Challenges in RF Simulation

Ken Kundert
Designer's Guide Consulting, Inc.

Version 1, 13 June 2005

RF simulators exploit the nature of RF circuits to overcome bottlenecks that prevent the use of general purpose simulators. While this makes RF simulation possible, it also leads to two significant problems. First, it is difficult to apply RF simulation to larger circuits that are not exclusively RF. Second, it leads to a large number of very specialized analyses, and it is difficult for designers to effectively learn and use the ones they need. Solutions to these two problems are explored in this paper.

This paper was first presented at the 2005 RFIC Symposium. It was last updated on May 12, 2006. You can find the most recent version at www.designers-guide.org. Contact the author via e-mail at ken@designers-guide.com.

1 The RF Simulation Trade-Off

RF circuits exhibit characteristics that make them difficult and often impractical to simulate using general purpose simulation methods, like those of SPICE [1]. RF simulators, such as SpectreRF, are specifically formulated to exploit the innate characteristics of RF circuits to overcome these limitations. In doing so, they trade generality for the ability to simulate RF circuits efficiently.

RF circuits are designed to process high frequency modulated carriers. It is the presence of the high frequency carrier that slows traditional circuit simulators. The designer is most often interested in the effect of the circuit on the modulation because it carriers the information being communicated through the RF system. However, the modulation signal is quite slow relative to the carrier. The slow modulation signal requires that a long time interval be simulated and the high carrier frequency forces the simulator to use very small time steps. The two combine to make the simulation very expensive. RF simulators exploit the fact that the carrier is highly repetitive to reformulate the problem. Instead of trying to follow a very detailed high-frequency signal with no assumed structure, the RF simulator instead assumes it is following a low-frequency signal as it modulates a high-frequency periodic signal. In doing so, RF simulators naturally place certain constraints on the signals that limit their applicability on non-RF circuits.

If these constraints are not satisfied, it may be that the RF simulator will not operate at all, or it may operate but with substantial overhead when compared to general methods.

The fact that RF simulators operate by exploiting specific characteristics of RF circuits creates two substantial issues.

1. An RF simulator requires a certain homogeneity in the circuit that it is to analyze, which is problematic as we try to take on larger more diverse circuits.
2. RF simulation algorithms tend to be very specific in the type of circuit they can be applied to and the result they produce. As a consequence, a wide variety of special purpose RF analyses have been developed. In SpectreRF there are currently 11, with variants on many, and that number continues to grow.[1] This places a substantial burden on users, forcing them to be expert in all of the analyses and know when and how to apply each one.

These two issues represent the key challenges facing the RF CAD community. Without addressing these issues, RF designers will continue to struggle to use RF simulation and will be unable to perform full chip verification.

2 Circuit Diversity

Today, RF simulators are quite capable of simulating large RF blocks, but they tend to fail when asked to take on either circuits that contain non-RF blocks, or circuits with multiple RF blocks with dissimilar natures. As just pointed out, RF simulation techniques place constraints on the circuit, and as the circuit becomes larger and more diverse, it is more likely that the constraints will be violated by some part of the circuit, making it impractical or impossible to apply the technique. Thus, despite substantial

1. PSS, PAC, PXF, PNoise, PSP, Envelope, QPSS, QPAC, QPXF, QPNoise, QPSP.

increases in capacity in recent years, RF simulators really only support the design and verification of RF blocks, not the entire chip. For example, it's usually possible to simulate an RF signal path, but it is generally not possible to include the ADCs at the end of the path or the frequency synthesizer that generates the LO.

In addition, as a natural response to the increasing quantity and decreasing quality of transistors in nanometer-scale processes, designers increasingly turn to self-calibrating and self-compensating circuitry. While such circuitry used to be purely analog, it is becoming more common to implement it using digital approaches. Such circuitry almost certainly prevents the use of RF simulation. For example, an increasingly popular approach to linearizing RF power amplifiers is to use feedback through a DSP to control errors in the magnitude and phase of the transmitted signal.

At this point it is not clear what should be done to address this problem. It may be possible to develop new, more flexible RF algorithms that are efficient on circuits that contain both RF and non-RF blocks. However, it is currently not easy to see how this would be done and there has been no progress in this direction.

2.1 Co-Simulation

The approach being tried today is one of co-simulation. Here a composite simulator is created by combining two or more core simulators. Each of the core simulators forms a kernel in the larger simulator. The circuit is partitioned with each partition assigned to a kernel. The kernels then cooperatively run the simulation. Primitive versions of this approach are available today in which the partitioning is performed manually by the user and the type of simulators is heavily constrained. For example, Agilent currently provides the ability to co-simulate using Circuit Envelope on the transistor-level RF blocks and Ptolemy on system-level blocks. In the future, it would be useful to have RF simulators also co-simulate with circuit simulators such as Spectre, with timing simulators such as UltraSim, and with logic simulators such as NCsim.

There are two main challenges to advancing this basic approach: partitioning the circuit and wiring the kernels together so that they co-simulate efficiently and reliably.

It would be extremely difficult to automatically identify the RF and non-RF sections of a design and then further identify the best way of splitting them. So difficult, in fact, that it is best to simply assume that it is impractical for the foreseeable future and move on. Thus, the circuit must be partitioned manually. In certain cases the partitioning is easy, as with Circuit Envelope and Ptolemy. Here the two kernels use two different types of components, and the choice of which type of component to use is obvious. Circuit Envelope uses primitive components such as resistors, capacitors, inductors, transistors, etc. Ptolemy, on the other hand, uses high-level blocks such as summers, multipliers, and the like.

There are three basic challenges involved when gluing an RF simulator to a general purpose simulator:

1. The class of solution (steady-state versus transient, the ability to account for small signals, etc.);
2. the state (must be resettable, being observable and controllable would be helpful); and

3. the coupling of the signals between the simulator kernels (frequency domain versus time domain, etc.).

Most general purpose simulators compute the transient response of the circuit, whereas RF simulators generally produce the steady-state response. The first thing to do is determine what the combined simulator computes. If it computes the steady-state response, then there must be some way of assuring that the general purpose simulator has simulated past the initial transient before producing the final results. The easiest way of doing this is to simply allow the user to specify an initial transient interval; however this is risky and slow. It is not obvious how to choose the duration, and choosing a value that is too small produces inaccurate results. A better approach is to have the simulator monitor the state of the portion of the circuit handled by the general purpose kernel and simply continue simulation until steady-state is achieved to within specified tolerances. However, most general purpose simulators are not designed to support such monitoring, and adding it is generally difficult.

Steady-state simulation is also complicated by the fact that the signals in both kernels must settle, and that the transient kernel must simulate for potentially a long time after the last change to assure its signals have settled. This may very well make steady-state co-simulation impractical in most cases.

If the combined simulator computes the transient response, then a transient-envelope analysis [1] should be employed in the RF simulation kernel. However, even in this case where both simulators formulate their solutions in the same domain there are challenges. Like SPICE, RF transient simulators reserve the right to propose possible solution trajectories, try them out, and then reject them if they are not sufficiently accurate. As such, the combined simulator must also support this behavior. To do so, the RF simulation kernel must be able to ask the general purpose kernel to save its state as it is about to propose a trajectory, and restore the previously saved state if the trajectory is rejected.

Small-signal analyses are generally problematic in co-simulation because the small-signal analysis must be performed on a linearized representation of the entire circuit. Thus, each kernel must be capable of producing a linearized representation of its portion of the circuit, and these must be combined and solved together. None of these simulators are currently able to produce a linearized representation in such a way that it can be treated as a subcircuit and efficiently combined with other subcircuits and solved. With sufficient effort, it might be practical to add this capability to the RF and circuit simulators, but doing so with abstract simulators such as system and logic simulators would be quite difficult.

General purpose simulators invariably operate in the time domain, whereas many RF simulators are harmonic balance based and so operate in the frequency domain. The conversion of signals between the frequency- and time-domains can only be done by after observing the signals for a while. For this reason, when combining frequency-domain and time-domain kernels, the signals must be clearly decomposable into high- and low-frequency components, with the signals in the low-frequency components changing very little over the period associated with the signals in the high-frequency components. This is the same constraint that the time-domain harmonic balance simulators must satisfy anyway (see Fourier-envelope methods in [1]), and so this constraint does not act as a barrier to using a frequency-domain RF kernel. However, it is worth noting that this constraint is not present when using an RF kernel that uses envelope following (a time-domain sample-envelope formulation [1]).

In summary, addressing the full-chip simulation problem by co-simulating with an RF kernel is both difficult and not completely general. And while not the panacea that one might hope for, does offer value. So, one should expect that these types of simulators will slowly become more available and increasingly more capable as suppliers complete the engineering required to glue various simulators together. However, it is hard to imagine co-simulation ever becoming a truly satisfying solution to the full RF chip verification problem. Clearly an innovative new approach is needed.

3 Automating Measurements

RF simulators are circuit simulators that provide additional analyses, tailored for RF circuits, beyond those available in SPICE. The new RF analyses are often patterned after those provided by SPICE. Consider SpectreRF. It provides the *periodic steady state*, *periodic AC*, *periodic noise*, and *envelope following* analyses as the RF versions of SPICE's DC, AC, noise, and transient analyses. However, despite the similarity between the two sets of analyses, most designers find the RF versions to be confusing and difficult to use. In fact, this is one of the main reasons for the slow adoption and propagation of RF simulation.

SPICE provides a small number of relatively generic analyses, and designers are trained from a young age to use them. They become quite adept at using them to perform a wide variety of simulated measurements on their circuits. RF simulators followed this same model, but the RF analyses are based on difficult concepts that are unfamiliar to most designers. Analog and RF designers are incredibly resourceful, and so these difficulties are not an insurmountable barrier, but rather simply represent a substantial investment that each designer must make in order to use the new simulators.

The problem itself stems from the use of analyses to provide the desired simulator functionality. Analyses are a concept that is unique to simulators and do not represent a desired result from the users perspective. Rather analyses are just a necessary intermediate step that must be taken when performing a simulated measurement. It is the measurement that the designer is intrinsically interested in, not the analysis.

3.1 Measurement Description Language

To make a measurement, the circuit must first be configured so that it is in the desired state and has the correct stimulus, then one or more analyses are run, and finally the results from the analyses are processed to extract the desired performance metric. There is often a high level of expertise needed to make a measurement correctly. The measurement itself must be well understood, which can be surprisingly difficult. For example, the document that describes the methodology for measuring jitter in the FibreChannel standard is almost 90 pages long. In addition, considerable sophistication in the workings of the simulator may be needed to properly make the measurement using the most efficient approach, and in a way that provides sufficient accuracy.

What is needed is a library of pre-defined measurements that are put together by experts that could be used by any designer. However, there is a very large number of possible measurements, with many variations, and so it is impractical to hard-code them into the simulator. While it seems like one could use a scripting language to describe the measurements, such as Ocean or the one that comes with SPICE3, there is an issue with reusability. These languages do not support the concept of a naturally reusable measurement, and it is difficult to even approximate them using the relatively generic programming constructs they provide. With these languages, designers have to write their own measurements, which most refuse to do.

What's needed is a measurement description language that allows measurements to be described easily and succinctly in a way that they can be reused on a wide variety of circuits. The measurements would then be written by measurement experts and designers that invest in learning the measurement description language. What follows is a description of how such a language would differ from existing languages [2].

First and foremost, a measurement description language must be able describe measurements in a direct and concise fashion, and the measurements must be naturally reusable. The existing .measure capability of HSPICE is not a measurement description language because it can do neither. It is only capable of describing the way analysis results are processed to compute the final result, and it provides no abstraction mechanism such as parameters that would allow a single set of measures to apply to a variety of circuits.

3.2 Requirements

A measurement description language must provide the ability to define parameterized measurements, where a measurement consists of the following operations:

1. Configure the test bench for the circuit, which includes specifying the stimulus;
2. simulate the circuit with the given stimulus;
3. analyze the resulting circuit response to determine the value of the quantity of interest; and
4. display the results and perhaps supporting information.

In order to be naturally reusable, the measurement must be completely parameterized. As such, there must be no direct access to the circuit from within the measurement definition. Instead, all circuit access must be indirect, through parameters. This implies that all stimulus points and waveshapes, and all observation points must be specified as parameters to the measurement. Consider the measurement description given in Listing 1. It measures the 1 dB compression point of an amplifier, a measure of its nonlinearity [1]. It starts by defining the measurement's parameters, as denoted with the *input*, *output*, and *export* statements. The remaining statements define the measurement process, which starts by defining the waveshape and frequency of the input source, progresses to performing a power sweep while running a periodic steady-state analysis at each power level, and then finally, extracts the compression point from the power transfer curves built during the sweep. Notice that the stimulus point is defined using the parameter *src*; the stimulus waveshape with *start*, *stop*, and *fund*; and the output observation point with *load.*

Once defined, this measurement can be applied to any amplifier-type circuit that has a resistive source and load. Additionally, notice that the measurement is largely self-documenting, which further increases its usability. And that only given the information in the declarations of the measurement, it is possible to build a form, shown in Figure 1, that would allow the measurement to be used from a graphical user interface by a user that

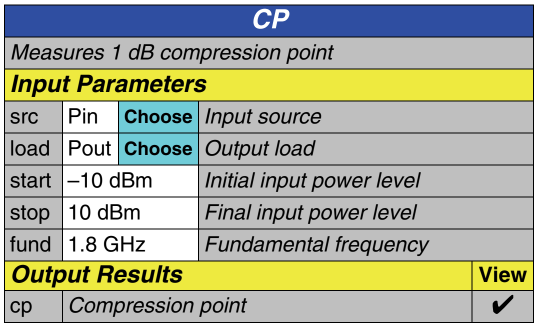
LISTING 1 *A 1 dB compression point measurement.*

```
measurement cp --- "Measures 1 dB compression point" {
    input inst src=Pin from {port} --- "Input source"
    input inst load=Rout from {port, resistor} --- "Output load"
    input real start=–10_dBm --- "Initial input power level"
    input real stop=10_dBm from (start ..) --- "Final input power level"
    input real fund=1GHz --- "Fundamental frequency"
    output point cp --- "Compression point"
    export signal Pout --- "Output power versus input power"

    src:type = 'sine
    src:freq = fund
    foreach src:dbm from swp(start=start, stop=stop) {
        run pss( fund=fund )
        real Vfund = V(load)[1]
        Pout=dBm(Vfund**2/(2*load:r) "W")
    }
    cp=compression_point(Pout,1)
}
```

would never have to see or understand the measurement definition. In this way, sophisticated measurements could be defined by a few experts, and used by a wide variety of users. Thus, one need not learn the language to benefit from it. This is critically important as only around 10-20% of users are expected to invest the time to learn such a language. Another important requirement of measurements is that by the design of the

FIGURE 1 *Form generated for the CP measurement of Listing 1.*



CP			
Measures 1 dB compression point			
Input Parameters			
src	Pin	Choose	Input source
load	Pout	Choose	Output load
start	–10 dBm		Initial input power level
stop	10 dBm		Final input power level
fund	1.8 GHz		Fundamental frequency
Output Results			View
cp	Compression point		✔

language they must be independent or side-effect free. In this way measurements can be freely run in any order or, if suitable hardware is available, run in parallel. To be independent, it must be that any change made to the circuit or the circuit's environment during the course of a measurement is reversed as the measurement terminates.

Finally, it must be possible to call measurements recursively (call a measurement from a measurement), allowing them to become powerful building blocks that can be flexibly combined to build more powerful measurements. For example, the compression point measurement of Listing 1 can be called from a measurement that sweeps the input frequency to determine the minimum compression point over frequency, and that this measurement can be called from one that runs it over the anticipated worst case corners to find the worst case compression point.

3.3 Applications

Once a collection of measurements have been defined and configured for a circuit, it becomes possible for the design system to completely characterize a circuit without any intervention by the designer. This provides considerable opportunity for automation. Imagine making a change to a circuit and then being interrupted by a phone call, and upon returning to work finding that your circuit had already been completely characterized.

If a designer further established target values for each measurement result, then the design system can autonomously determine the quality of a circuit. In this way it becomes easy to summarize the performance of the circuit in the form of a datasheet, with those areas that still need improvement clearly denoted. It would also be possible for the design system to take the next step and apply optimization techniques and propose possibly improved versions.

In summary, a measurement description language and reusable measurements fully address one of the key issues facing the RF CAD community: the complexity of RF simulation. They raise the interaction between the designer and the simulator to a level that is much more natural, and in doing so make designers much more productive. They spend considerably less time thinking about the simulator and more time thinking about their circuit. They also provide the same benefits to analog and mixed-signal designers.

References

[1] Ken Kundert. Introduction to RF simulation and its application. *Journal of Solid-State Circuits,* vol. 34, no. 9, September 1999. Available from *www.designers-guide.org/Analysis*.

[2] Ken Kundert. A measurement description language. Unpublished, available upon request.