Edge Capacitance Compensation
by
Lateral Capacitance Extraction

S. de Graaf

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

Report EWI-ENS 14-04
March 13, 2014

Last revision: March 20, 2014.

1. INTRODUCTION

See also the "Space User’s Manual", section 2.5.4 "Edge Capacitances".

Two metal lanes (length 40 um, width 0.5 um, bottom-height 0.7 um, thickness 0.5 um) laying in a dielectric medium with a relative permittivity of 3.86, have by different spacing distances, doing a 3D capacitance extraction (be_mode=0c, max_be_area=1, be_window=40, min_coup_cap=0), the following couple capacitances (fF):

```
spacing:      0.5 um   1 um     2 um     4 um     8 um     16 um    32 um
cap(P1,P2) : 2.246193 1.173493 0.510664 0.176731 0.051013 0.011220 0.001236
cap(P1,GND): 3.437741 3.797202 4.199253 4.475913 4.594348 4.633504 4.643457
cap(P2,GND): 3.437741 3.797202 4.199253 4.475913 4.594348 4.633504 4.643457
cap of 1 metal lane = 4.644692 fF.
```

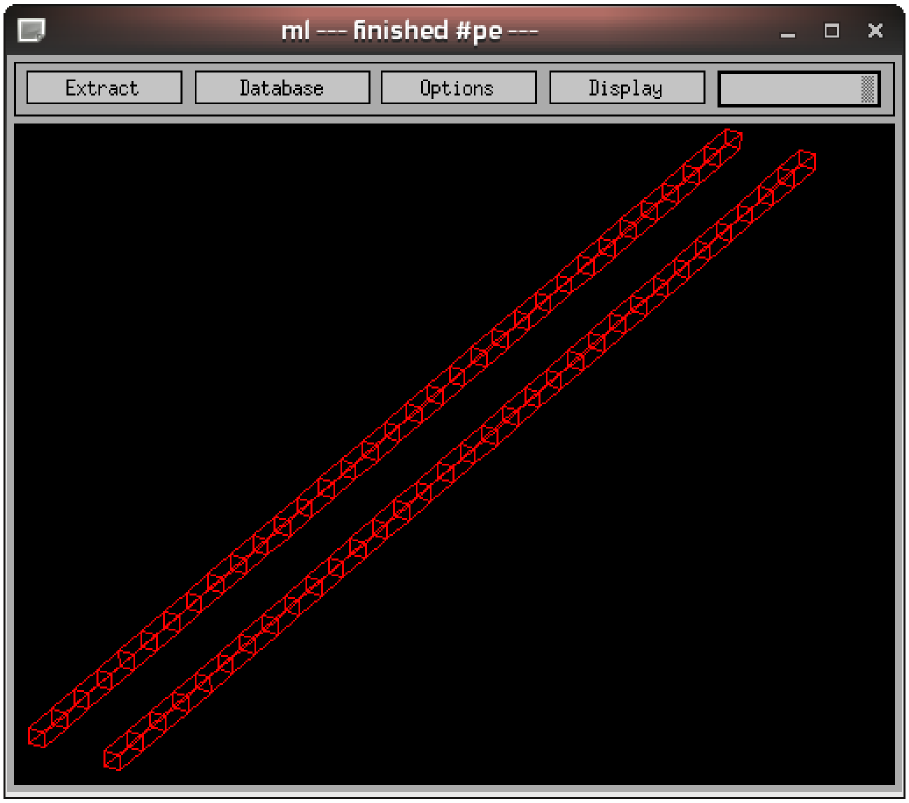
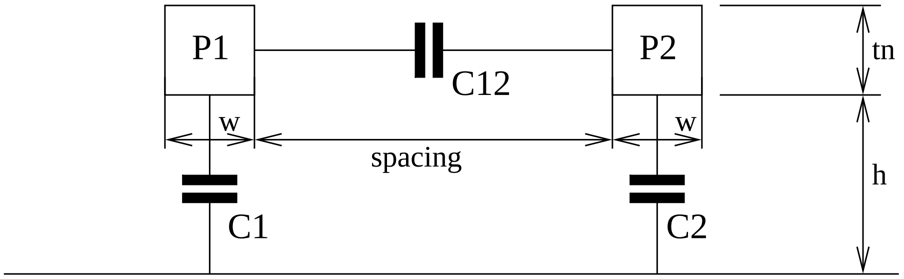


Fig.1 Two metal lanes with a spacing of 2 um.

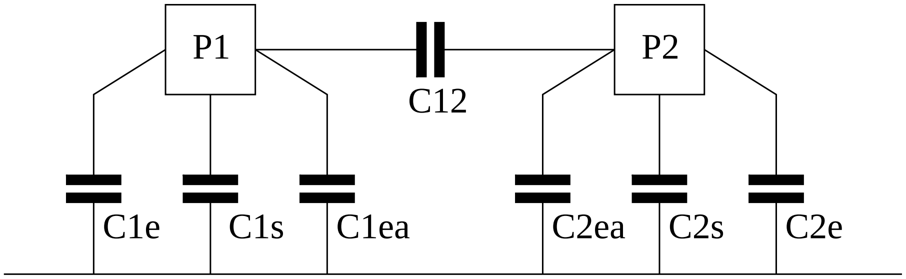
You see that the lateral capacitance between P1,P2 is higher by a smaller spacing. From figure 1, you see that a lane is split in 130 faces (the top of a lane in 32 faces), because of the max_be_area of 1 um^2. A bigger be_window than 40 um does not change the calculated capacitance values any more. However, using a smaller window has some effect on the calculated values. Using another max_be_area has also effect on the calculated capacitance values. The capacitance to the ground plane (GND) is for both lanes P1 and P2 equal.

2. 3D AND 2D CAP MODEL

The 3D cap model:



The 2D cap model:



In the 2D cap model you have edge-surface caps (C1e, C1ea, C2e, C2ea) and surface caps (C1s, C2s) and the lateral cap (C12).

Without the lateral cap, when the spacing distance is large enough, you can say:

```
C1 = C1s + 2*C1e          C2 = C2s + 2*C2e         C1 = C2 = 4.644692 fF
```

With a lateral cap, when the spacing is for example 2 um, you can say:

```
C1 = C1s + C1e + C1ea     C2 = C2s + C2e + C2ea    C1 = C2 = 4.199253 fF
=> C1ea = C1e - (4.644692 - 4.199253) = C1e - 0.445439 fF
```

Thus, the edge-surface cap. C1e must be compensated for 0.445439 fF by a spacing of 2 um. We can make the following table:

spacing	C1	comp_value of C1e
0.5	3.437741	1.206951
1.0	3.797202	0.847490
2.0	4.199253	0.445439 <=
4.0	4.475913	0.168779
8.0	4.594348	0.050344
16.0	4.633504	0.011188
32.0	4.643457	0.001235
64.0	4.644692	0.000000

To get a list of edge-surface cap values, we need to know C1s. On the following page you see a part of the lateral cap compensateEdgeCap code.

```
void do_latcap (capElem_t *lced, coor_t len, double d)
{
    snA = begTile -> cons[lced -> pCon];
    snB = endTile -> cons[lced -> nCon];
    lcap = calc_latcap (lced, d);
    capAdd (snA, snB, lcap * len, lced -> sortNr);

    if (compensate_lat_part > 0) {
        compensateEdgeCap (lced, lcap, d, lced -> pCon,
                                 snA, begTile, begTileNext, len, begElem);
        compensateEdgeCap (lced, lcap, d, lced -> nCon,
                                 snB, endTile, endTilePrev, len, endElem);
    }
}

void compensateEdgeCap (capElem_t *lced, double lcap, double dist, int lcon,
   subnode_t *subn, tile_t *tile, tile_t *tileAdj, double len, elem_t **elem)
{
    totEdgeCap = 0; k1 = k2 = -1;
    for (k = 0; elem[k]; k++) {
        if (elem[k] -> type == EDGECAPELEM) {
            eced = &elem[k] -> s.cap;
            if ((eced -> pCon == lcon ||
                (eced -> nCon == lcon && eced -> nOccurrence == EDGE))
                    && eced -> sortNr == lced -> sortNr) {
                k2 = k; if (k1 < 0) k1 = k;
                totEdgeCap += eced -> val;
                eced -> done = 1;
            }
            else eced -> done = 0;
        }
    }
    if (k1 < 0) return; /* no edge caps found */

    lcapPart = lcap * compensate_lat_part;
    if (totEdgeCap < lcapPart) lcapPart = totEdgeCap;

    for (k = k1; k <= k2; k++) {
        eced = &elem[k] -> s.cap;
        if (!eced -> done) continue;
        if (eced -> mval) { /* distance/cap pair(s) */
            if (lced -> nCon != lced -> pCon) continue;
            comp_value = eced -> val - edgecapval (eced -> mval, dist);
        }
        else comp_value = (eced -> val / totEdgeCap) * lcapPart;
        esubn = ...;
        capAddNEG (subn, esubn, comp_value * len, eced -> sortNr);
    }
}
```

You see that compensate is not done for unequal ’lced’ nCon and pCon! That the value of ’compensate_lat_part’ only is used for edge caps which don’t have distance/cap pairs! Note that in the 5.4.5 version of *space* parameter ’compensate_lat_part’ is default ’0’!

3. 2D CAP XY-PAIRS

See the "Space User's Manual" section 3.4.13, also called distance,capacitivity pairs. A "space.def.s" technology file, for example, has defined the following xy-pairs:

```
capacitances: # edge-surface caps   : dist1 value1  dist2 value2
   ecap_M1_M2 : !M1 -M1 M2 : -M1 M2 : 8.64 0.067861 17.28 0.08
   ecap_M2_M1 : !M2 -M2 M1 : -M2 M1 : 8.64 0.11132
```

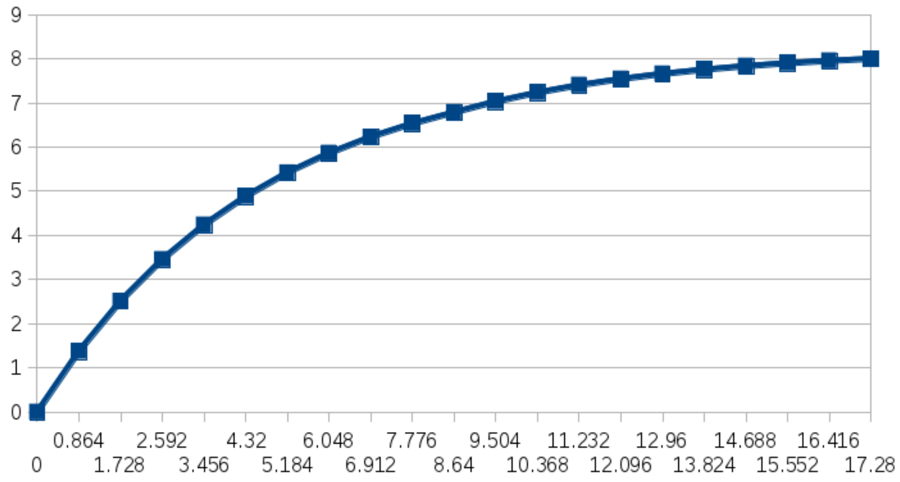
This gives the following xy-pairs table in the "space.def.t" file:

```
3 #--- x --- #--- y ---    #--- a ---    #--- b ---    #--- p ---
0.000000e+00 0.000000e+00 8.000000e-11 1.000000e+00 2.182411e+05
8.640000e-06 6.786100e-11 8.217143e-11 1.147728e+00 2.182411e+05
1.728000e-05 8.000000e-11 8.000000e-11 0.000000e+00 0.000000e+00
2
0.000000e+00 0.000000e+00 1.113200e-10 0.000000e+00 0.000000e+00
8.640000e-06 1.113200e-10 1.113200e-10 0.000000e+00 0.000000e+00
```

The following function "edgecapval" is used for the compensation:

```
double edgecapval (xyEl_t *xy, double dist)
{
   while (xy -> next && dist >= xy -> next -> x) xy = xy -> next;
   return (xy -> a * (1.0 - xy -> b * exp (-dist * xy -> p)));
}
```

When there only is one distance,value pair you see that compensation does not work! Because always the value of xy -> a is returned (xy -> b is 0.0). For the edge-surface cap "ecap_M1_M2", which has two distance,value pairs the following compensation are done. For 'x' distance lower than 8.64e-06 the 1-st record (the return value is between 0 and 6.7861e-11). For 'x' distance lower than 17.28e-06 the 2-nd record (the return value is between 6.7861e-11 and 8e-11). And, for 'x' distance equal and higher than 17.28e-06 the 3-rd record (the return value is always 8e-11). See figure:



4. NEW CHANGES

This is new for the 5.4.6 distributed version of the *space* and *space3d* program.

The default value of *space* parameter ’compensate_lat_part’ is changed back to ’1.0’ (100% compensation). This change was made in "scan/getparam.c":

```
    compensate_lat_part = paramLookupD ("compensate_lat_part", "1");
```

Thus, default there is always 100% compensation of edge-edge and edge-surface capacitances by lateral capacitance extraction. If you don’t want to have compensation, set the value of parameter ’compensate_lat_part’ to ’0’. This can be done in the "space.def.p" parameter file or on the command line using option **-S**. For example:

```
    % space -F -Cl -Scompensate_lat_part=0 switchbox4
```

Also the code of function ’compensateEdgeCap’ is changed. Now there is always compensation for the found edge capacitances. Also compensation for a single found distance/cap pair and also for distance/cap pairs by unequal ’lced’ nCon/pCon conductor pins. See following part of code from "extract/latcap.c":

```
void compensateEdgeCap (capElem_t *lced, double lcap, double dist, ...)
{
    ...
    lcapPart = lcap * compensate_lat_part;
    if (totEdgeCap < lcapPart) lcapPart = totEdgeCap;

    for (k = k1; k <= k2; k++) {
        eced = &elem[k] -> s.cap;
        if (!eced -> done) continue;
        comp_value = 0;
        if (eced -> mval) { /* distance/cap pair(s) */
           if (lced -> nCon == lced -> pCon)
              comp_value = eced -> val - edgecapval (eced -> mval, dist);
        }
        if (comp_value <= 0)
            comp_value = (eced -> val / totEdgeCap) * lcapPart;
        ...
        capAddNEG (subn, esubn, comp_value * len, eced -> sortNr);
    }
}
```

See also the "Space User’s Manual", section 3.4.13 "The capacitance list", how to define distance,capacitivity pairs in the "space.def.s" technology file.
Note, however, that you don’t need to define distance,capacitivity pairs. That is only needed for a more detailed compensation. Normally, you define one (maximum) edge capacitivity value and use the standard compensation procedure.

Note that the last capacitivity value in a distance,capacitivity pairs list must be the maximum capacitivity value. This value is used for the last given distance and all higher distances.