: Schur dimension 884, maxorder 459
makesubres: strip 252 352 (subtract)
makesubres: Schur dimension 460, maxorder 247
makesubres: strip 152 352 (add)
makesubres: Schur dimension 884, maxorder 459
makesubres: strip 352 452 (subtract)
makesubres: Schur dimension 176, maxorder 175
makesubres: strip 252 452 (add)
makesubres: Schur dimension 636, maxorder 423
makesubres: strip 352 500 (add)
makesubres: Schur dimension 212, maxorder 175
```

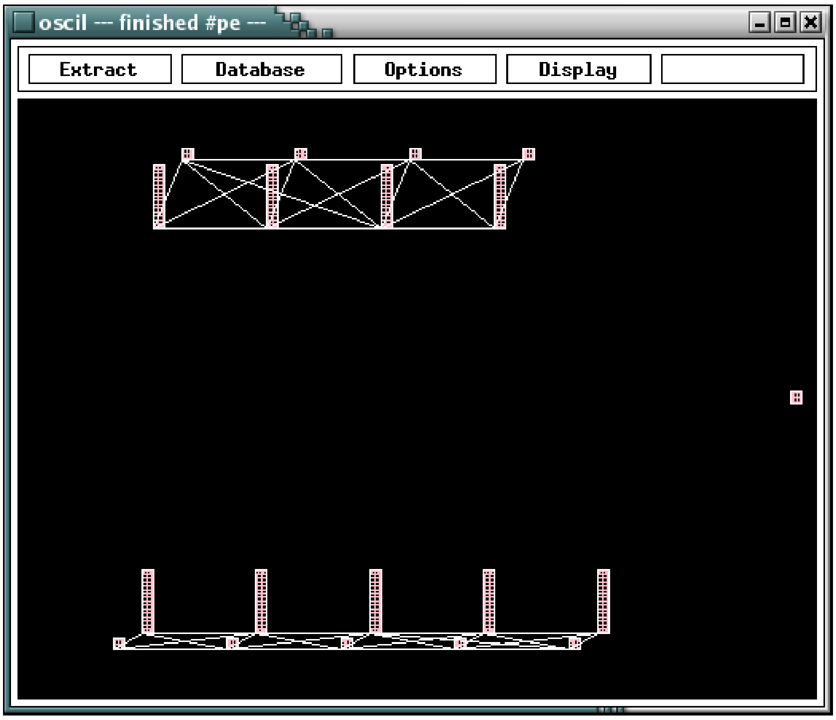
```
makesubres:
makesubres: substrateStatistics for layout oscil:
makesubres:         total num. of tiles : 72
makesubres:         substrate tiles     : 19
makesubres:         substrate terminals : 19
makesubres: overall resource utilization:
makesubres:         memory allocation  : 1.21 Mbyte
makesubres:         user time          :         7.1
makesubres:         system time        :         0.1
makesubres:         real time          :         7.3 100%
makesubres:
makesubres: makesubres: --- Finished ---
extracting oscil

extraction statistics for layout oscil:
        capacitances        : 21
        resistances         : 27
        nodes               : 23
        mos transistors     : 18
        bipolar vertical    : 0
        bipolar lateral     : 0
        substrate terminals : 19
        substrate nodes     : 11

overall resource utilization:
        memory allocation  : 0.56 Mbyte
        user time          :         0.0
        system time        :         0.0
        real time          :         9.0   0%

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

As you see above, the extraction process uses a number of passes. First, there are two preprocessing phases. In phase 1, the layout is expanded and flattened using tool `makeboxl`. In phase 2, the expanded mask data is converted with tool `makegln` into a new set gln-files, which geometric lines are sorted and the overlaps are removed. Note that the preprocessing phases are only done, when the gln-files are out of date. After this preprocess, prepass 1 is done. This pass prepares the mask data for the `makesubres` program. This program is started to compute the substrate effects (prepass 1b). The last pass is always the extraction pass. This pass reads the substrate results (stored in a ”subres” file) and combines it with the other extracted netlist elements.

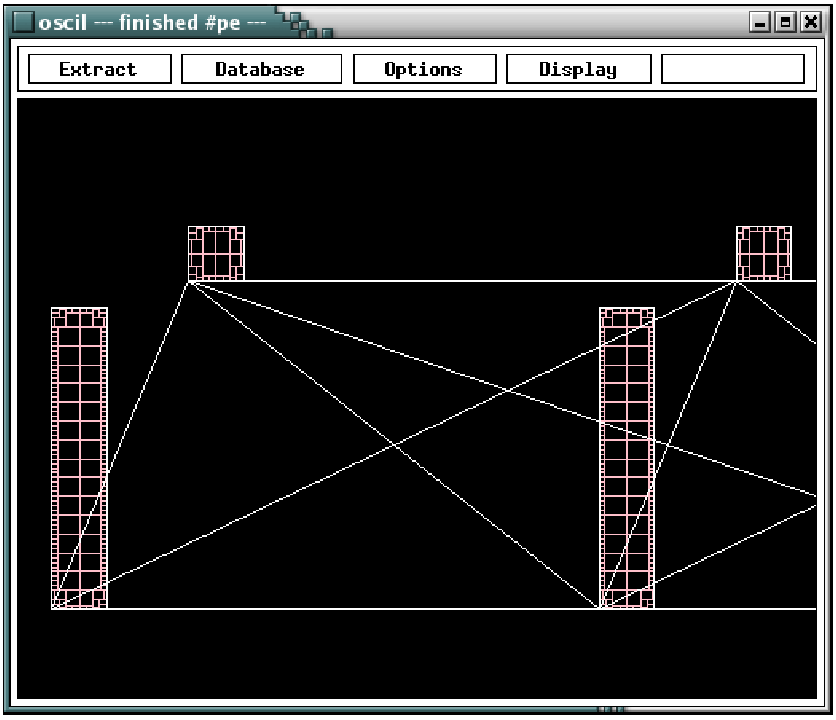
Use also the visualization tool `Xspace` and perform an interactive 3D substrate extraction, type:

```
% Xspace -BC -E tech.t -P param.p oscil
```

We want to show the following items: the boundary element mesh, the substrate terminals and the substrate resistors. Go to the ”Display” menu and click on buttons ”DrawBEMesh”, ”DrawSubTerm” and ”DrawSubResistor”. To start the extraction use hotkey ’e’. You see 19 substrate terminals. But not all terminals are connected with each other by substrate resistors (because of the boundary element window). And each substrate terminal has a very fine boundary element mesh. See the picture below.



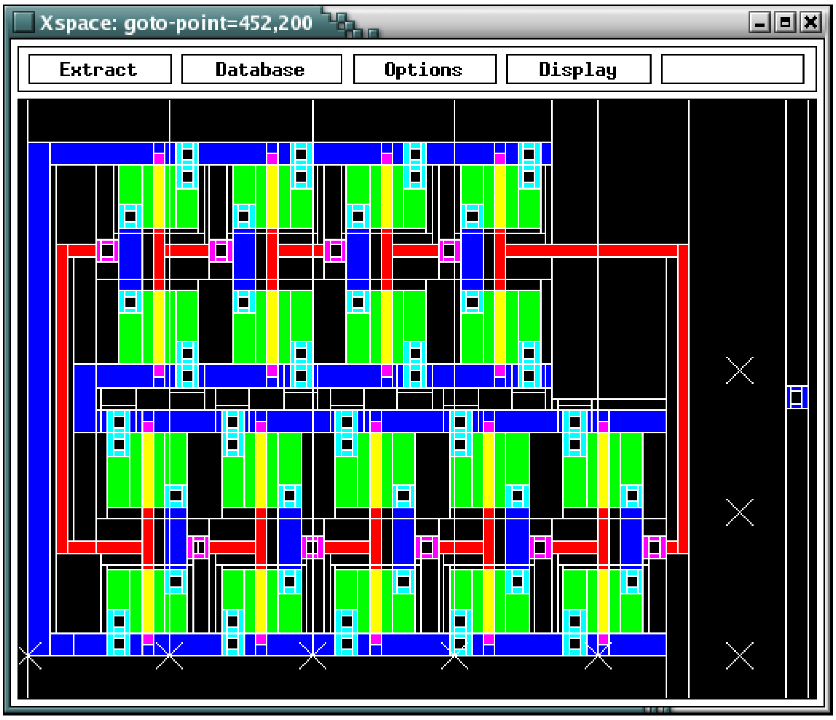
Use hotkey 'i' to zoom-in on the cursor position, to show the mesh in more detail. Use the arrow keys to pan around. Use the 'o' key to zoom-out and 'b' key to set the bounding-box view again. Use hotkey 'q', when you want to quit the visualization tool. The following picture shows a zoom-in example:



The 3D substrate method uses a boundary element window (parameter `sub3d.be_window`). In the x-direction, the layout is split into strips, which each have a width of the `be_window`. A good choice of the window size is important, because the strips give extra tile splitting. When you look to the verbose output of the extractor, you see information about the strip positions. Because the window size is 10μ and $\lambda = 0.1\mu$, the window size is 100λ units. The bounding box of the design is from $xl = -48$ to $xr = 500$ units. Between each two strips are results calculated. The double calculated results of intermediate single strips are subtracted. You can use the visualization tool to see the strips by looking to the tile splitting. Start the tool with the following two display options added to the command line:

```
% Xspace -BC -E tech.t -P param.p -Sdisp.pair_to_infinity \
                                  -Sdisp.coord_in_dbunits oscil
```

Before starting the extraction, go to the "Display" menu and click on buttons "PairBoundary" and "DrawTile". To stop displaying the image after the first prepass, go to the "Extract" menu and click on button "pause afterpp". Now, hit the extract key 'e' to start the extraction. The result is shown below:



Note that not all strip edges can be shown, because no tile edges are generated. You see on the design top side some tile edges to infinity, which are from the strips.

To identify the strips, i have added a number of coordinate crosses to the image. To add coordinates in database units, you must set parameter `disp.coord_in_dbunits`. Because, default `Xspace` is using an internal unit, whereby all units are multiplied by 4. I shall explain, how you can easy enter coordinate crosses. Use hotkey 'g' and type the x-coordinate number and type a minus sign to negate it. Type 'Enter' to finish the input. Note that we have not entered the y-coordinate. If you want to enter (or change) an y-coordinate, you must begin the number with hotkey ','. For more information, see also the "Xspace User's Manual".

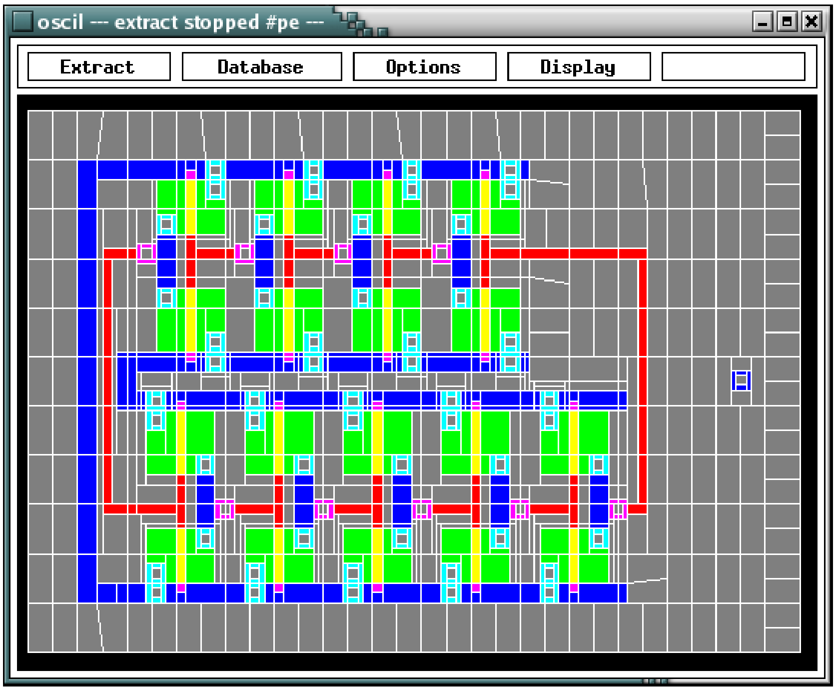
10.8 Running a BEM/FEM Substrate Extraction

With this new method, you can combine the 3D BEM substrate extraction method with a 2D FEM resistance extraction method for the top of the substrate. This makes it possible, to model the regions of the top of the substrate with different conductance values.

To use this method, you must add `wafer` definitions to the technology file. See technology file `tech3.s` (a small part is listed below).

```
listing of file tech3.s
        ...
        set bem_depth 6

        wafer : pdef  : 1000 6 2
        wafer : pcont : 2000 6 2
        wafer : nsd   : 2000 1 2 : restype=n
        wafer : nsd   : 2000 5 2
        wafer : nchan : 2000 6 2
        ...
```

Because this method is modeling the complete substrate, you get a big number of substrate terminals. The node elimination of the resulting network cost (at this moment) a lot of computer time. Below, the visualization tool gives an impression:



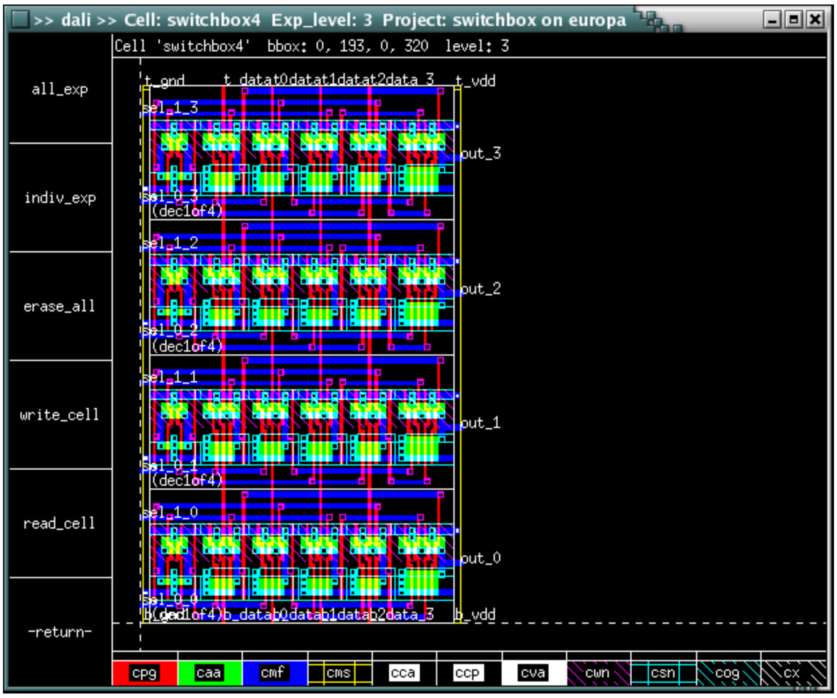
Note that the method is also using the `be_window` size for a substrate bounding box around the design. And that the strips give also extra tile splittings in the y-direction.

11 switchbox Example of Space Tutorial

11.1 Introduction

In this example, we will be studying a switchbox design. We will see how the Space system can be used. See also the ”Space Tutorial” document.

The layout looks as follows, using the layout editor `dali` (see `icdman dali`):



11.2 Files

This tutorial is located in the directory `$ICDPATH/share/demo/switchbox`. Initially, it contains the following files:

`README:` A file containing information about the demo.

`switchbox4.gds:` The GDS2 file of the switchbox layout.

`switchbox4_f.gds:` The GDS2 file of the flat switchbox layout.

`swbox_ref.spc:` SPICE file of the switchbox reference circuit.

`script.sh:` A file containing the commands for executing all steps of the demo in sequence.

11.3 Running a Hierarchical Extraction

Before we can run a hierarchical extraction on the ”switchbox” layout, we need a project directory. The file `script.sh` is a batch file for running all the commands for this example, which you can use. First, it changes the current working directory ’.’ into a project directory:

```
% mkpr -p scmos_n -l 1 .
```

We use a lambda value (option `-l`) of $1\mu m$. For the CMOS technology, we use the `scmos_n` process from the technology library.

Second, the layout description is put into the project database. The layout is supplied in a GDS2 file, which can be converted to internal database format with the `cgi` program:

```
% cgi switchbox4.gds
```

Third, we can list the contents of the project database. To get a hierarchical cell listing, use the `dblist` program as follows:

```
% dblist -h
```

```

layout:
1 - switchbox4      (4)
    2 - dec1of4          4 (6)
        3 - nan4rout         1 (0)
        3 - nan3             4 (0)
        3 - dubinv           1 (0)

circuit:

floorplan:

```

You see, that the layout of the ”switchbox” design is hierarchical. There are 3 hierarchical levels. The top cell is `switchbox4`, it calls 4 times sub cell `dec1of4`. The cell `dec1of4` has also 3 sub cells (the cell `nan3` is used 4 times). To inspect the layout of the design, use the layout editor/viewer `dali`. Type the following command:

```
% dali
```

Click on the ”DB_menu” menu item and use the ”read_cell” command to read the different cells. To see more hierarchical levels of the cell, use the number keys. Type for example ’3’ to expand the cell to 3 levels deep.

Now, we can extract a circuit description for the layout of each cell. Because the layout is hierarchical, we can perform a hierarchical extraction of the `switchbox4` top cell. Each cell is individual extracted, because the flat option is not used. We start the extractor with the verbose option, as follows:

```
% space -v switchbox4
```

Note that `space` is using the default technology file `space.def.s` and the default parameter file `space.def.p` of the `scmos_n` example process. The output of the hierarchical extraction is listed below.

```
Version 5.3.1, compiled on Fri Feb 03 12:45:53 GMT 2006
See http://www.space.tudelft.nl
parameter file: $ICDPATH/share/lib/process/scmos_n/space.def.p
technology file: $ICDPATH/share/lib/process/scmos_n/space.def.t
```

```
extract hierarchy of switchbox4
preprocessing nan4rout (phase 1)
preprocessing nan4rout (phase 2 - removing overlap)
extracting nan4rout

extraction statistics for layout nan4rout:
        capacitances        : 0
        resistances         : 0
        nodes               : 10
        mos transistors     : 8
        ...

preprocessing nan3 (phase 1)
preprocessing nan3 (phase 2 - removing overlap)
extracting nan3

extraction statistics for layout nan3:
        capacitances        : 0
        resistances         : 0
        nodes               : 8
        mos transistors     : 6
        ...

preprocessing dubinv (phase 1)
preprocessing dubinv (phase 2 - removing overlap)
extracting dubinv

extraction statistics for layout dubinv:
        capacitances        : 0
        resistances         : 0
        nodes               : 6
        mos transistors     : 4
        ...

preprocessing dec1of4 (phase 1)
preprocessing dec1of4 (phase 2 - removing overlap)
extracting dec1of4

extraction statistics for layout dec1of4:
        capacitances        : 0
        resistances         : 0
        nodes               : 51
        mos transistors     : 0
        ...

preprocessing switchbox4 (phase 1)
preprocessing switchbox4 (phase 2 - removing overlap)
extracting switchbox4

extraction statistics for layout switchbox4:
        capacitances        : 0
        resistances         : 0
        nodes               : 42
```

```
        mos transistors     : 0
        ...

overall resource utilization:
        memory allocation  : 0.233 Mbyte
        user time          :         0.0
        system time        :         0.0
        real time          :         7.1   0%

space: --- Finished ---
```

We make again a hierarchical cell listing of the project database.

```
% dblist -h
```

```

layout:
1 - switchbox4      (4)
    2 - dec1of4           4 (6)
        3 - nan4rout          1 (0)
        3 - nan3              4 (0)
        3 - dubinv            1 (0)

circuit:
1 - switchbox4      (4)
    2 - dec1of4           4 (6)
        3 - dubinv            1 (4)
        3 - nan3              4 (6)
        3 - nan4rout          1 (8)

floorplan:
```

We see, that the extracted circuit cells have also a hierarchical structure.

11.4 Running a Circuit Comparison

The following demonstrates the use of the circuit comparison program `match`.

First, add a reference circuit description for the ”switchbox” circuit to the database using the program `cspice`.

```
% cspice swbox_ref.spc
```

```
File swbox_ref.spc:
Parsing network: swbox_ref
```

This reference circuit is a flat circuit description of the ”switchbox”. Compare this reference circuit with the hierarchical description, type:

```
% match swbox_ref switchbox4
```

```

match: Succeeded.

```

If you want, you can use the `-fullbindings` option to get more information.

You can also perform a flat extraction of the ”switchbox4” cell and do the compare again. This must also give a succesfull match.

```
% space -F switchbox4
% match swbox_ref switchbox4
```

Second, use `cgi` to put a flat version of the ”switchbox” layout into the database and extract the circuit and do the compare again.

```
% cgi switchbox4_f.gds
% space switchbox4_f
% match switchbox4 switchbox4_f
```

This must also give a succesfull `match` result. Now, use the program `match` to compare the extracted circuit against the reference circuit.

```
% match swbox_ref switchbox4_f
```

```
match: Succeeded.
```

The result shows that the circuits are identical.

We use the flat layout, because it is more easy to change. Now, make an error in the layout of ”switchbox4_f” using `dali` (e.g. remove some metal) and run the programs `space` and `match` again.

```
% dali switchbox4_f
% space switchbox4_f
% match swbox_ref switchbox4_f
```

```
match: Failed.
```

The result shows that the circuits are not more identical.

Now, use the option `-fullbindings` to see which network parts are matched.

```
% match -fullbindings swbox_ref switchbox4_f
```

11.5 Layout Back-Annotation

You can use the layout back-annotation option `-x` of `space` to see with the layout viewer `dali` which conductors are high-lighted. Using `match` and after that the high-light tool `highlay`, you can write a cell ”HIGH_OUT” to inspect the unmatched condcutors. See also section 9 of the ”Space Tutorial” document.