# The SALVADOR simulation framework

Michael Weiner[1], Daniel Arumi Delgado[2], and Salvador Manich[2]

[1] Technische Universität München, Munich, Germany, m.weiner@tum.de
[2] Universitat Politècnica de Catalunya, Barcelona, {daniel.arumi,salvador.manich}@upc.edu

## 1 Introduction

Analog simulation is an essential tool for the design space exploration of circuits that are difficult to describe on a more abstract level. These types of circuits often come from the hardware security domain – e.g. ring oscillator based physical unclonable functions (RO-PUFs), or microprobing detectors. The correct operation of such circuits usually depends on low level parameters such as, for example, component dimensions or number of components connected in series or parallel. Finding a good set of design parameter values for such designs is a tedious task that often requires large amounts of simulations.

We present a lightweight simulation framework named SALVADOR (Simulation Automation Library for Verification and Analysis of Design Operating Regions) that combines template-based netlist generation, parallelized execution of a command-line simulator such as hspice or spectre, as well as evaluation of results and important simulator messages. Compared to large frameworks such as the state-of-the-art-tool Cadence Virtuoso [1], all parts of the netlist can be parameterized generically: this allows avoiding the time-consuming and error prone step of manual netlist creation whenever the Virtuoso support for netlist parameterization comes to its limits. Simulator processing messages such as the number of warnings or execution time can be included in the results: this allows fast plausibility checks and can be a helpful indicator for potential netlist errors. Its lightweight open-source code base and flexible design of building blocks allows easy adaption to different use cases.

## 2 Automating simulations with SALVADOR

A sample workflow that was used in the design phase of the Low Area Probing Detector [2] is shown in Figure 1. The workflow can in general be divided into the context-free template instantiation and the simulation-specific execution and result evaluation part.

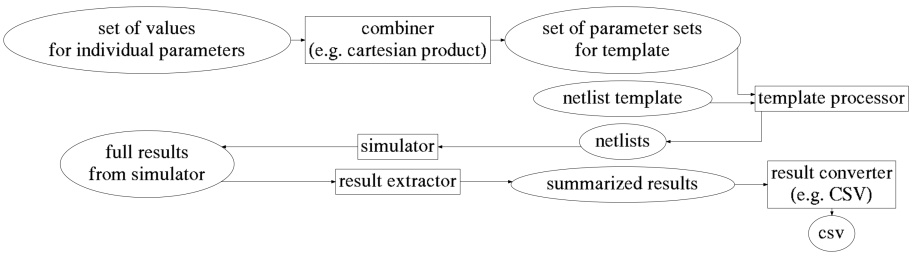


**Fig. 1.** sample SALVADOR workflow

### 2.1 template instantiation

In Figure 1, the *set of values for individual parameters* is a file that specifies lists of concrete values for each parameter in the netlist template. The **combiner** then generates the desired combinations of parameter values; our use case implies the generation of all possible combinations of values of each parameter, i.e. the cartesian product. Our implementation of the combiner expects the input data to be Python code – in particular, one `dict` in which the keys correspond to template parameter names and the dict values are iterables containing the possible values. This format is syntactically similar to JSON (JavaScript Object Notation), but has the advantage that it allows comments and thus facilitates annotation of the input data.

The output of the combiner is a *set of parameter sets* in which each element of the outer set is a set in which each parameter is assigned one concrete value. The cardinal number of the outer set is equal to the number of template instances, in our context the *netlists*, to be created by the **template processor**. The second input of our template processor is the *netlist template* itself. In our implementation, this is a directory containing plain text files; the parameters are specified using the Python Format String synax.

Up to this point, the workflow is completely context free; the only constraint is that the template and the parameter names and values must be printable text. This gives us a high degree of freedom with respect to the selection of the simulator as well as its configuration. The workflow up to the template instantiation can theoretically even be used in completely different contexts that are unrelated to analog simulation.

### 2.2 actual simulation and result extraction

The workflow starting from the **simulator** is application dependent. The variety of simulation environments and configuration options is very large, so this section instead focusses on the generic parts as well as ideas about optimizing the workflow.

In our example, each template instance inherits a start script from the template that runs the actual simulator and terminates as soon as the simulator has terminated. The simulator shall be called in such a way that all output files (e.g. logfiles, results) remain in the instance directory and do not influence other instances.

The start scripts of each instance are called one after another by a "meta-level" program that supports concurrent execution of a given number of start scripts. In our implementation, the `bash` command `jobs` is used for that purpose.

The **result extractor** is completely application specific: It must know the location and format of the results, which is usually defined by the simulator used as well as the contents of the template.

At this step, we consider it useful to not only extract the actual results, but also meta-information such as number of warnings and notices as well as execution time. This can be helpful to detect anomalies in the simulation environment. Our workflow, for example, once revealed an unconnected net that only produced a warning – but no error – and the results turned out to be inaccurate. If such a situation only occurs in a subset of templates, it can be hard to detect without displaying such "meta-information".

## 3 Summary

SALVADOR is a generic analog simulation automation framework that supports all simulators that can be configured by printable text. This is the case for state-of-the-art tools such as spectre or hspice. Compared to state-of-the-art frameworks such as Cadence Virtuoso, it gives a higher degree of flexibility and avoids errors caused by manually repeating non-automatable steps. In addition to result extraction, it allows to parse the logfile to detect possibly important warning messages that would otherwise not attract attention. SALVADOR is open source software that is, as of October 2016, available at `https://gitlab.lrz.de/michael.weiner/salvador`.

## References


1. Cadence Design Systems, Inc.: Virtuoso Analog Design Environment Family. `https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/custom-ic-analog-rf-design/virtuoso-analog-design-fam-ds.pdf` (2014)
2. Weiner, M., Manich, S., Sigl, G.: A Low Area Probing Detector for Power Efficient Security ICs. In: Radio Frequency Identification: Security and Privacy Issues. vol. 8651. Oxford, UK (July 2014)


## Acknowledgements


This work has been partially financed by Spanish TEC2013-J41209-P government project.