GDS2M

Andreea Alexandru, Gabriela Ciuprina, Sorin Lup

8 March 2013

Contents

1. Purpose of this document

This is the documentation of the GDS2M software tool for preprocessing MEMS devices. The facilities and functions of GDS2M, explanations as to how to create the input files and many examples are shown, for a complete understanding of the program.

2. What is GDS2M?

GDS2M is an effective software tool that extracts geometric information of MEMS (Micro-Electro-Mechanical Systems) and other IC devices and exports it to data structures in MATLAB [1]. Its main objectives are to ease the access and altering of certain geometric aspects, to provide a link between the files obtained from the design process and the testing and modelling software tool, before the prototype is approved for fabrication.

GDS2M is structured in two main parts. The first part is dedicated to obtaining all the characteristic information for the switch, along with the 2D and 3D representation, while the second part links the information to *chamy*, an in-house tool developed in the frame of the Chameleon-RF European research project [2], for the RF simulation of the constructed switch.

2.1 Input files

The necessary input files for GDS2m are the lyn files (extracted from KLayout [3], which store the layout and the 2D information of the masks used in the switch), the technology file (which offers 3D information about de vertical disposition of the switch, about the material and containment of holes) and the problem file (needed for the RF simulation, which contains information about the terminals and parameters).

2.2 Output

The first part of the program exports the following data structures:

- 'Obj' - data structure which contains the geometric and material information for each object in each layer from the switch
- 'LayersExt' - data structure which contains the geometric and material information for the exterior layers
- 'xmax' and 'zmax' - dimensions of the domain, used as borders in *chamy*

These information are used when creating the device and layout in *chamy* (along with the problem file).

The second part of the program exports the grid created with the given information and ... .

The functions in GDS2M can :

- extract the geometric information from the lyn files and create graphic 2D representations of the objects

- extract the 3D information from the technology file and together with the information obtained from the lyn files, to create graphic 3D representations of the switch
- create a more complex structure (by splitting the objects in 'bricks') and be able to export it in *chamy*
- analyze and process the extracted information (ex : to fill/eliminate the holes in the membrane, to approximate the surface of each object with a union of rectangles with edges parallel to the axes etc.)
- reconstruct the switch from bricks and add terminals to it

3. MEMS

During the past 20 years, technological research was aimed towards minimizing the circuits and working in microwave and millimetre waves areas. The development of microfabrication and processing techniques favoured the use of MEMS circuits. Bulk production and reduced dimensions brought increasing interest in the MEMS area and especially in the radio frequency (RF) MEMS. The term RF MEMS refers to the design and fabrication of MEMS for RF integrated circuits, which is different from the traditional MEMS devices operating at RF frequencies [4]. One of the applications of RF MEMS *that have attracted much interest* are switches, widely used in the communications area (satellite communication: 12-35 GHz, radar systems: 5-94 GHz, mobile communication systems: 0.8 - 6 GHz, instrumentation systems: 0.01 - 50 GHz). The main advantages of the RF MEMS switches over the currently used devices (PIN diodes and field effect transistors) are low insertion losses, high isolation, null power consumption and reduced costs [5].

The final outcome of the MEMS design process consists of a file describing the masks that will be used during the fabrication, accompanied by a technology documentation describing the layers and the materials used. If further research needs to be carried out and new prototype tools need to be developed, then it is useful to easily access the geometrical and material information from the files that designers provide. The main contribution of GDS2M is related to this preprocessing step.

The available file formats for MEMS fabrication include GDS II (Graphic Database System), OASIS (Open Artwork System Interchange Standard), CIF (Caltech Intermediate Form), DXF (Drawing Exchange Format) and Gerber (RS-274X) . The most commonly used stream format is .GDSII, preferred because of its binary format and small file size. The information provided by the GDSII format are the two-dimensional geometrical shapes, text labels and database units, grouped by labels in a hierarchical form.

The small number of available (and free) software tools capable of extracting the information from a .GDS II file and exporting it to other programming environments has determined us to create GDS2M, a MATLAB based tool that satisfies the requirements above.

4. KLayout

There are several software tools for viewing and editing .GDS files. We mention KLayout, CleWin, Layout Editor, Koala, Java GDS [6]. KLayout is a free .GDS viewer and editor that supports *Ruby* scripting, as well as manual manipulation of the .GDS files.

The layers in the .GDS file represent the masks used. By using the facility Trace Net in KLayout, the user can export the 2D information of each object from the layers in a .lyn file, which uses the xml format.

4.1 Lyn files

KLayout offers the facility of exporting the geometrical information in lyn files, which use the XML format. These files can be obtained by using the Trace Net function, for each object in the layer. The lyn files contain the following fields : <nets> and <net>, with the <name> of the net and <top_cell>, the name of the cell, <layout> the address where the file is exported to, <dbu> the database units in which the coordinates are scaled, <complete> Boolean field, with values of true or false, <layer> the number of layer in the layout, <cell> the name of the cell in which the object is drawn, <trans> information about the linear transformations that can be applied to the structure, <shape> the type of shape and the coordinates.

4.2 Database units

The database units are specified in the <dbu> tag and has usual values of 0.001, 0.01, 0.1. . This value is very important, since all the coordinates are given as micrometers multiplied by 1/dbu. The dbu value can be modified before exporting the lyn files in KLayout, by saving the gds and changing the value in the field of 'Database unit' or by File -> Layout Properties -> Database Unit. However, GDS2M handles all the dbu values.

4.3 Shapes

There are several shapes than can be implemented in KLayout. The ones that are used for MEMS and IC designs are: box, polygon, path (additional, point and text). The shape is written in the tag <shape>.

4.3.1 Box

Box is basically a rectangle, defined by two opposite corners: the corner from lower left and the corner from upper right (Fig. 1).

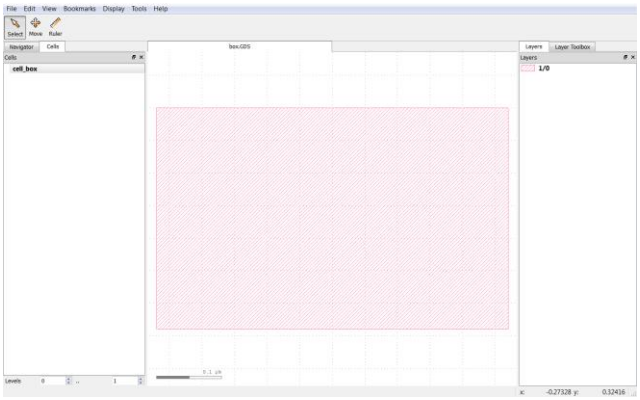


Figure 1. Box Shape in KLayout

```
<?xml version="1.0"?>
```

```
<nets>
 <net>
  <name>Net1</name>
  <top_cell>cell_box</top_cell>
  <layout>C:/KLayout/shapes/box.GDS</layout>
  <dbu>0.001</dbu>
  <complete>true</complete>
  <shapes>
   <element>
    <layer>1/0</layer>
    <cell>cell_box</cell>
    <trans>r0 *1 0,0</trans>
    <shape>box (-820,360;-280,700)</shape>
   </element>
  </shapes>
 </net>
</nets>
```

4.3.2 Simple Polygon / Polygon

Simple polygon is a polygon described by the *x* and *y* coordinates of the vertices, which can not contain open spaces (hulls) without having the inside vertices connected to the exterior (Fig. 2). *Polygon* represents an extension to the *Simple Polygon* shape, that can contain hulls and it's defined by the coordinates of the exterior contour and the coordinates of the interior contours. GDS2 format doesn't support the *Polygon* shape and automatically converts it in a *Simple Polygon* by introducing links between the interior and exterior points (Fig. 3).

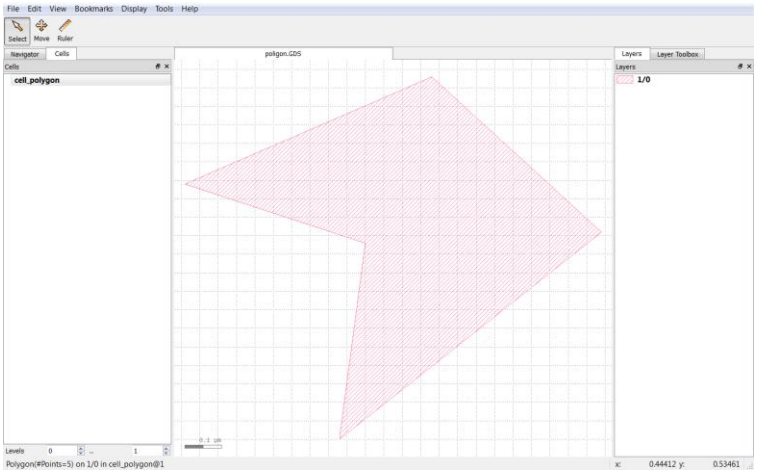


Figure 2. Simple polygon in KLayout.

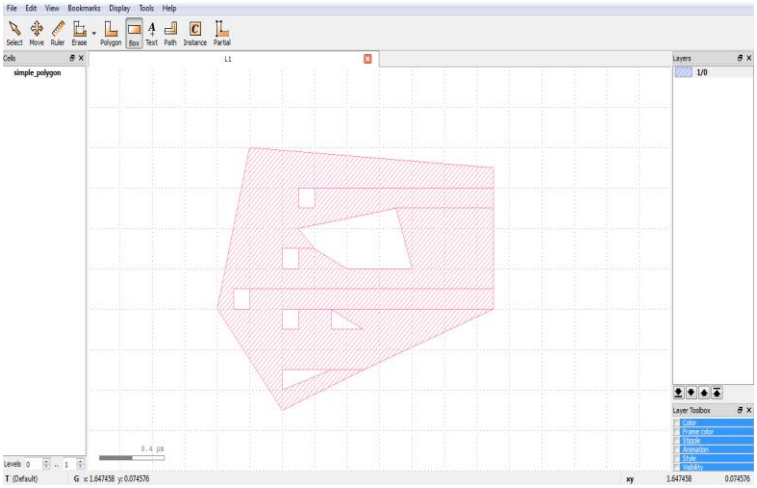


Figure 3. Simple polygon with hulls in KLayout.

```
<?xml version="1.0"?>
<nets>
 <net>
  <name>Net1</name>
  <top_cell>cell_polygon</top_cell>
  <layout> C:/KLayout/shapes/polygon.GDS</layout>
  <dbu>0.001</dbu>
  <complete>true</complete>
  <shapes>
   <element>
    <layer>1/0</layer>
    <cell>cell_polygon</cell>
    <trans>r0 *1 0,0</trans>
    <shape>polygon (-270,-450;-200,80;-690,240;-
20,530;440,110)</shape>
   </element>
  </shapes>
 </net>
</nets>
```

4.3.3 Path

Path is a polygonal shape, a line with a predefined width. A *path* is described by the vertices of its spine (a sequence of points that the line follows) and the width, measured transversally (Fig. 4). A *path*'s end caps can be rectangular (by default), or round (there are problems with the compatibility of round end caps with GDSII).

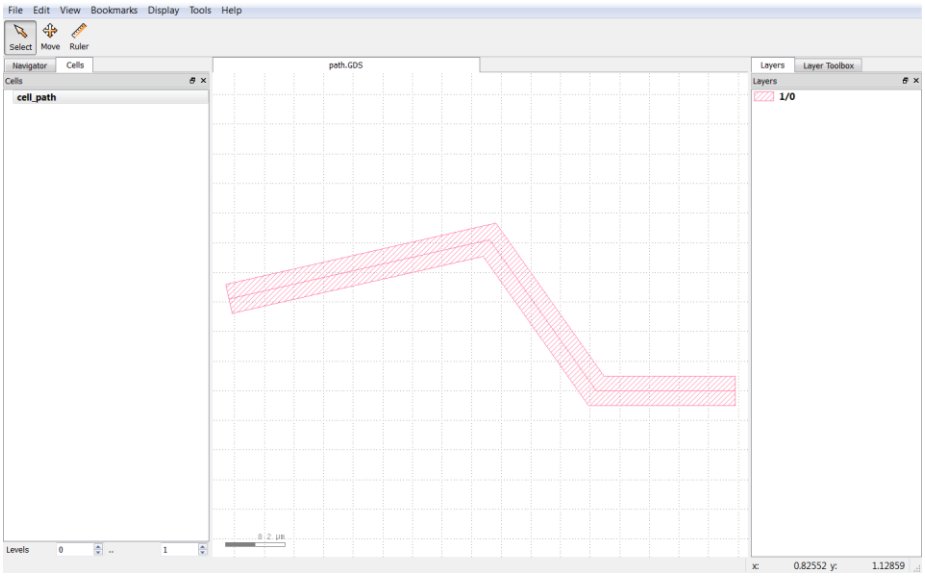


Figure 4. Path in KLayout.

```
<?xml version="1.0"?>
<nets>
 <net>
  <name>Net1</name>
  <top_cell>cell_path</top_cell>
  <layout> C:/KLayout/shapes/path.GDS</layout>
  <dbu>0.001</dbu>
  <complete>true</complete>
  <shapes>
   <element>
    <layer>1/0</layer>
    <cell>cell_path</cell>
    <trans>r0 *1 0,0</trans>
    <shape>path (-720,410;160,610;520,100;990,100) w=100 bx=0 ex=0
r=false</shape>
   </element>
  </shapes>
 </net>
</nets>
```

4.4 Transformations

Linear transformations such as translation, scaling rotation and mirroring can be applied to objects. Such transformations can be viewed in the tag <trans>. The format of the transformation is ([<dx>,<dy>] [r<angle> | m<angle>] [*<mag>]).

4.4.1 Scaling

*<s>: Scaling is a magnification by the factor "s". If no scaling is specified, none is assumed.

`<trans>r0 *23 0,0</trans>` - scaling by '23'

4.4.2 Rotation and mirroring

r<a> or **m**<a>: A rotation by angle "a" (in degrees) or mirroring at the "a" axis (the x axis rotated by "a" degree). The two transformations exclusive, only one can be applied at a time. If no rotation or mirroring is specified, none is assumed.

`<trans>r90 *1 0,0</trans>` - rotation by 90 degrees

`<trans>m45 *1 0,0</trans>` - mirror at 45 degrees axis (swap x and y)

4.4.3 Translation

<dx>,<dy>: The translation is applied after rotation and scaling in micron units, on x axis and on y axis. If no displacement is specified, none is assumed.

`<trans>r0 *1 100,50</trans>` - displacement by 100 µ on the x axis and by 50 µ on the y axis

5. Input files

GDS2M is able to directly implement a switch, with no coding required. However, the information has to be provided in some files : the lyn files (the files describing the 2D layout of the layers), the technology file (the file describing the 3D layout and other specific information about the layers) and the problem file (the file related to the introduction of information linked to the solving of the problem).

5.1 2D information files

The 2D information files are the lyn files (Lyn Files). The layers in the .GDS file represent the masks used. By using the facility Trace Net in KLayout, the user can export the 2D information of each object from the layers in a .lyn file which uses the XML format. These files are used to extract the geometric information and to draw the two-dimensional view of the switch.

5.2 3D information file - the technology file

The other information necessary for processing, such as the 3D information (height and thickness) and the material information for each layer, are taken from the documentation provided by the designers. Since there is no standard in this respect, we defined another .xml file, called the technology file, that holds this information. This file completely describes the geometry and constitution of the switch. It includes details about the substrates and exterior layers that encapsulate the switch and the masks that compose it. The exterior layers are defined by the thickness and material. The masks contain information about the path where the associate lyn file is stored, the type of the object the mask represent (ex: electrode, coplanar waveguide, membrane etc), the height, thickness and material. There is also a tag named <holes> which is '1' if the layer contains holes and '0' otherwise. If the value is '1', then the <holes_dimensions> tag becomes valid and holds the maximum area of a hole, specified to avoid elimination of important geometry. The color tag is specified for the case in which the designers prefer a certain code color.

Example of technology file for a capacitive shunt switch, described in [7].

```
<?xml version="1.0"?>
```

```
<nets>

	<layersExt>
		<layer>
			<material>SI</material>
			<thickness>600</thickness>
		</layer>
		<layer>
			<material>SIO2</material>
			<thickness>1</thickness>
		</layer>
		<layer>
			<material>AIR</material>
			<thickness>600</thickness>
		</layer>
	</layersExt>
	<masks>
		<mask>
			<filepath>problemsQian\strat1\</filepath>
			<object>CPW</object>
			<material>ALUM</material>
			<color>r</color>
			<height>601</height>
			<thickness>4</thickness>
			<holes>0</holes>
		</mask>
		<mask>
			<filepath>problemsQian\strat2\</filepath>
			<object>electrode</object>
			<material>ALUM</material>
			<color>c</color>
			<height>601</height>
			<thickness>0.4</thickness>
			<holes>0</holes>
		</mask>
		<mask>
			<filepath>problemsQian\strat3\</filepath>
			<object>dielectric</object>
			<material>NITRIDE</material>
			<color>g</color>
			<height>601.4</height>
			<thickness>0.1</thickness>
			<holes>0</holes>
		</mask>
		<mask>
			<filepath>problemsQian\strat4\</filepath>
			<object>membrane</object>
			<material>ALUM</material>
			<color>y</color>
			<height>605</height>
			<thickness>0.4</thickness>
			<holes>0</holes>
		</mask>
	</masks>
</nets>
```

5.3 The problem file

Chamy requires another input file, which describes the solved problem, containing details about the electric or magnetic terminals, geometric parameters and other extra dimensions for the layout included in the computational domain used in the EM analysis.

The terminals, on which the boundary conditions are set, are described by their labels, the bricks they are attached to, the domains they are included in and their types. A domain is defined by their limits: "xmin", "xmax", "ymin", "ymax", "zmin", "zmax". The terminal types may be: "eg" – electrical ground terminal, "ev" – electrical terminal excited in electric voltage, "ec" – electrical terminal excited in current, "mg" – magnetic ground terminal, "mv" – magnetic terminal excited in magnetic voltage, "mg" – magnetic terminal excited in magnetic flow.

The parameters represent the dimensions that can be varied in the parametric analysis, e.g. for variation studies or design optimization. They are attached to one object (example: membrane, coplanar waveguide, dielectric). The parameters must contain a tag that defines the type of dimension that is varied: W (width), L (length), H (height). Finally, the interval in which the value can be varied is specified in <lower_value> and <upper_value>. There are also other values given in the problem file, e.g. the values regarding the positioning of the switch in the computational box for *chamy*.

An example of the problem file used for the simulation of a capacitive shunt switch [7].

```
<problem>
	<terminals>
		<terminal>
			<brick_label>brick_1</brick_label>
			<terminal_label>ground_terminal_1</terminal_label>
			<plane>zmin</plane>
			<type>eg</type>
		</terminal>
		<terminal>
			<brick_label>brick_2</brick_label>
			<terminal_label>ground_terminal_2</terminal_label>
			<plane>zmin</plane>
			<type>eg</type>
		</terminal>
		<terminal>
			<brick_label>brick_3</brick_label>
			<terminal_label>cpw_terminal_1</terminal_label>
			<plane>zmin</plane>
			<type>ev</type>
		</terminal>
		<terminal>
			<brick_label>brick_1</brick_label>
			<terminal_label>ground_terminal_3</terminal_label>
			<plane>zmax</plane>
			<type>eg</type>
		</terminal>
		<terminal>
			<brick_label>brick_2</brick_label>
			<terminal_label>ground_terminal_4</terminal_label>
			<plane>zmax</plane>
			<type>eg</type>
		</terminal>
		<terminal>
			<brick_label>brick_4</brick_label>
```

```
                <terminal_label>cpw_terminal_2</terminal_label>
                <plane>zmax</plane>
                <type>ev</type>
            </terminal>
        </terminals>
        <params>
            <param>
                <object>membrane</object>
                <dimension>W</dimension>
                <lower_value>50</lower_value>
                <upper_value>200</upper_value>
            </param>
        </params>
        <extra_space>100</extra_space>
</problem>
```

6. Functions, algorithms

6.1 The two modules

The first module is destined for geometric modelling of the switch, based on the information stored in the lyn files and in the technology file. The main script for completing this action and viewing the results is:

```
[objStruct,rectangleStruct,xmax,zmax,LayersExtStruct] =
main_script_extract_info_from_gds(),
```

where:
- objStruct = matrix of data structures, each line representing a mask in the constitution of the switch. Each data structure describes one object, giving information about : *x* and *y* coordinates of each point in the object, shape, netname, cellname (from the lyn files), type, *z* coordinate, thickness and material (from the technology file).
- rectangleStruct = array of data structures, derived from objStruct. Each element represents a layer and contains information about the rectangles that, through their union, can approximate the shape of the layer. The material, type, *z* coordinate and thickness are identical for all the rectangles in the layer and are the same as in objStruct. There are four more arrays: *x* and *y* are the coordinates of the bottom left vertex of the rectangle, *w* and *h* are the width and height of the rectangle.
- LayersExtStruct = array of data structures, describing the exterior layers that encapsulate the switch. Those layers do not exist in the .GDS format and are depicted in the technology files. Each layer is characterized by material, thickness, height. The height is a derived quantity, obtained by processing the height information and order of the layers.
- xmax = the maximum value of the *x* axis of the domain.
- zmax = the maximum value of the *z* axis of the domain.

The information summarised and offered by `main_script_extract_info_from_gds` is used in the add_layout_to_device_* function.

```
device = add_layout_to_device_*(device)
```

Along with the information extracted from the problem file, xmax, zmax, rectangleStruct and LayersExtStruct assemble the layout of the switch and make it available and fit for the RF simulation. To view the representation of the device and the results of the simulations, *chamy*-specific scripts such as `script_dev2sys_*` or `script_dev2snp_*` have to be called.

6.2 Extracting information from KLayout

Functions :

- `[noObj,listObj,domain] = read_lyn(lyn_files_path,sep);`

- called from main_script_extract_info_from_gds
- reads in all the *.lyn files found in the folder given by lyn_files_path

- `[noObj,listObj,domain] = read_netlist(filename_net,noObj,listObj);`

- called from read_lyn
- reads in all the objects found in a specified *.lyn file and attach them to the existing list of objects

- `net_struct = xml_load(filename_net)`

- called from read_netlist, read_xml and read_problemFile
- function available in MATLAB XML toolbox [8] for parsing xml files

- `elem_info = find_geometric_info(current_shape,current_transformation,dbu)`

- called from read_netlist
- attributes the geometric information according to the shapes to the objects and applies the linear transformations (Transformations)

6.3 Extracting information from the technology and problem files

- `[NoLayers, problempath, type, material, colorlayer, zlayer, thickness, holes, LayersExt] = read_xml('technologyFile.xml');`

- called from main_script_extract_info_from_gds
- parses the technology file

- `[terminals, params,l_extra_space] = read_problemFile('problemFile.xml');`

- called from add_layout_to_device_*
- parses the problem file

6.4 Processing the information and drawing

- `draw_background(domain,color)`

- called from main_script_extract_ info_from_gds and from every function that draws a part of the switch
- draws a rectangle corresponding to the domain and fill it with color

- `[r] = rectangle_reunion (noObj,listObj,color)`

- called from main_script_extract_info_from_gds
- breaks the object (listObj) into rectangles on the basis of the horizontal and vertical lines and then unites the neighbour rectangles by updating the dimensions of the first rectangle and eliminating the added rectangle

- `draw_union(r,color)`

- called from rectangle_union
- draws the approximation of the layer's surface by the rectangles found in rectangle_union

- `[listObj] = prepare_rectangular (listObj)`

- called from main_script_extract_ info_from_gds

- checks for oblique lines in the object and if replaces them with an horizontal and vertical line (stairways segments)

- **dimensions**(noObj,listObj,color,domain)

- called from main_script_extract_ info_from_gds
- this program draws the objects in 2D and computes their dimensions. It is similar to draw_shapes

- **draw_shapes**(noObj,listObj,color)

- called from main_script_extract_ info_from_gds
- draws shapes and and fills them with color

- [listObj] = **no_gaps** (listObj,dim,color,domain,k)

- called from main_script_extract_ info_from_gds
- eliminates the gaps from the specified object (listObj)
- dim - the maximum area of one gap; color - the color for the specified object; domain - the domain of the object; k - the number of the present figure

- [pos] = **holes_coord**(listObj,dim)

- called from no_gaps
- returns an array with the positions of first and last vertex of a holes sequence

- [listObj] = **membrane** (listObj,pos,color)

- called from no_gaps
- draws the membrane without the holes; pos is the array that stores the positions of the holes vertices

- **draw_shapes_3D** (noObj,listObj,color,z0,height)

- called from main_script_extract_ info_from_gds
- this function uses fill3 to create the above and below 2D faces of each shape and patch to create faces in between. The resulted objects are empty inside

6.5 Adding the information to *chamy*

- device = **add_layout_to_device_*** (device)

- called from read_device_*
- uses the information from `main_script_extract_info_from_gds` and the information from the problem file to add the layout to device structure
- LayersExtStruct is used to define the exterior layout, *xmax*, *zmax* and the sum of the layers' thicknesses as *ymax* define the borders of the computational box, l_extra_space from the problem file is used to adjust the positioning of the switch in the computational box mentioned above
- creates the bricks with the information in rectangleStruct
- attaches the terminals and the parameters from the problem file to the specified bricks

- [j,p] = **find_object_underneath**(nr,r,l,m)

- called from add_layout_to_device_*
- nr = the number of the layer, r = rectangleStruct, l = the number of the object in the layer, m = the number of the rectangle in the object (m=1 if the object has a rectangular shape; if the object has a complex shape, m can have other values), j = the number of the object from the below layer (nr-1) that is under the l-object, p = the number of the rectangle from the j-object
- finds which rectangles can be found underneath the object for which it's called
- used to create vertical bricks, as in the case of the dielectric which drips on the underneath object

- [p] = **find_disposition_up**(r,m)

- called from add_layout_to_device_*
- r = an object from a layer, m = the number of the rectangle from r for which we want to find if it makes contact with another rectangle on the up-side, p = the number of the rectangle that touches rectangle m on the specified side
- finds how the overlapping of the superior side of the object is

- used to create vertical bricks, as in the case of the dielectric which drips on the underneath object

- `[p] = find_disposition_down(r,m)`

- called from add_layout_to_device_*
- r = an object from a layer, m = the number of the rectangle from r for which we want to find if it makes contact with another rectangle on the down-side, p = the number of the rectangle that touches rectangle m on the specified
- finds how the overlapping of the inferior side of the object is
- used to create vertical bricks, as in the case of the dielectric which drips on the underneath object

7. Chamy

Chamy is a software tool developed by the LMN team [9] for the modeling of high-frequency integrated circuits components and their interaction with the electromagnetic environment. It computes the frequency characteristics of analyzed devices. It can be regarded as a MATLAB toolbox for RF simulations of HF integrated microsystems. In a straight forward manner, the entire process consists of the following steps: *import* the passive device description, *extract* the model and *generate* the state-space representation of the device, *compute* the frequency characteristics of the device and *export* the results in a standard *.snp* format [10].

The input parameters are mainly related to the layout description and technology description, in other words the device geometry and material properties. After problem description, the next step in *chamy* is to generate the state-space model of the device, by using the Finite Integration Technique (FIT) to discretize Maxwell's equations with EMCE boundary conditions:

$$\boldsymbol{C}\frac{d\boldsymbol{x}(t)}{dt} + \boldsymbol{G}\boldsymbol{u}(t) = \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{y}(t) = \boldsymbol{L}\boldsymbol{x}(t), \tag{1}$$

where **x** is the state vector, **u** is the input vector and **y** is the vector of output signals. In the frequency domain, the relationship between the input and output signals is obtained by solving an algebraic system of linear complex equations for each frequency, where the input/output behavior in the frequency domain is given by the transfer matrix:

$$\boldsymbol{H}_{FIT}(\omega) = \boldsymbol{L}\,(\boldsymbol{G} + j\omega\boldsymbol{C})^{-1}\,\boldsymbol{B}, \quad \boldsymbol{y} = \boldsymbol{H}_{FIT}\boldsymbol{u}\,. \tag{2}$$

This step is done by using Adaptive Frequency Sampling (AFS) combined with Vector Fitting (VF) [11] [12]. The VF procedure uses as input a set of pairs (ω_k, $\mathbf{H}(\omega_k)$), k=$\overline{1,F}$, where *F* is the number of frequency samples. Its goal is to identify the poles p_i, the residual matrices $\boldsymbol{K}_i$ and the constant terms *K*, K_0 of the rational approximation for $\boldsymbol{H}_{FIT}(\omega)$:

$$\boldsymbol{H}_{VF}(\omega) = \sum_{i=1}^{q}\frac{\boldsymbol{K}_i}{j\omega - p_i} + \boldsymbol{K}_{\infty} + j\omega\boldsymbol{K}_0. \tag{3}$$

This step represents the model order reduction phase, which is essential for the extraction of a circuit model that can be further used in (re)designing more efficient micro-electro-mechanical devices [13].

Next, we will present the functions in *chamy* that are relevant to GDS2M. This functions interact with add_layout_to_device_*, which is the function in which the layout is described with the information extracted by GDS2M.

- **`script_dev2sys_*`** and **`script_dev2snp_*`**

`Script_dev2sys_*` and `script_dev2snp_*` are two of the main functions from *chamy*. They link the folder with the source files used for the analysis and the folder where the

problem is defined.

`Script_dev2sys` calls `read_device_*` function, creating the model of the structure, and generating the FIT matrices required by the EM computation. It also creates figures of the structure for visualization. `Script_dev2snp_*` is used for the EM computation using the FIT matrices generated by `script_dev2sys_*`, for a number of frequency samples calculated using AFS.

- `device = read_device_*()`

- called from script_dev2sys_*
- is designed to gather information about the structure (geometry, position, material properties, boundary conditions) and mesh domain. The structure intended for analysis is placed in the computational domain resulted from read_device_*.

- `device = initiate_device_*()`

- called from read_device_*
- assigns to information such as structure type, software version, id and the name of the folder for output or intermediate data to the structure.

- `device = add_layout_to_device_*(device)`

- called from read_device_*
- describes the geometric information of the computational domain, structure and boundary conditions.

- `device = add_matlib_to_device_*(device)`

- called from read_device_*
- gives detailed information about the materials of switch's components.

- `device = add_infogrid_to_device_*(device)`

- describes the discretization grid for the computational domain

8. Running the code

This section shows how to run the code for an example of an RF-MEMS shunt capacitive switch and what the returned results are. The description of the benchmark can be found in [7].

For a given .gds file (Fig. 5), describing the switch, we extract from KLayout the .lyn files, using the Trace Net function. For the specified switch, there are 7 objects, therefore 7 .lyn files, organised in folder by masks (4 masks : CPW, electrode, dielectric and membrane). These files can be found in Appendix A.

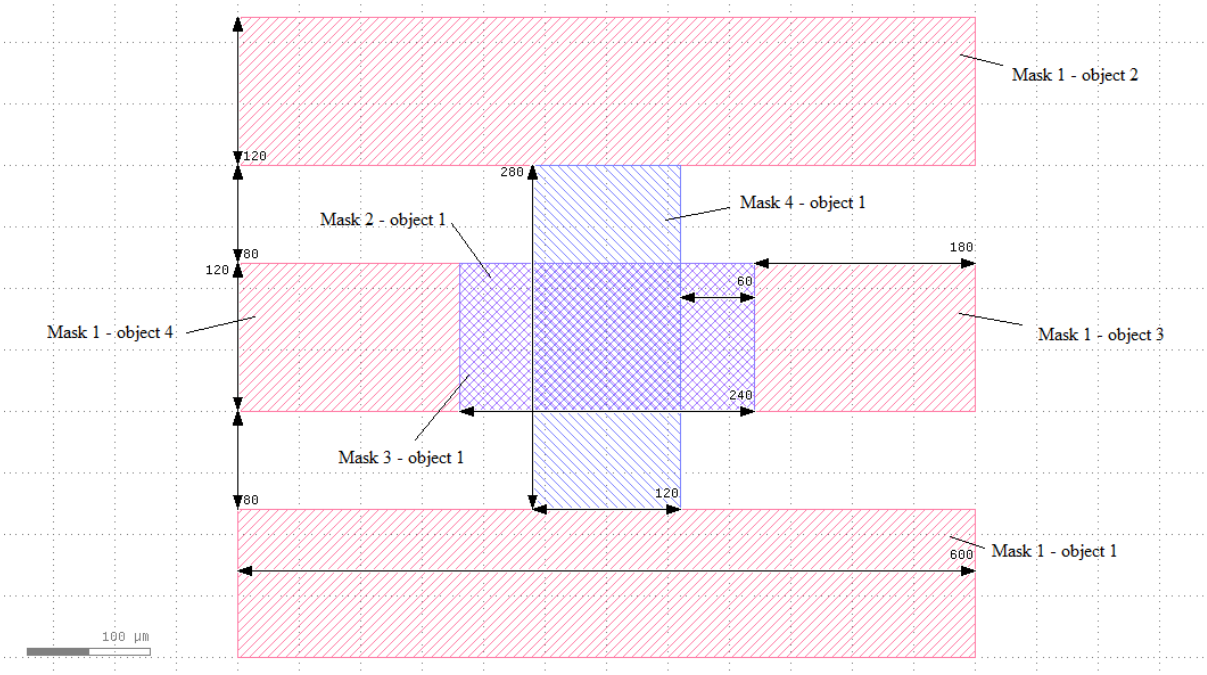
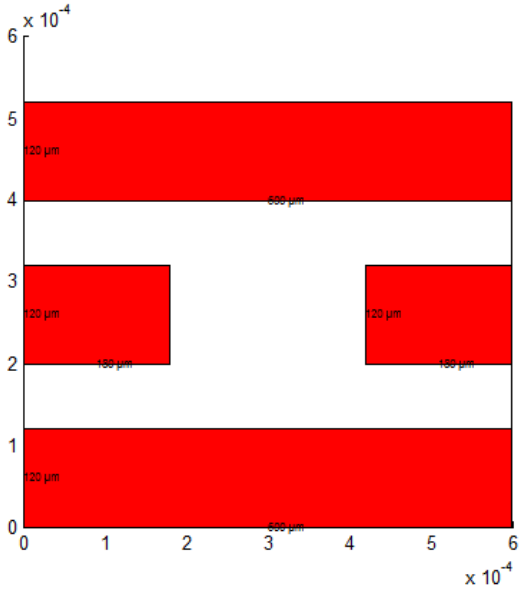


Figure 5. GDS file for an RF shunt capacitive switch. Dimensions and numbering are shown.

The other two input files for this switch can be found in the Technology file section and in the Problem file section in this document. While the file paths in the .lyn files are not important and the files can be used in the actual form (the <layout> tag represents the file path where the GDS file can be found and it is not used in the program), the <filepath> tag in the technology file must match the address where the .lyn files can be found.

By calling `main_script_extract_info_from_gds`, the following results and figures are obtained:

```
Obj =

    [1x1 struct]    [1x1 struct]    [1x1 struct]    [1x1 struct]
    [1x1 struct]              []              []              []
    [1x1 struct]              []              []              []
    [1x1 struct]              []              []              []

r =

    {1x4 cell}    {1x1 cell}    {1x1 cell}    {1x1 cell}

xmax =

   6.0000e-04

zmax =

   5.2000e-04


LayersExt =

    [1x1 struct]    [1x1 struct]    [1x1 struct].
```



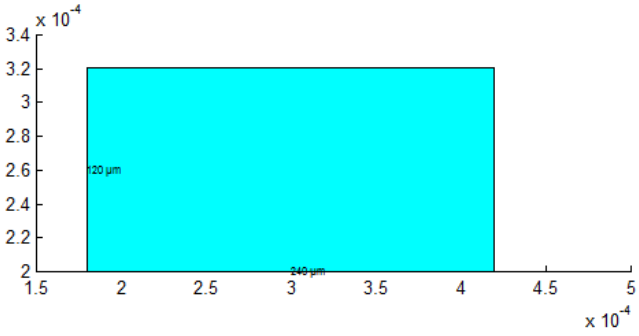


Figure 6. Mask 2 (electrode) from the RF shunt capacitive switch with dimensions.

Figure 7. Mask 1 (CPW lines) from the RF shunt capacitive switch with dimensions.

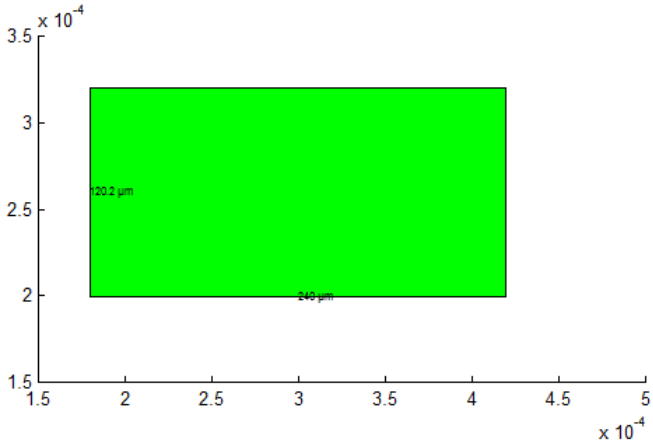


Figure 8. Mask 3 (dielectric) from the RF shunt capacitive switch with dimensions.

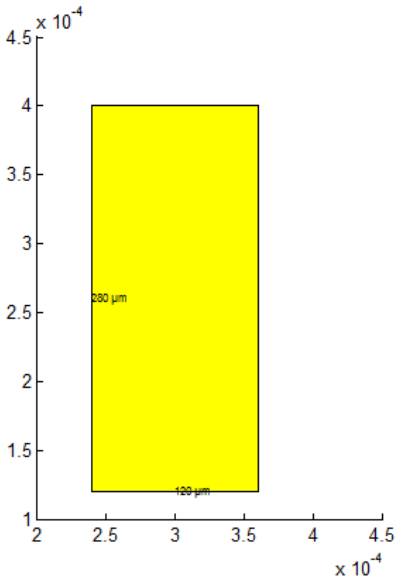


Figure 9. Mask 4 (membrane) from the RF shunt capacitive switch with dimensions.

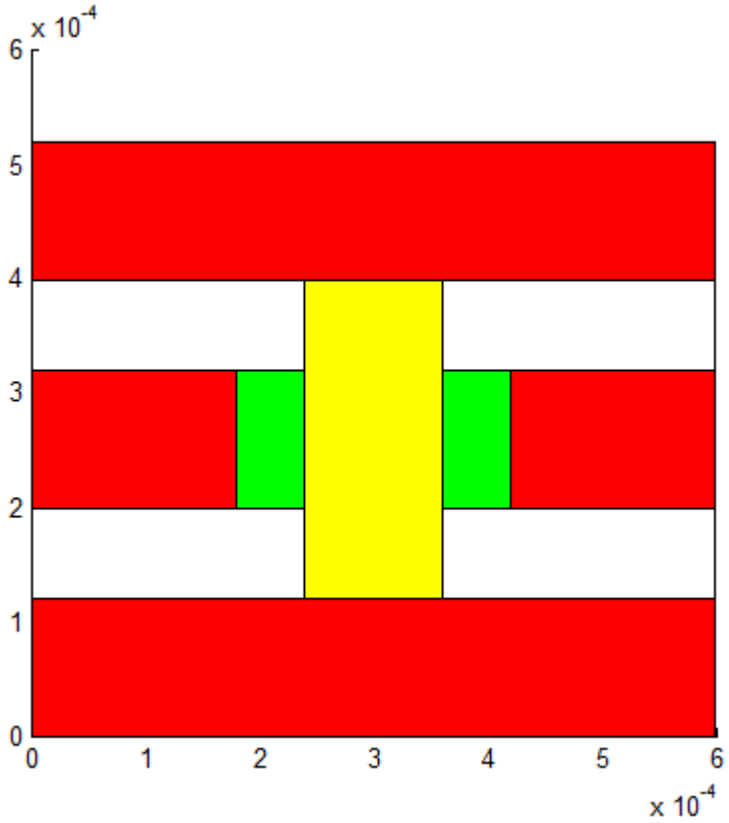


Figure 10. The 2D view of the RF shunt capacitive switch, created by overlaying the masks.

In Fig. 6 to Fig. 10, the two-dimensional layout, obtained only by using the .lyn files, is shown.

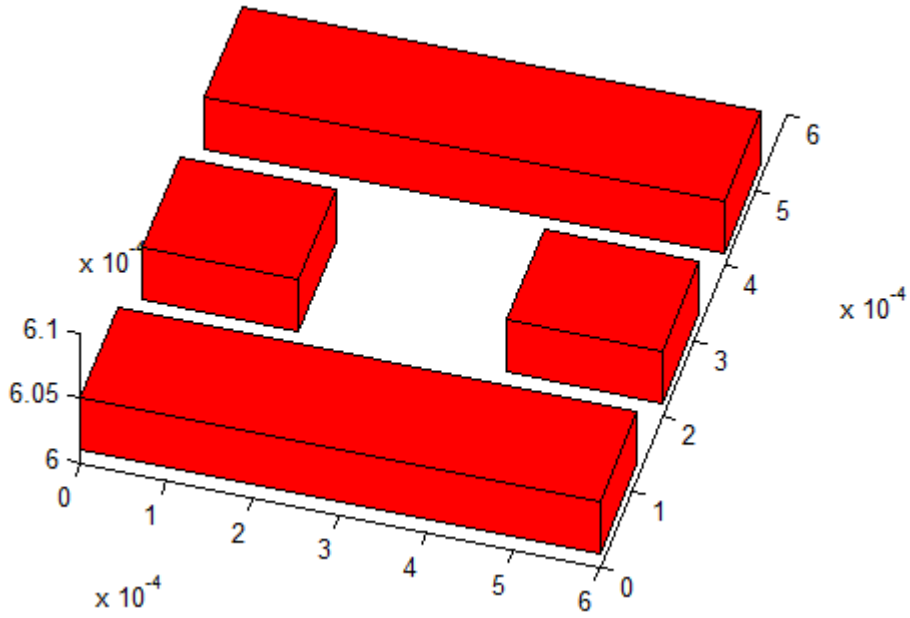


Figure 11. 3D view of Mask 1 for the RF shunt capacitive switch.

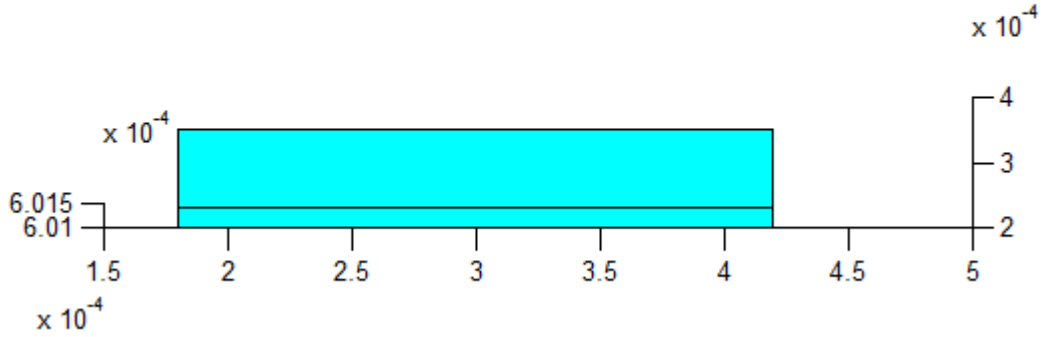


Figure 12. 3D view of Mask 2 for the RF shunt capacitive switch.

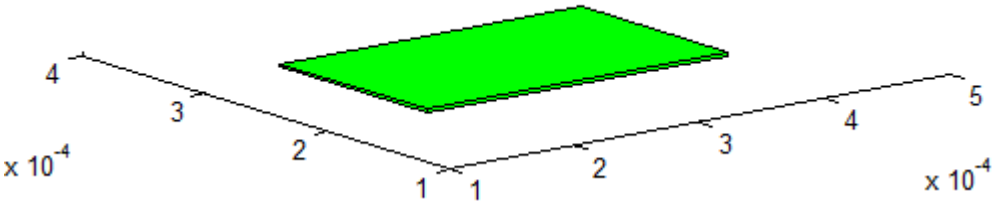


Figure 13. 3D view of Mask 3 for the RF shunt capacitive switch.

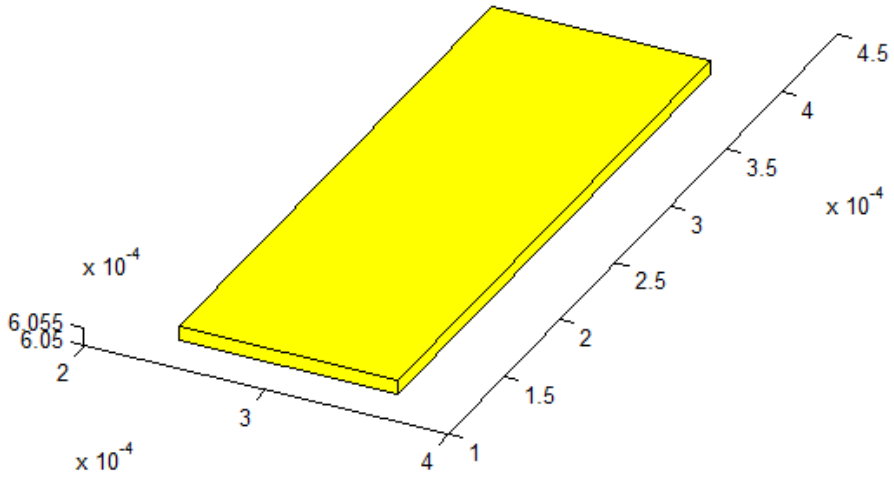


Figure 14. 3D view of Mask 4 for the RF shunt capacitive switch.

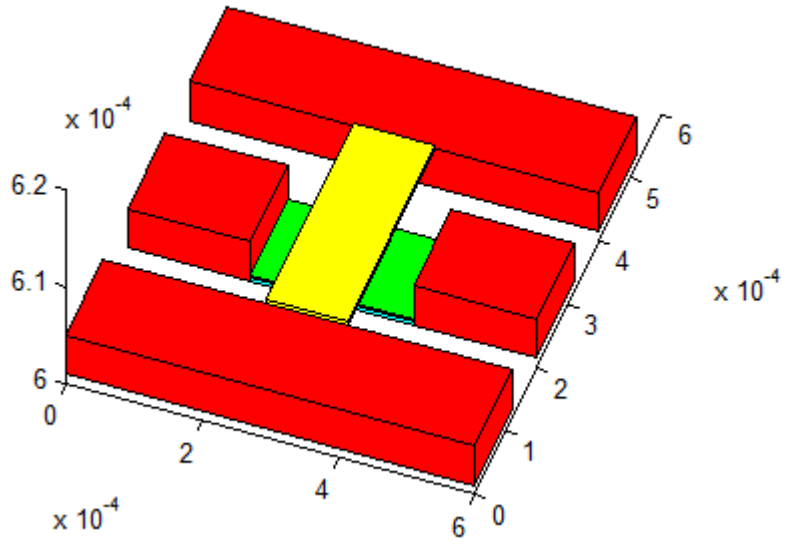


Figure 15. 3D view of the layout for the RF shunt capacitive switch, obtained by overlaying the masks.

In Fig. 11 to Fig. 15, the three-dimensional layout of the RF shunt capacitive switch, obtained by merging the information from the lyn files and the technology file, is shown.

By running `script_dev2sys_SwC_Qian`, which uses `add_layout_to_device_SwC_Qian_GDS2M`, the following figures are shown:

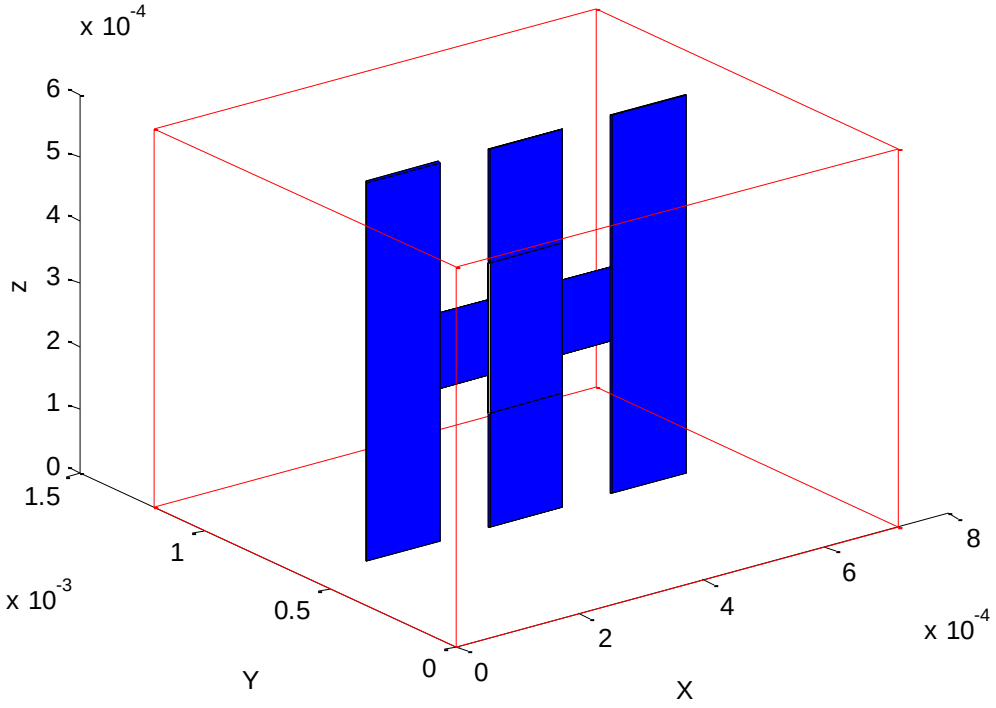


Figure 16. Model for the RF shunt capacitive switch in *chamy*.

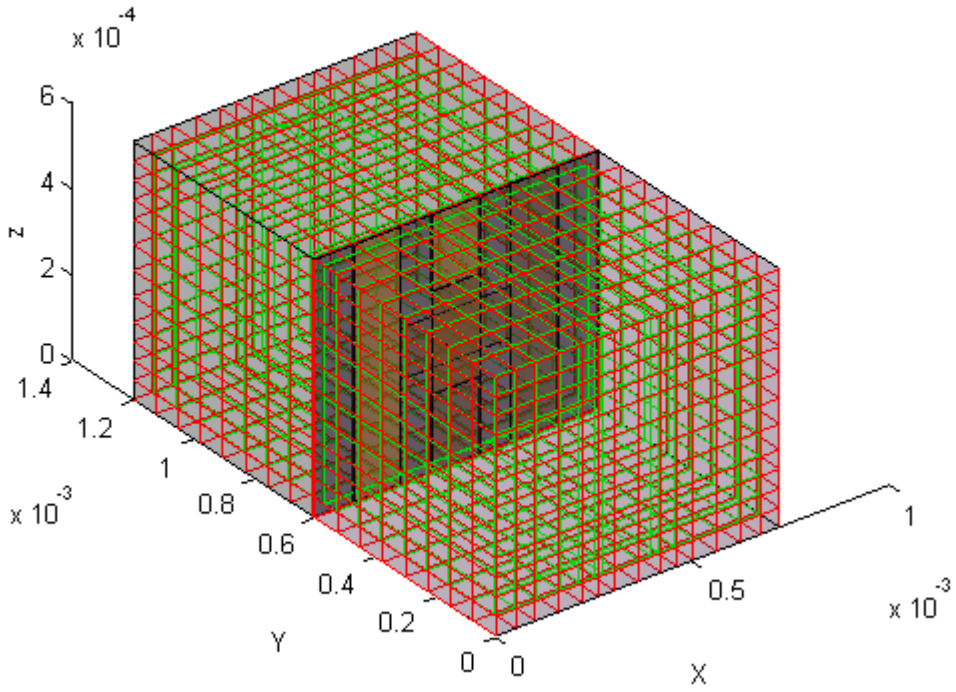


Figure 17. Discretization grid used in *chamy* for the RF shunt capacitive switch.

These two figures show the correct transfer of information from GDS2M to *chamyI.* The script returns other results as well, regarding the full wave analysis.

9. Important observations and future improvements

This software tool has been recently released and still has some features that need improvement. Most of these are related to the covering all the possibilities when transferring the data to *chamy*. For example, at the moment, only one parameter of a type (W, L, H) can be introduced. Connecting the parameters to the specified bricks must be improved, because the parameter types have special features and a standard variable can only inherit the numerical value of the parameter. Two more functions for building vertical bricks for the left and right side in the case of pouring dielectric must be added.

Changing the state of the membrane from UP to DOWN is done by modifying the height tag for the respective mask in the technology file. In the future, a function for doing this automatically will be designed.

10. Conclusions

GDS2M is a software tool aimed to facilitate the processing of geometrical information of MEMS. It starts from the .GDS file, which describes the 2D geometry of the device's layers, and from

the technology file, which describes the vertical structure of the device. These two input files are translated in XML format and then, along with the problem file (which contains information about the solved problem), by merging their content, the input file for the EM analysis tool, *chamy*, is generated.

In the process of setting up a new simulation in *chamy*, the step of introducing the geometric layout is the one that takes the longest user's time to complete. This happens mainly because this is usually done by hard coding into the layout file, defining the parameters, the domain, layers, bricks and terminals. This is not only time consuming, but is also the main source of runtime errors. By automating this step, GDS2M removes the possibility of human errors, especially in the case of complex device layouts, and makes a solid correspondence with the .GDS layout source files that are usually provided by our industrial partners.

11. Acknowledgments

This work was done under the guidance of Prof. Daniel Ioan and Assoc. Prof. Gabriela Ciuprina, UPB. The program is part of the ToMeMS project, conducted under the financial support of the Romanian Government program PN-II-PT-PCCA-2011-3, managed by ANCS, CNDI – UEFISCDI, grant no. 5/2012.

12. References

[1] MATLAB, an interactive numerical computing environment and high-level language, developed by MathWorks http://www.mathworks.com/matlabcentral/

[2] CHAMELEON-RF project: http://www.hitech-projects.com/euprojects/chameleon%20RF/

[3] M. Kofferlein, KLayout High Performance Layout Viewer And Editor, Version 0.21.16, Free software and documentation available at http://klayout.de

[4] V. K. Varadan, K.J. Vinoy, K.A. Jose, *RF MEMS and Their Application Electricity and Magnetism,* John Wiley & Sons Ltd, West Sussex, England 2003, pp.13-16.

[5] G. M. Rebeiz, G.B. Muldavin, *RF MEMS Switches and Switch Circuits,* IEEE Microwave Magazine, December 2001, pp. 59-71

[6] Jürgen Thies, Layout Editor, IC and MEMS designs Viewers and Editor; extensive list of similar software tools available at http://www.layouteditor.net/links/

[7] J.Y. Qian, G.P. Li and F. De Flaviis, *A parametric model of low-loss RF MEMS capacitive switches*, Asia-Pacific Microwave Conference, APMC 2001, Taipei, Taiwan, 2001.

[8] M. Molinari, XML toolbox, Conversion of MATLAB data types into XML and vice versa, April 2005 http://www.mathworks.com/matlabcentral/fileexchange/4278-xml-toolbox

[9] Numerical Methods Laboratory (LMN), research center in the domain of computerized electrical engineering, http://www.lmn.pub.ro/

[10] D. Ioan, G. Ciuprina, M. Radulescu, *Compact modeling and fast simulation of the on-chip interconnect lines*, IEEE Transactions on Magnetics, vol.42, issue 4, pp.547-550, 2006.

[11] B. Gustavsen, A.Semlyen, *Rational approximations of frequency domain response by vector fitting*, IEEE Transactions on Power Delivery, vol.14, July 1999.

[12] I.A. Lazar, G. Ciuprina, D. Ioan, *Effective extraction of accurate reduced models for HF-IC using multi-CPU arhitectures*, Inverse Problems in Science and Engineering, vol.20,no.1, 2010.

[13] G. Ciuprina, D. Ioan, I.A. Lazar, C.B. Dita, *Vector Fitting Based Adaptive Frequency Sampling for Compact Model Extraction on HPC Systems*, IEEE Transactions on Magnetics, vol.48, issue 2, pp.431-434, 2012.

APPENDIX A

Mask 1 - object 1

```
<?xml version="1.0" encoding="utf-8"?>
<nets>
 <net>
  <name>Net2</name>
  <top_cell>MainCell</top_cell>
  <layout>C:\ToMeMS\GDS\qian.GDS</layout>
  <dbu>0.01</dbu>
  <complete>true</complete>
  <shapes>
   <element>
    <layer>1/0</layer>
    <cell>MainCell</cell>
    <trans>r0 *1 0,0</trans>
    <shape>box (0,0;60000,12000)</shape>
   </element>
  </shapes>
 </net>
</nets>
```

Mask 1 - object 2

```
<?xml version="1.0" encoding="utf-8"?>
<nets>
 <net>
  <name>Net3</name>
  <top_cell>MainCell</top_cell>
  <layout>C:\ToMeMS\GDS\qian.GDS</layout>
  <dbu>0.01</dbu>
  <complete>true</complete>
  <shapes>
   <element>
    <layer>1/0</layer>
    <cell>MainCell</cell>
    <trans>r0 *1 0,0</trans>
    <shape>box (0,40000;60000,52000)</shape>
   </element>
  </shapes>
 </net>
</nets>
```

Mask 1 - object 3

```
<?xml version="1.0" encoding="utf-8"?>
<nets>
 <net>
  <name>Net2</name>
  <top_cell>MainCell</top_cell>
  <layout>C:\ToMeMS\GDS\qian.GDS</layout>
  <dbu>0.01</dbu>
  <complete>true</complete>
  <shapes>
```

```
   <element>
    <layer>1/0</layer>
    <cell>MainCell</cell>
    <trans>r0 *1 0,0</trans>
    <shape>box (42000,20000;60000,32000)</shape>
   </element>
  </shapes>
 </net>
</nets>
```

Mask 1 - object 4

```
<?xml version="1.0" encoding="utf-8"?>
<nets>
 <net>
  <name>Net1</name>
  <top_cell>MainCell</top_cell>
  <layout>C:\ToMeMS\GDS\qian.GDS</layout>
  <dbu>0.01</dbu>
  <complete>true</complete>
  <shapes>
   <element>
    <layer>1/0</layer>
    <cell>MainCell</cell>
    <trans>r0 *1 0,0</trans>
    <shape>box (0,20000;18000,32000)</shape>
   </element>
  </shapes>
 </net>
</nets>
```

Mask 2 - object 1

```
<?xml version="1.0" encoding="utf-8"?>
<nets>
 <net>
  <name>Net4</name>
  <top_cell>MainCell</top_cell>
  <layout>C:\ToMeMS\GDS\qian.GDS</layout>
  <dbu>0.01</dbu>
  <complete>true</complete>
  <shapes>
   <element>
    <layer>2/0</layer>
    <cell>MainCell</cell>
    <trans>r0 *1 0,0</trans>
    <shape>box (18000,20000;42000,32000)</shape>
   </element>
  </shapes>
 </net>
</nets>
```

Mask 3 - object 1

```
<?xml version="1.0" encoding="utf-8"?>
<nets>
 <net>
  <name>Net4</name>
  <top_cell>MainCell</top_cell>
  <layout>C:\ToMeMS\GDS\qian.GDS</layout>
```

```
  <dbu>0.01</dbu>
  <complete>true</complete>
  <shapes>
   <element>
    <layer>3/0</layer>
    <cell>MainCell</cell>
    <trans>r0 *1 0,0</trans>
    <shape>box (18000,19990;42000,32010)</shape>
   </element>
  </shapes>
 </net>
</nets>
```

Mask 4 - object 1

```
<?xml version="1.0" encoding="utf-8"?>
<nets>
 <net>
  <name>Net2</name>
  <top_cell>MainCell</top_cell>
  <layout>C:\ToMeMS\GDS\qian.GDS</layout>
  <dbu>0.01</dbu>
  <complete>true</complete>
  <shapes>
   <element>
    <layer>4/0</layer>
    <cell>MainCell</cell>
    <trans>r0 *1 0,0</trans>
    <shape>box (24000,12000;36000,40000)</shape>
   </element>
  </shapes>
 </net>
</nets>
```