How Space does Extract Asymmetric Field Effect Transistors

S. de Graaf

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

Report EWI-ENS 09-03
November 13, 2009

Last revision: November 16, 2009.

1. INTRODUCTION

Field Effect Transistors (FETs) are normally symmetric. That means that the drain and source regions may be interchanged. For both regions the same conductor is used.

Asymmetric FETs, however, use different conductors for drain and source regions. Therefor the drain and source regions may not be interchanged.

To make the specification of asymmetric FETs possible in the technology file, there is added an optional source mask specification to the FET element definition. This mask must be given after the drain mask and must start with a slash in front.

See also the *tecc* manual page for the syntax of the FET element definition.

Note that the source and drain areas for asymmetric FETs may not touch or overlap each other. If it happens, then a warning message is given. The w/l attributes are in that case wrong calculated.

The *tecc* and *space* programs are changed for the asymmetric FET implementation. The new version of *space* can only read the new technology file format. It can also read old technology files.

2. TECHNOLOGY FILE EXAMPLE

The technology file below contains two definitions for asymmetric fets. To distinct them from the symmetric ones the condition list has got an extra "cog" mask. Also the new "caa2" conductor, used for the source pin, has got the "cog" mask to distinct it from the "caa" conductor.

```
new : caa cog : caa2

conductors :
  # name    : condition     : mask : resistivity : type
    cond_mf : cmf           : cmf  : 0.045       : m    # first metal
    cond_pg : cpg           : cpg  : 40          : m    # poly interconnect
    cond_pa : caa !cpg !csn !cog : caa  : 70     : p    # p+ active area
    cond_na : caa !cpg  csn !cog : caa  : 50     : n    # n+ active area
    cond_pa2: caa !cpg !csn  cog : caa2 : 63     : p    # p+ active area 2
    cond_na2: caa !cpg  csn  cog : caa2 : 45     : n    # n+ active area 2
    cond_well : cwn         : cwn  : 0           : n    # n well

fets :
  # name : condition         : gate d/s     : bulk
    nenh : cpg caa  csn !cog : cpg caa      : @sub  # nenh MOS
    penh : cpg caa !csn !cog : cpg caa      : cwn   # penh MOS
    nenh2: cpg caa  csn  cog : cpg caa/caa2 : @sub  # asymmetric nenh MOS
    penh2: cpg caa !csn  cog : cpg caa/caa2 : cwn   # asymmetric penh MOS

contacts :
  # name   : condition     : lay1 lay2 : resistivity
    cont_p : ccp cmf cpg   : cmf  cpg  : 100   # metal to poly
    cont_a : cca cmf caa !cpg cwn !csn !cog    # metal to active area
           | cca cmf caa !cpg !cwn csn !cog : cmf caa  : 100
    cont_a2: cca cmf caa !cpg cwn !csn  cog
           | cca cmf caa !cpg !cwn csn  cog : cmf caa2 : 100
```

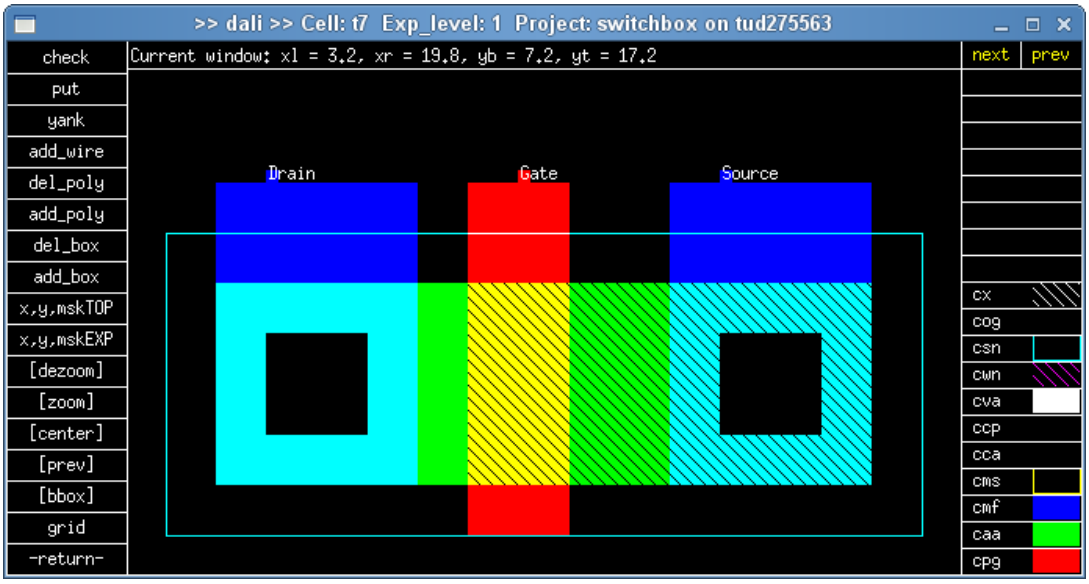
For example, compile the technology file as follows:

```
% tecc space.def2.s

-- keys: cpg caa cog csn cmf cwn cca ccp
-- keys2:
-- number of keys: 0 + 8 (8)
-- number of keys2: 0 + 0 (0)
-- number of key slots: 256 (1)
-- maximum number of elements per key slot: 6 (0)
-- maximum number of additional conditions per element: 0
-- average number of additional conditions per element: 0.000

-- add. cond.  :  0
   no. of elem.: 17 (17)
```

3. EXTRACTION EXAMPLE



For example, extract the above layout cell "t7" as follows:

```
% space -E space.def2.t t7
```

For netlisting, i used the following setup file:

```
% cat xspicerc
model xxx nenh2 nmos ()
model yyy penh2 pmos ()
```

Spice netlist example:

```
% xspice -au t7
t7

* Generated by: xspice 2.51 06-May-2009
* Date: 16-Nov-09 9:42:05
* Path: /home/simon/CACD/@demo/demo/switchbox
* Language: SPICE

* circuit t7 Gate Drain Source
m1 Drain Gate Source SUBSTR xxx w=4u l=2u
* end t7

*.model xxx nmos
```

4. EXTRACTION OPTIONS

FETs are default extracted with the not separate d/s boundaries mode.

There is a secret option to switch the separate d/s boundaries mode on (-%J).

Default, if one d/s terminal is missing, it is added and connected to the drain terminal. If this happens a warning message is given. For the -%J option no terminal is added.
Note that for an asymmetric FET, also default no terminal is added.

If there are no d/s terminals at all, a message is given, but no terminals are added.

Note that parameter "omit_incomplete_tors" can be used to skip incomplete FETs.

Default, if there are more than two d/s terminals, then one of the unconnected terminals is silently removed and only two are used for the drain and source. (Note that the first outputted net gets always the drain terminal.) For the -%J option no terminals are removed, but a warning message is given.

Note that netlisting can be unexpected, when more than one source (or drain) terminal for a FET is outputted. The netlist tools shall only connect the last found net terminal to the FET. (Two or more drains can only happen for asymmetric FETs.)

The x,y coordinates of FETs can be outputted (as attributes) with option -t or with parameter "component_coordinates=on". This option is also set with the backannotation option -x.

5. CHANGED EXTRACT BEHAVIOUR AND OPTIMALISATIONS

For option -%J, the first outputted net gets now also always the drain terminal.

Fix for default mode, if there are more than two d/s terminals, the last unconnected source terminal can now be removed.

Added for default mode, if two (or more) source terminals for a FET are outputted a warning message is given.

Added test in outNode, unconnected source terminals are not removed in fine network mode (-%f). (Note that also parameter "network_reduction=off" can set the fine network mode.) Added test in outNode, to remove unconnected nodes/nets when not in fine network mode.

Removed secret option -%d, which could be used to omit the output of w/l attributes for FETs.

5.1 Extract Optimalisations

No partAdd's are more done for the d/s boundaries. (That was already the case, when the -%J mode was used.) The created nodeTorLinks have no function than giving the connected nodes free. But, because the d/s boundaries have also there own subnodes, the subnodeDel of these subnodes frees after that the connected nodes. Note that nodeTorLinks are only used for gate and bulk connections, to hold the nodes till the FET is outputted.

The number of boundary points is changed. The number is now always two (or zero when the boundary is closed). The "nr" field is removed, because there is another way to detect a closed boundary. Less boundary points gives also less computer time, but there is also not more looked to the boundary points in the output stage to detect closed boundaries. (Note: it is even possible to remove all points and to put the two points in the boundary structure, then a flag must be added to sign its close.)

Two functions used by function subtorJoin (addDsBdr and joinDsBdr) are removed. There was a lot work done to detect a closed boundary after the join of boundaries. I am sure, that boundaries cannot get closed by a call to subtorJoin.

The used subnodeJoin tests are also a little optimized.