Online CAD for Simulating and Laying Out Parameterized Arrays of Reference MEMS Designs

*Prabhakar Marepalli[1], Jason V. Clark[1,2,3]

[1]School of Mechanical Engineering, [2]School of Electrical and Computer Engineering,
[3]Birck Nanotechnology Center, Purdue University, West Lafayette, Indiana, USA

ABSTRACT

We present an online MEMS design tool called Sugar2GDSII for simulating and laying out parameterized arrays of a reference microdevice. Our tool is integrated in our MEMS simulation tool called SugarCube, which is readily available at nanoHUB.org. The reference device can be selected from a library of ready-made systems, or new devices can be imported from our Sugar tool. An important aspect of SugarCube is that the tool facilitates design and exploration by novice users. Various geometric, material, and drive parameters may be swept over design-rule ranges by using sliders. One- or two-dimensional layout arrays may be grounded to common bonding pads through automatically generated tracers and pads. Automatic features like etch holes, multi-layer bonding pads, etc., can be generated. The layout output is in GDS-II format. Design rules for PolyMUMPs or SOIMUMPs multi-layer are defined. Static, model, or transient analysis of the array of devices by SugarCube results in a value, 2D curve, or 3D plot manifold. Such automated features are expected to reduce the time that many designers might otherwise spend on manually laying out parameterized arrays of MEMS. Time can be reduced from days to minutes in many cases.

Keywords: Layout, GDSII, Sugar, SugarCube, Simulation, Parameterization, CAD, nanoHUB

1 INTRODUCTION

In MEMS design cycle, layout generation is one of the most critical and time consuming stages. Mistakes can be costly in terms of failed fabrication runs and lengthy turnaround times. Especially when the layout has multiple features like etch holes, tracing lines, multilayer bonding pads, etc., the chances of making a mistake are high. Similar problems that exist in electronics industry were addressed by developing layout tools (e.g. L-edit, Cadence, etc.) that can automatically generate features that might be required in an IC layout. Conventionally, MEMS designers used these tools from the electronics industry to generate their layouts. Although these tools provide many features, they are not specific to the needs of MEMS designers, making the layout process tedious and time consuming. With the progress in MEMS field, specialty CAD for MEMS tools also evolved very quickly. Currently tools like Coventorware [2], Intellisense [3], softMEMS [4], NODAS [5], etc., provide features that can automatically generate layout from MEMS model. But these tools have limitations in that they cannot readily automate the generation of parameterized arrays of devices, connecting tracers, ground planes, etc. And tools that may be extended to generate such features ([2], [6]) have advanced operating instructions, requiring users to develop complicated scripting code.

Our tool called Sugar2GDSII is developed to address the above needs for novices. It can automatically generate a layout for a MEMS device or an array of devices from a Sugar netlist [13]. The layout output file is in GDS-II format and adheres to a set of design rules required for a fabrication process. Features like etch holes, multilayer bonding pads, tracers, etc. can be automatically added to the layout upon the user's request. What is quite useful about our advance is that it may be accessed through a web browser at the nanoHUB (SugarCube [12]). To use this SugarCube advance, the user first loads a reference MEMS device from a library of systems, then modifies its parameters using design-rule limited sliders. Based on the desired parameters to sweep, the new design modifications may be either simulated for performance exploration or laid out for fabrication, all with a single button click.

The rest of the paper is organized as follows. In Section 2 we provide a brief description of a standard GDS-II file structure. In Section 3 we discuss how Sugar2GDSII converts a Sugar netlist to GDS-II format. We then present features that can be automatically generated by Sugar2GDSII. Section 4 discusses how this tool is integrated into SugarCube. We conclude our discussion in Section 5.

2 GDSII FILE DESCRIPTION

A GDSII file is a stream format description of 2D layout data [7], [8]. It contains hierarchy of structures, with each structure containing the information of layout elements in the form of records. The layout can have different layers with the layer number of each element specified as parameter. As GDSII file is in stream format (binary), it is platform independent. When opened in a GDSII viewer, the viewer converts this stream format into local platform datatype. A text representation of GDSII format is also available, called ASCII format. This format is easily readable and helps the designer to understand the layout and make any modification, if necessary. Conversion from

ASCII text format to stream format is done using standard binary to stream conversion process. Information in a GDSII file is stored as sequential set of records. These records store the details like layer number, x-y coordinates of each polygon described in the layout, datatype of coordinates, etc. Sugar2GDSII described in this paper converts the nodal coordinates of Sugar elements into x-y coordinates of GDSII polygon and stores them in the record format.

3 SUGAR2GDSII

Using the above information of a GDSII file, Sugar2GDSII generates the x-y coordinates of the vertices of every element in the netlist and inserts them in the form of a record with necessary information like layer number, data type of x-y coordinates, etc. See Figure 1. These xy-coordinates are generated from the node information that is extracted from Sugar netlist. Additional information that might be required for a GDSII layout file can be specified as parameters in the netlist. For example, if there is a feature in the layout that has to be released in the fabrication process, the user simply adds the parameter *release=1*, and Sugar2GDSII automatically identifies this value and generates the features required to release the structure (discussed in Section 3.1). Once the GDSII ASCII file is generated, it is converted to stream format using binary to stream conversions.

In this section, we describe some automatic features that can be generated by Sugar2GDSII. All the features discussed below adhere to standard design rules of a process like PolyMUMPs [10] or SOIMUMPs [11].

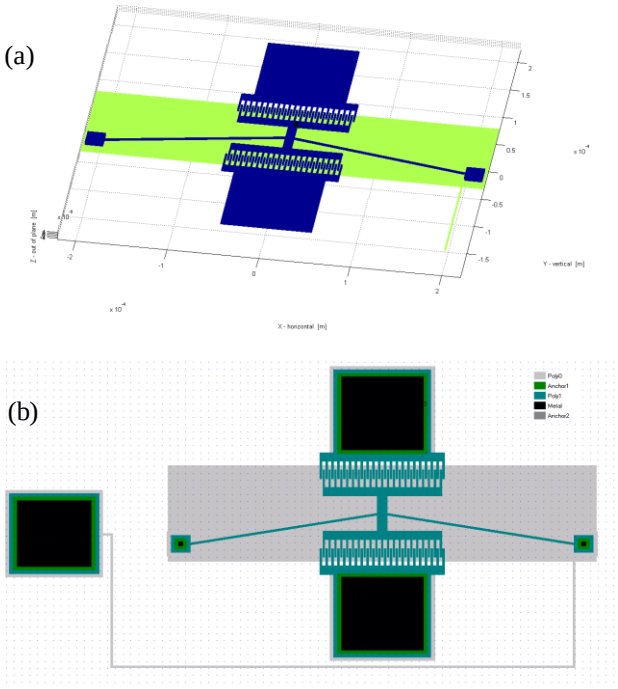

Figure 1: SugarCube and GDSII counterparts. An electrothermal senor is shown. The design as shown in SugarCube is given in (a). The layout generated by Sugar2GDSII is shown in (b) as seen in CleWin, a free GDSII viewer [14].

3.1 Automatic Etch Holes

Sugar2GDSII can automatically generate etch holes in a given layout. These etch holes might be required for proper release of the oxide layer underneath the device layer. Generally, drawing etch holes in a layout is a time consuming process. Any small error could lead in device that is not released. Sugar2GDSII's automatic etch hole generator can address these problems. To achieve this, the user just needs to specify those structures that need to be released. This can be specified in the element parameter as *release=1* in the netlist. Sugar2GDSII identifies these elements and decides if the etch holes are required or not. Particularly, in case of SOIMUMPS [11] if the dimensions of a structure to be released is less than a specific tolerance, no etch holes are required. They will be released when exposed to an etchant. This type of information is specified in process file. If Sugar2GDSII determines that etch holes might be required, then they can be automatically generated, see Figure 2.

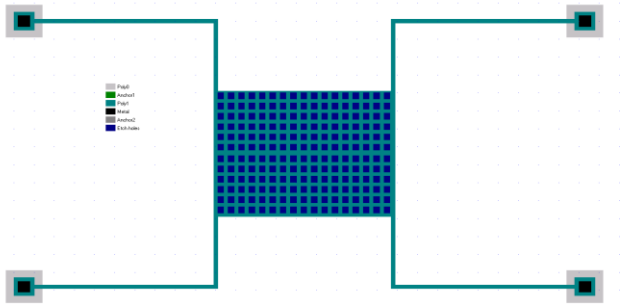

Figure 2: Automatically generated etch holes. The etch holes are shown as small purple squares on the center mass of the crableg flexure of this GDSII file. The dimensions of these etch holes adhere to the specified design rules.

3.2 Multilayer Bonding Pads / Anchors

Sugar2GDSII is able to automatically generate multilayer bonding pads (or anchors) and common-ground pads that are often used in multilayer fabrication processes like PolyMUMPs [10]. Such multi-layered bonding pads are able to connect to any other structural layer in the process. They are topped by a metal layer for wire bonding or probing. Design rule layer size specifications for bonding pads are specified in the Sugar process file. Figures 2 and 3 show examples of our multilayer bond pad that were created in Sugar2GDSII, and imported into a free GDSII layout viewer called CleWin [14].

3.3 Common Voltage Tracers

Sugar2GDSII is able to automatically configure common voltage tracers from each device in an array to a shared bond pad, which is useful for achieving a common ground between a multitude of devices. Such tracers are useful for reducing chip real estate by reducing the need for a large ground pad for each device. Such tracers are also

useful for decreasing the time to probe device arrays, by requiring the re-positioning of one probe instead of two probes. And common ground tracers are useful for reducing the possibility of undesirable voltage loops, which is beneficial for side by side comparisons of actuators. We show an example of automatically-generated tracers attached to a common ground in Figure 3.

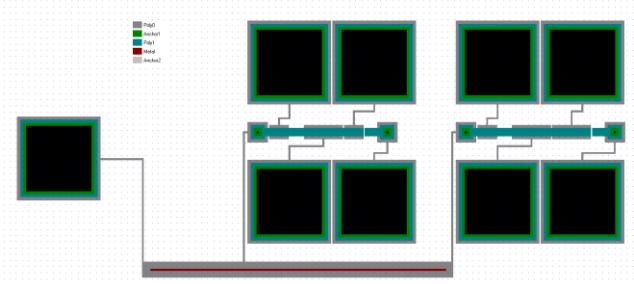

Figure 3: Automatically-generated tracers connect to a common ground. The tracers are shown in gray color (poly0 layer of PolyMUMPs [10]). The tracers from each devices in an array connects to a common ground pad (far left). Also shown is an additional metal layer on the tracer (red layer) that is provided for increased conductivity.

3.4 Parameterized Array of devices

A most-efficient feature in Sugar2GDSII is its ability to easily generate parameterized layout arrays of MEMS. Designers often layout an array of devices with slightly varying dimensions. This is often done to explore the dependence of particular design parameters on performance, or determine the limits of linearity, limits of fabrication, etc. With conventional CAD, changing the dimensions of complicated device geometries can be difficult. It often requires the designer to recreate large portions of the design configuration, which can be tedious and error prone. If a large, varying array of devices must be configured, several hours to days may be spent on designing, debugging, and re-designing before the array is ready for layout submission. Using Sugar2GDSII, the user can reduce this time to seconds or minutes. In addition, our tool is able to simulate the entire array and plot the performance manifold, or optimize the design to achieve a particular performance metric such as resonance frequency.

4 TESTCASE

To create layout, the user loads a reference device and selects which parameters to modify along the rows and columns of the array, and the step size of parameter change. The parameters might be the number of comb fingers or the gap between the fingers, the orientation of the device or the size of the proof mass, etc. If a variation of the design is not already in our library, a new device may be configured using Sugar and easily imported into SugarCube. Upon clicking the *Layout* button, Sugar2GDSII automatically generates an array of devices with required bonding pads, etch holes, and tracers that connect to a common ground.

For example, in Figure 4 we show the SugarCube interface as shown in Matlab. The features shown here are also what is seen in the online version on the nanoHUB. The right-most widow consists of file management controls, followed underneath by select parameters. Which parameters are displayed for the user are chosen in the netlist. Although every parameter of the design can be modified, usually the user will find the most significant parameters of the design appear in this window. Default parameters are initially given; however, the user is able to change one or all parameters. Parameter values may be changed using a numeric pad or by dragging the slider. Parameters remain constant during simulation or layout if only the leftmost column is filled in. If the middle (division) column and rightmost (max value) is filled in, then that particular parameter will be swept during simulation or layout. Currently, a maximum of two parameters may be swept. Once parameters are chosen, an array can be laid out by pressing the *Layout* button, or various simulation options can be investigated.

The lower left window in Figure 4 shows the result of static simulation due to a couple of swept parameters. Its corresponding layout array is shown in Figure 5.

The bond pads of the devices may be aligned vertically and horizontally to facilitate automated probing, or the devices may be positioned closely in a nonrectangular array to conserve chip real estate.

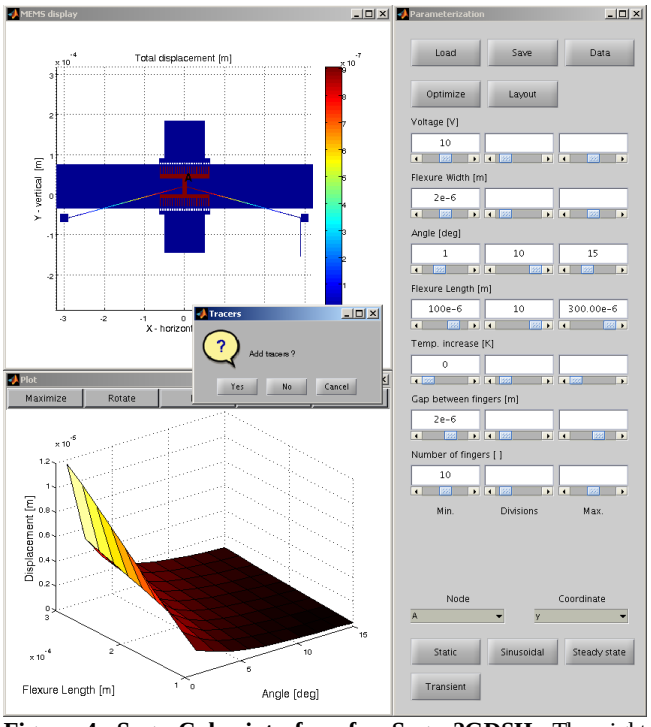

Figure 4: SugarCube interface for Sugar2GDSII. The right window consists of file controls, parameter options, and simulation options. The upper left window shows the deflected structure, which may be rotated in 3D. The lower left window shows the result of pressing the *Static* simulation button. The result of pressing the *Layout* button is shown in Figure 5.

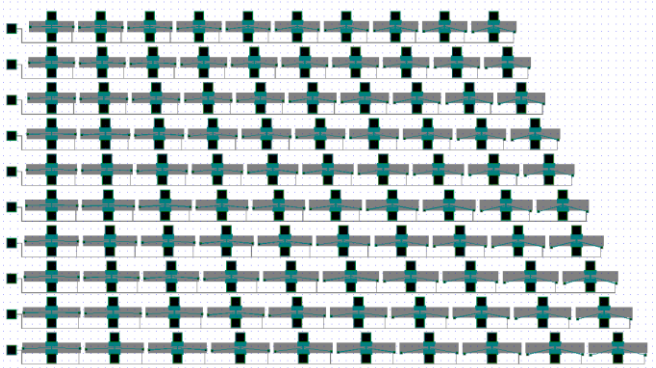

Figure 5: Layout result from Figure 4. By loading a ready-made MEMS configuration into SugarCube, a novice user can easily manipulate sliders to specify parameters value for layout and or simulation. The resulting GDSII file image shown here was captured using the free CleWin layout viewer [14].

5 CONCLUSION

We presented an efficient online tool for both simulating and laying out ready-made MEMS that is amenable to novices and may reduce the time that many expert MEMS usually spend on design. Some of the most tedious and time-consuming aspects of layout are replaced by automated processes in our tool. These procedures include new advances such as coupled parameterized simulation and layout arrays, and automatically-generated common ground tracers. Other useful procedures include automatically-generated etch holes, anchor and bonding pad layers, and backside etch layers.

REFERENCES

[1] Cadence Design Systems, San Jose, California, USA

[2] Coventorware Designer http://www.coventor.com/mems/designer/MEMS-design-layout-and-solid-model-tool.html

[3] Intellisense Blueprint http://www.intellisense.com/modules/Blueprint.html

[4] softMEMS, 2391 Nobili Ave, Santa Clara, CA 95051, USA

[5] E. Vandemeer, *Nodal Design of Actuators and Sensors (NODAS)*, M.S. Thesis, May 1998, Carnegie Mellon University, Pittsburgh, PA

[6] R. Johnstone, Automated Design For Micromachining, 2006. http://www.sfu.ca/adm/index.html

[7] MEMSCAP Inc., 12 TW Alexander Drive Building 100, Research Triangle park, NC 27709, USA http://memscap.com

[8] GDSII Format, http://boolean.klaasholwerda.nl/interface/bnf/gdsformat.html#intro

[9] GDSII Stream Format, http://www.buchanan1.net/stream_description.shtml

[10] C. Allen, H. Greg, M. DeMaul, W. Steve, and H. Busbee, "PolyMUMPS Design Handbook, a MUMPS process", 2005.

[11] C. Allen, H. Greg, M. DeMaul, W. Steve, and H. Busbee, "SOIMUMPs Design Handbook, a MUMPS process", 2009.

[12] P. Marepalli and J. V. Clark, "An Online Tool for Investigating the Performance of Ready-Made Parameterized MEMS", International Conference on Modeling and Simulation of Microsystems Anaheim CA, June 21-25, 2010.

[13] J. V. Clark and K. S. J. Pister "Modeling, Simulation, and Verification of an Advanced Microdevice Using Sugar," Journal of Microelectromechanical Systems, Vol 16, No 6, Dec 2007.

[14] CleWin, WieWeb Software, Achterhoekse, Molenweg 76, 7556 GN Hengelo, The Netherlands http://www.wieweb.com/nojava/layoutframe.html