# iSugar: A Systems Design Framework that Integrates Lumped, Distributed, and System Analyses

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

[1]School of Mechanical Engineering, [2]School of Electrical and Computer Engineering,
[3]Discovery Park, Purdue University, West Lafayette, Indiana, USA

## ABSTRACT

We present a systems design framework called iSugar that integrates lumped, distributed, and system level analyses in a Matlab environment. For lumped analysis we use Sugar for its ease of device configuration, parameterization, and layout capabilities; for distributed analysis, we use COMSOL for its transparent interface; and for system analysis we use SIMULINK for its simple graphical building-block style of modeling. We also accommodate SPICE circuit analysis. For modeling some systems more completely, it may be necessary to model components of the system using different numerical methods so that computational efficiency is optimized without sacrificing model accuracy. A few commercial tools have the ability to integrate with MATLAB and be controlled by SIMULINK. However, iSugar has the ability to control all aspects of the integration from within itself. Doing so facilitates a more holistic approach to design and analysis. We demonstrate several benefits gained from Sugar's efficient and versatile capabilities.

***Keywords***: Lumped analysis, distributed analysis, system analysis, Sugar, SPICE, layout

## 1 INTRODUCTION

Comprehensive MEMS design and analysis often requires a complicated mix of multiple modeling domains and numerical methods. Modeling domains might include electrical circuits, mechanical flexures, electromagnetic radiation, noise, packaging, temperature, pressure, non-inertial forces, various parasitics, and coupling between the domains. An example of such coupling is electrical current passing through a flexure. As the structure heats and expands, its resistivity and resonance frequency will be affected. Although theoretically possible, it is not computationally efficient to represent every aspect of a system using large sets of partial differential equations. Depending on the level of analysis required, some solutions methods provide good computational efficiency at the cost of losing high-order detail. It is this level of detail that an analyst typically considers when determining which methods to use when modeling various system components.

There are many stages in a design cycle. These might include modeling, simulation, optimization, layout generation, process design, system integration, fabrication, calibration, and performance testing. Due to the diverse methodologies involved in handling each stage, specialty CAD tools are often used. To name a few, there are tools that specialize in multiphyiscal distributed [1], [2]; that specialize in layout [3], [4]; that specialize in circuit analysis include [5], [13]; and that specialize in system level analysis [6], [14]. At times it may be necessary to create different versions of the same device if working between modeling domains. CAD for MEMS tools such as [7] and [8] have addressed this need by being able to plug into Cadence, MATLAB, SIMULINK, and others.

Without a hierarchical tool to facilitate seamless integration between a system of tools, a holistic approach to analysis can be difficult. This need is being addressed in iSugar with its ability to fully configure and control all aspects of lumped, distributed, and system level integration within the iSugar tool itself. That is, the efficiency and versatility our MEMS netlist language can be used to not only configure advanced structural designs, but can also be used to specify SPICE circuits, configure the geometry and boundary conditions for components that require distributed or finite element analysis (FEA), control SIMULINK elements, and layout the resulting device for fabrication. Multi-objective optimization features are also available in iSugar, including the ability to determine geometry given desired performance. To facilitate user-modifications, iSugar is open source. Our tool should appeal to users that desire Sugar-style MEMS design with the addition of more sophisticated modeling capabilities.

The rest of this paper is organized as follows: In Section 2 we present the integrated tool framework of iSugar. In Section 3 we describe the methodologies used to integrate each component of our framework. In Section 4 we discuss on how this framework might be useful to the MEMS community. And we summarize this effort in Section 5.

## 2 FRAMEWORK

Our objective with iSugar is to explore extending Sugar's versatile design methodology into areas that are better modeled by distributed analysis, control theory, digital signal processing, etc. Previously, Sugar's modeling capabilities were limited to parameterized lumped models, which meant that models for systems components had to already exist. Although many MEMS can be decomposed into a small set of commonly-used components, such as small deflection flexures, linear comb drives, and simple plates, more complex components such as those with unusually-shaped structures, or those requiring fluid

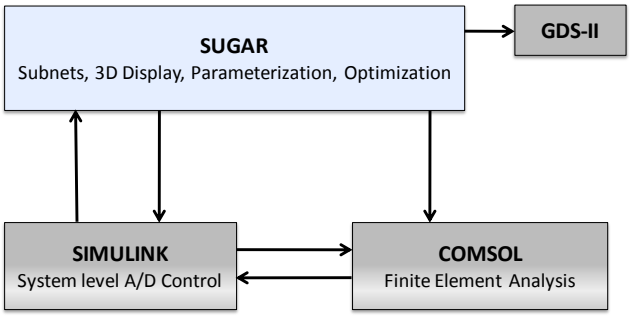


**Figure 1: iSugar framework.** Arrows indicate data flow directions. By hiding the complexities involved in finite element analysis and layout packages, this framework simplifies and quickens the MEMS design and engineering path from idea to fabrication. The user just needs to create a MEMS design using Sugar's simple netlist description language. The system may be controlled from within the netlist or through SIMULINK. Though seamless integration, COMSOL FEA models are automatically generated through iSugar and the resulting building COMSOL building block is available in SIMULINK. Finalized designs may be exported to layout in GDS-II format for fabrication.

dynamics or electrodynamics, could not be fully accommodated in earlier versions of Sugar. The present version is seamlessly integrated through Sugar. That is, it is not necessary for the user to learn how to use the other tools that iSugar is integrated with to take advantage of their benefits. Although iSugar is readily available and open source, the tools that we have integrated it with (MATLAB, SIMULINK, and COMSOL) are available commercially.

COMSOL is a distributed analysis tool that is based on a finite element analysis (FEA). It has a wide range of capabilities to model and simulate multiple energy domains, which is especially important in a field like MEMS. The accuracy of complicated models computed by COMSOL is usually better than those computed by Sugar; however, Sugar is usually more accurate for very simple models if they can accurately be expressed analytically, which can be directly implemented in Sugar. A useful feature in COMSOL that we exploit is COMSOL Script, which is based in MATLAB. That is, every operation in COMSOL can be performed from MATLAB's workspace. This allows users to effectively control all COMSOL capabilities from within iSugar. This also allows parameterized designs that are difficult or too time-consuming to configure within COMSOL to be easily configured in iSugar and then seamlessly imported into COMSOL.

SIMULINK is a system-level simulation tool that is based in MATLAB. It uses graphical building-blocks to configure systems. SIMULINK has a large library of building blocks that span a wide variety of modules including control theory, digital signal processing, COMSOL, Sugar, etc. For instance, SIMULINK can be used to impart feedback and control signals, or environmental disturbances such as non-inertial forces, temperature fluctuations, or noise, etc. Like COMSOL, SIMULINK operations can also be carried out the MATLAB workspace, which we exploit with iSugar. The seemless integration of iSugar with SIMLINK allows for parametric optimization of the MEMS component as its performance is explored in a more complete system.

It is often the case that optimizing single components alone does not yield an optimized system with the components assembled. However, iSugar allows the user to explore a more holistic approach to system analysis. We show iSugar's framework in Figure 1, which indicates data flow directions between its integrated packages.

# 3 METHODOLOGY

## 3.1 Integration of Sugar with COMSOL

Configuring geometries in COMSOL and many other CAD tools is typically done by using geometric Boolean arithmetic. That is, there is a basic set of parameterizable shapes, such as spheres, boxes, etc., and by applying a combination of translations, rotations, unions, intersections, etc., a desired structure is obtained. This common method of construction can be difficult and time-consuming for intricate structures like many MEMS. Parameterization is also difficult because shapes are usually positioned on a global, rather than local, reference frame; and sometimes the number of shapes may need to change. For instance, if the lengths of flexures change, then the elements that they are connected to may need to be repositioned automatically. Or if the number of comb fingers needs to change, shapes may need to be created or deleted automatically.

To overcome this limitation, we developed an algorithm called *cho2comsol* that converts geometry configured in Sugar into geometry that can be imported into COMSOL. We have previously reported on the efficiency and ease of configuring geometries using a parameterized Sugar netlist in [9], where an advanced MEMS with over one thousand shapes was configured using about 20 lines of netlist code. With our conversion algorithm, users can more efficiently define their intricate geometries in Sugar and import them in to COMSOL for finite element analysis. For example, in Figure 2 we show a microrobot design that was configured in Sugar and then imported into COMSOL. It is important to note that modifying this design is very easy to do Sugar, yet very difficult to do in COMSOL.

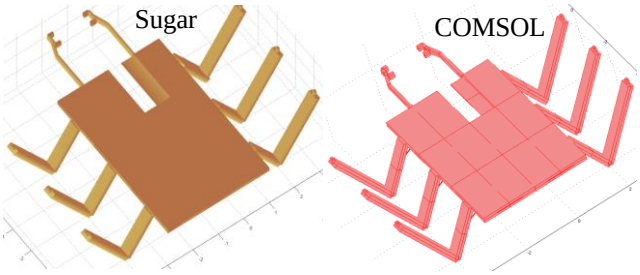


**Figure 2: Sugar to COMSOL.** Creating intricate geometries in many FEA tools can be time-consuming. And making them parameterizable can be difficult. However, doing so in Sugar is quick and easy. We show a microrobot from [11] that was easily configured in Sugar and then easily imported in COMSOL.

Using Sugar's set of parameterized geometries we create corresponding geometries in COMSOL as follows. COMSOL has a scripting language based in MATLAB. Each geometry object in COMSOL may be defined by a geometry function that describes its shape. For example, rect2 for a 2D rectangle, circ2 for a 2D circle, etc. [10]. Parameters to these functions include dimensions, position, orientation, etc. Similarly, there are COMSOL Script commands for defining material properties such as Young's modulus, etc.; and boundary conditions such as fixed, free, roller, electric potential, etc. Our Sugar-to-COMSOL algorithm can automatically generate the required COMSOL Script file for each geometric object that is configured in Sugar. However, since Sugar also does layout, some layout-specific geometries are optionally converted to COMSOL. For example, large wire bonding pads and tracers are often not converted over. This is because the dynamics of such objects are usually not required and their presence in COMSOL would be a large computational expense.

## 3.2 Verification of Lumped Analysis

Although lumped analysis is much more computationally efficient than distributed analysis, this is usually done at the cost of refined information. For example, distributed analysis often provides temperature, charge, and stress distributions on structures; yet, lumped analysis is often limited to the effective equivalent information lumped at the nodes. Moreover, lumped modes are often created by reducing various types physics involved in the problem to the bare minimum. So determining the accuracy and limits of lumped models is often necessary; and even more so, determining the accuracy and limits of a system of lumped models due to possible proximity effects is often necessary.

Such verification can be done more easily than before using iSugar. This is because we are able to not only import geometric and material properties from Sugar to COMSOL as discussed in Section 3.1, but we also have automated the application of actuation efforts, meshing, and solver analyses. In this way, after configuring a design in Sugar, users may automatically verify their assembled models and simulations in iSugar. Although this process requires the user to have the COMSOL engine, iSugar's automation implies that the user is not required to have expertise in the use of COMSOL.

We show an example of iSugar's automatic verification in Figure 3. After a serpentine-flexure structure created in Sugar (with just 9 lines of netlist text) is automatically exported into to COMSOL, boundary conditions applied, meshed, and simulated, all with just a single command within the MATLAB workspace or within SIMULINK.

## 3.3 Integration of Sugar with SIMULINK

A goal of MEMS designers is to predict the performance of their systems under realistic operating conditions. Modeling such systems more completely than convention includes interface electronics, packaging, temperature variations, external vibrations, electromagnetic radiation, non-inertial forces, etc. A system level simulation tool can be used to efficiently control such disturbances, since such sources do not require as detailed modeling as the MEMS structure. In iSugar we integrate Sugar with SIMULINK by implementing a SIMULINK Sugar-block. These blocks can be used to perform different Sugar operations like simulating static, modal, and transient performance of MEMS, displaying the MEMS in their deflected states, etc.

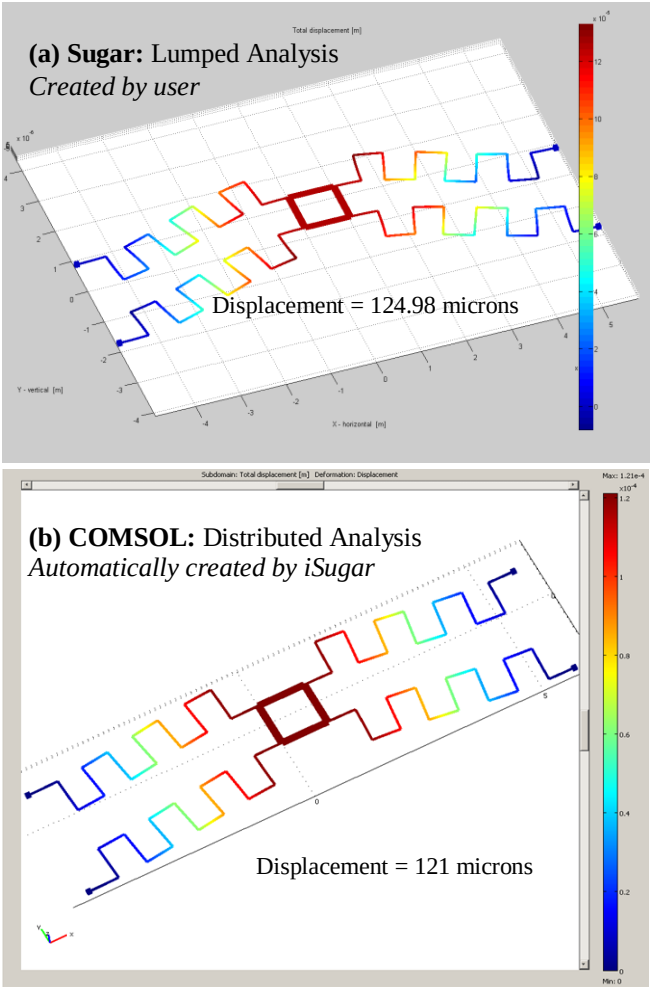


**Figure 3. Verifcation of Lumped Analysis.** An example of lumped analysis verification with distributed analysis is shown. A lumped model of a serpentine flexure is shown in (a) using just 9 lines of netlist text. This lumped model can be automatically converted into COMSOL for distributed analysis, as shown in (b). This conversion process includes positioning and rotating structural elements, applying contraint and effort boundary conditions, meshing, selecting solver and solver parameters, and plotting generation the results. This conversion is done with a single command in MATLAB; i.e. no interaction with COSMOL is required by the user. In regards to the validation of static displacement for this model, the relative error of Sugar with respect to COMSOL is 3.3%.

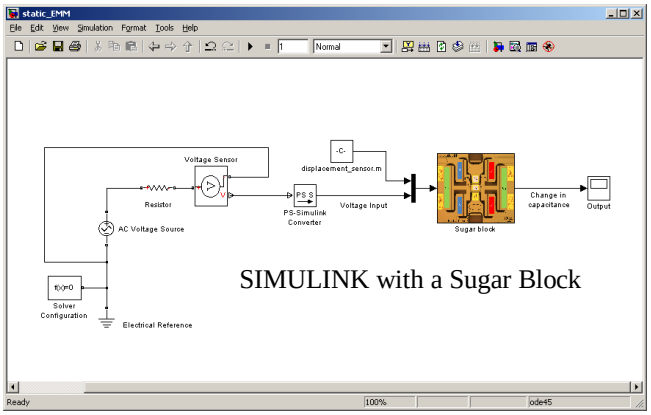


**Figure 4. Integrating Sugar to SIMULINK components.** With the integration of Sugar to COMSOL and SIMULINK, iSugar shares the Sugar's ease of use, COMSOL's depth in simulating multi-energy domain problems, and SIMULINK breadth in solving system level problems. In this example, we show a Sugar block can be integrated to a system level circuitry inside SIMULINK.

In use, the user is able to interconnect one or more Sugar blocks of MEMS, one or more COMSOL blocks, and a host of other SIMULINK blocks to emulate a more complete system. In Figure 4, we show an example of a system level configuration in SIMULINK that connects control circuitry to a MEMS Sugar block. The output of the Sugar block is defined by the user. For instance, the output might be the mechanical deflection of node, resonance amplitude, capacitance of a comb drive, etc.

## 3.4 Integration of Sugar with SPICE

SPICE [5] is a popular tool known for its breadth in simulating integrated circuits. Sugar was initially created to be a MEMS version of SPICE. In iSugar, we integrate Sugar with SPICE by enabling the user to write pure SPICE netlist syntax within a Sugar netlist. A pre-processor separates the SPICE circuit part of the netlist from MEMS part. Once these partitions are identified, either one or both of the MEMS structure and or SPICE circuitry can be imported and simulated in COMSOL. That is, COMSOL includes a SPICE simulation engine.

# 4 CONCLUSION

In this paper we presented our systems design framework called iSugar that integrates lumped, distributed, and system level analyses. Sugar is the tool used for lumped analysis, COMSOL is used for distributed analysis, and SIMULINK for system level simulation. A common attribute in these tools is that their scripting is based in MATLAB, which we exploit in iSugar to seemlessly integrate these tools. With iSugar users are also able to integrate SPICE analysis and layout in GDS-II format. The automation and control of these tools through iSugar is expected to enable greater efficiency and versitility in modeling MEMS.

# REFERENCES


[1] COMSOL, Inc. 1100 Glendon Avenue 17th Floor, Los Angeles, CA, 90024. http://www.comsol.com.

[2] ANSYS, Inc. Southpointe, 275 Technology Drive, Canonsburg, PA 15317, www.ansys.com

[3] L-Edit- Tanner Research, 825 South Myrtle Avenue, Monrovia, CA 91016, USA. http://www.tanner.com

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

[5] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," *ERL Memo*. No. UCB/ERL, Vol. M75/520 (1975)

[6] SIMULINK - Mathworks, Natick, Massachusetts, U.S.A. http://www.mathworks.com/products/simulink

[7] Coventorware's Architect and MEMS+ for Systems Design. http://www.coventor.com/mems-system/mems-system-design.html

[8] Intellisense's Synple and EDA Linger for Systems Design. http://intellisense.com/MEMS-SoC/MEMS-SoC.html

[9] Jason Vaughn Clark and Kristofer S. J. Pister, "Modeling, Simulation, and Verification of an Advanced Micromirror Using SUGAR", *Journal of Microelectromechanical Systems, Vol. 16*, No. 6, December, 2007, pp.1524-1536.

[10] COMSOL Multiphysics Matlab Interface Guide, Version 3.5, COMSOL Inc.

[11] J. V. Clark, D. Bindel, W. Kao, E. Zhu, A Kuo, N. Zhou, J. Nie, J. Demmel, Z. Bai, S. Govindjee, K. S. J. Pister, M. Gu, A. Agogino, "Addressing the Needs of Complex MEMS Design," *Proceedings IEEE The Fifteenth Annual International Conference on Micro Electro Mechanical Systems*, Las Vegas, Nevada, January 20-24, 2002.

[12] P. Marepalli, J.V.Clark, "-A Systems Design Framework Based in Matlab That Integrates Sugar, Spice, Simulink, Fea Comsol, and GDS-II Layout", International Conference on Modeling and Simulation of Microsystems, Boston, MA, USA.

[13] Cadence Virtuoso Spectre Circuit Simulator, Cadence Design Systems, Inc. 2655 Seely Avenue, San Jose, CA 95134

[14] Ptolemy Project – Heterogeneous Modeling and Design. http://ptolemy.eecs.berkeley.edu