# Using an Analog Netlist Generation Tool to Evaluate a Mixed Circuit Verification Framework

Danilo D. Almeida[1], Abner L. P. Marciano[2], and José Augusto M. Nacif[1],
[1]Instituto de Ciências Exatas e Tecnológicas, Campus UFV-Florestal, Universidade Federal de Viçosa, Brazil
[2]Cadence Design Systems
{danilo.damiao, jnacif}@ufv.br, abner@cadence.com

***Abstract*—The growing demand for powerful integrated circuits in terms of power consumption and processing, creates a big obstacle for hardware verification engineers to ensure Integrated Circuit (IC) quality. Actually, IC verification consumes more than half of the development cycle. And with the growing complexity, the verification also increases. When we consider analog blocks in mixed-signal ICs we have an even more challenging verification process. Unfortunately, the environments for mixed-signal verification are inefficient because it is not possible to generate realistic analog outputs to the digital blocks inputs during simulation. This work evaluates ALIAS, a tool that generates digital abstractions of analog circuits for verification. To evaluate ALIAS efficiency we have created an ADC and DAC netlist tool generator, to perform verification in different mixed-signal scenarios and circuit configurations.**

***Index Terms*—Verification, Analog Circuit Abstraction, Converter, ALIAS, ADC, DAC.**

## I. Introduction

The growing demand for performance and reduced power consumption on integrated circuits lead to more and more complex systems. Actually, the Integrated circuit (IC) verification takes more than half of the development cycle [1], and with the expansion in complexity the verification time also increases. When considering analog and mixed signal IC's the verification problem is even more challenging.

When compared to digital circuits, verifying analog circuits suffers from performance issues and the lack of automated flows. However, when verifying a system at digital levels, engineers tend to focus the verification effort at the digital portions of the design-under-verification (DUV). Therein, such analog components are either ignored or replaced with over-simplified versions. On overlooking such components at digital levels, the overall IC quality is compromised. Abner et. al [2] presented ALIAS, a tool that creates precise digitized versions of analog circuits. On this work we present a series of case studies directed on evaluating ALIAS' efficiency. To that end, we have created a tool to generate analog-to-digital (ADC) and digital-to-analog converters (DAC) with selective precision marks.

This paper is organized as follows. We present ALIAS and selected circuits in Section II, followed by a review on related work, that we present in Section III. Methodology, ADC/DAC circuit generator and routing algorithms are presented in Section IV. Sections V and VI present results and future work, respectively.

## II. Background

In this Section we present an overview of ALIAS as well common mixed-signal circuits.

### A. ALIAS

The ALIAS creates digital approximations of analog circuits based in observed circuit behavior [2]. As depicted by Figure 1, the process of creating analog abstraction starts with data acquisition, which is the waveform generated by the analog circuit. Reading from SPICE-based simulations [3] data, ALIAS discretizes continuous analog points based on a parsing specification which goes from global settings, like the interval into which the waveform should be sampled, to local settings, as how each signal should be sampled. Figure 2 illustrates wave sampling process.

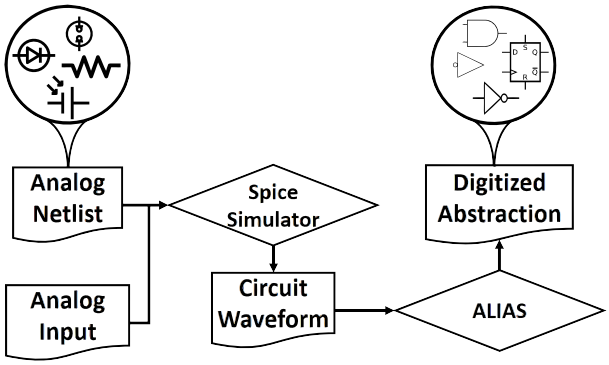


Fig. 1: An Example of ALIAS use flow.

The next step is on understanding the analog data, now in digital domain, which has been made based on the observation that electronic systems usually start at an idle state, until they is signaled to perform an action, to then perform it and settle back in an idle state. ALIAS uses this idea to transform an analog behavior on an abstract model in form of Verilog Code or a DOT file. Figure 1 illustrates the steps to create a digital abstraction using ALIAS, starting in analog netlist input ending in the digitized abstraction.

### B. Mixed-Signal Circuits

The selected circuits convert digital signals in analog and vice-versa. They are used for creating an interface between real and digital world and are used in various groups of electronic

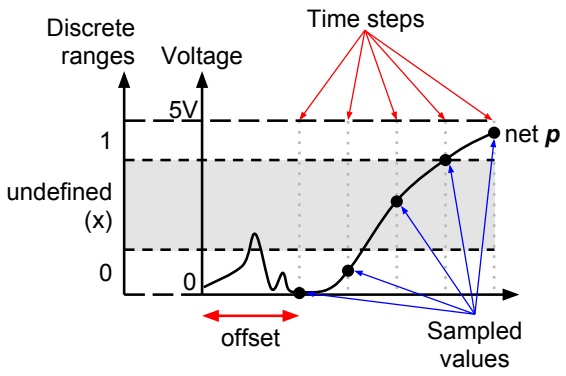


Fig. 2: Sampling wave process [2].

products used daily (e.g. Cellphones, Sound Mixers, Radio Controllers). Figure 3 illustrates the interface created for this type of electronic circuit.

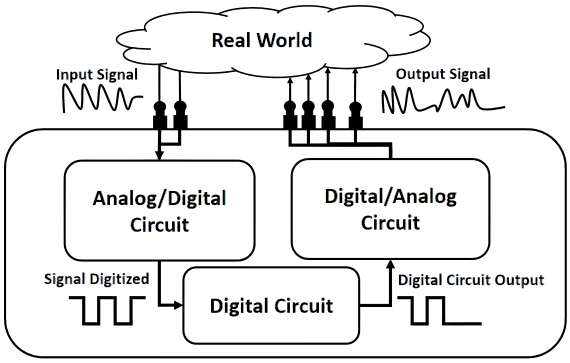


Fig. 3: Mixed Signal Circuit.

*1) DAC R/2R:* On this case study we target a R/2R DAC, formed by a simple resistor network and an analog comparator.

The output can be seen as the sum of each bit voltage, and value assumed for each bit is represented for expression $V_{out} = \frac{V_r}{2^n}$, where the $n$ represents the bit number and $V_r$ is the maximum voltage value. In this case, $V_r$ value is still between $0V \leq V_r \leq 5V$. Figure 4 presents a 3-bit DAC.

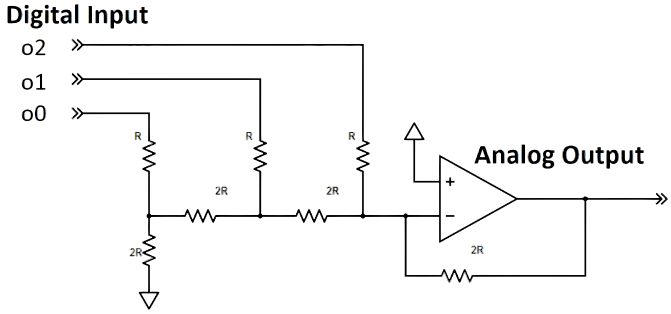


Fig. 4: 3-bit DAC circuit.

*2) ADC Flash:* An ADC Flash converts a voltage value into a digital signal. It is built using a voltage comparator and *XOR* logic gates for each defined value. The flash architecture is one of the fastest among ADC architectures, but is it also more complex in terms of construction. A 2-bit Flash ADC can be built from four **resistors**, three **comparators**, and three **xor** gates. Generalizing, in order to build a **N**-bit ADC flash circuit we need $2^n - 1$ **comparators** and **xor** gates, and $2^n$ **resistors** [4]. Figure 5 illustrates a 2-bit Flash ADC.

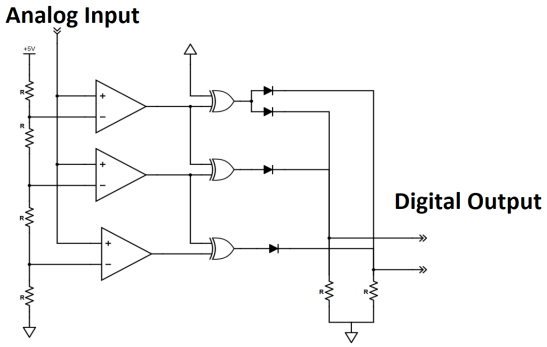


Fig. 5: 2-bit Flash ADC circuit.

## III. Related Work

In this section we present work related to analog verification and benchmark circuits. Harjani *et al.* [5] present a framework for analog synthesis and build common analog circuit blocks. Unfortunately this system is not publicly available.

Kaminska *et al.* [6] present a benchmark tool that allows engineers and researchers to perform analog simulations. The available benchmarks include analog circuit netlist described in HSPICE format and information about their type, description and behavior. This system is not publicly available and the tool does not allow the creation of circuits in arbitrary sizes.

Kondagunturi *et al.* [7] present an extension of [6]. The authors include a set of fault models to evaluate and validate different types of analog and mixed-signal. This tool is a good benchmark framework to test the efficiency of analog validation methodologies, but is not publicly available, and the benchmark tool is very limited in terms of size and type of circuits.

## IV. Methodology

In this section we describe tools used and created for simulating the circuits described in Section II. Moreover, we generate circuit netlists while describing more important parts in our implementation.

### A. ADC and DAC Netlist Generator

In order to represent circuits used in this work, we create a tool to generate ADC or DAC circuits or arbitrary sizes. This tool creates a netlist representation to be used by SPICE simulators and generates waveform stimuli using the Piecewise Linear *(PWL)* [8] format. The netlist is based in the SPICE

model and output file is a *(.cir)* file containing the circuit and respective behavior.

Algorithm 1 connects two input signals of comparator in the resistor output node. Algorithm 2 is responsible for connecting the generated outputs for each **xor** logic gate.

**Algorithm 1** Node Connect Algorithm

```
1:  procedure INSTANTIATES AND CONNECTS COMPARATORS
2:      i ← Bitss
3:      x ← pow(2, i) − 1
4:      Comp ← 0
5:      while Comp ≤ x do
6:          CompEst[comp].input1 ← resis[comp].output
7:          CompEst[comp].input2 ← AnalogInput
8:          CompEst[comp].output ← Emptynode
9:          Comp ← Comp + 1
10:         Comp ← EmptNode + 1
```

**Algorithm 2** Output Connection Algorithm

```
1:  procedure CONNECTS OUTPUT
2:      j ← 0
3:      i ← Bits
4:      x ← pow(2, i) − 1
5:      Xor ← 0
6:      Outputs ← Bits
7:      while Xor ≤ x do
8:          shift ← Xor
9:          while Shift > 0 do
10:             if Shift % 2 > 0 then
11:                 OutputEstr[j] ← XorEstr[Xor].output
12:             Shift ← Shift >> 1
13:             j ← j + 1
14:         j ← 0
15:         Xor ← Xor + 1
```

### B. SPICE Simulator

LTspice IV is a SPICE simulator, schematic capture and waveform viewer [9]. It is used to provide ALIAS input and compare the generated results on analog and digital environments. Figure 6 presents LTspice working diagram. LTSpice input is the analog circuit file and the output is a SPICE simulation result and metrics about the circuit (e.g., simulation time, circuit simplifications).

## V. RESULTS

In this section, we have used the tool described in Section IV to generate the ADC and DAC with different sizes. We have chosen the LTSPICE [9] to perform analog simulation and Icarus Verilog [10] to run digital simulation. The simulation has been conducted on an Intel Core TM i7-4770 @3.4GHz with 16GB of main memory, and results are presented in terms of time (seconds).

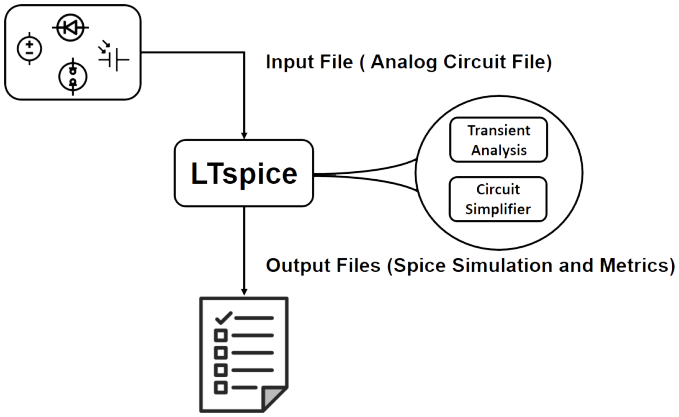


Fig. 6: LTSpice work flow.

TABLE I: DAC: Analog x Digital environment.

| Circuit Size | Time (s) | |
|---|---|---|
| | SPICE | Digital |
| 6 Bits | 1.109 | 2.257 |
| 8 Bits | 5.418 | 9.464 |
| 9 Bits | 13.456 | 9.580 |

### A. DAC 6, 8, 9 bits

We have generated DAC circuits of 6, 8, 9 bits. Table I presents simulation time using three DAC circuits with different number of bits using an increasing digital input ranging from 0 to $2^n - 1$ where $n$ is the number of bits. Figure 7 shows the elapsed time to simulate a DAC circuit using LTSpice and Icarus with the ALIAS abstraction. For this low complexity circuits the running times do not present considerable variations.

### B. ADC 5, 8, 9 bits

We have used our tool to generate ADC circuits of 5, 8, and 9 bits. The waveform generation is defined in the number of reachable states to each circuit, for example: a 9 bits ADC is able to reach the interval $0 \leq x \leq 512$. For each state we have the following equation: $V(n) = \sum_{i=0}^{i=n} \frac{\alpha}{\phi}$ where $V(n)$ is the voltage value, $n$ state number, $\alpha$ the maximum voltage, and $\phi$ the last state reached by ADC circuit. Using as example a 5 bits ADC, for reach state, we need a voltage variation of $0.01953125V$ for each state. Table II presents simulation time ADC using three ADC circuits with different number of bits using an increasing analog input ranging from $0V$ to $5V$. Figure 8 shows the time spent to simulate an ADC circuit using LTSpice and Icarus with the ALIAS abstraction. When we consider high complex analog circuits our methodology presents faster results.

TABLE II: ADC: Analog x Digital environment.

| Circuit Size | Time (s) | |
|---|---|---|
| | SPICE | Digital |
| 5 Bits | 9.514 | 9.338 |
| 8 Bits | 58,785.877 | 11.861 |
| 9 Bits | 807,921.520 | 175.046 |

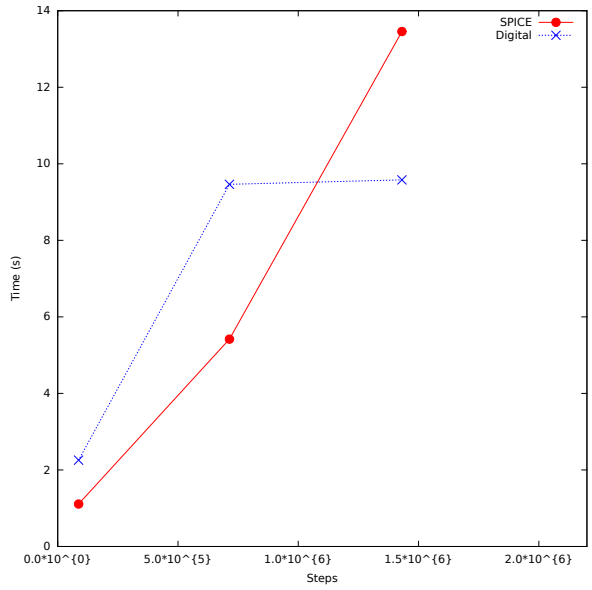


Fig. 7: DAC simulation time.

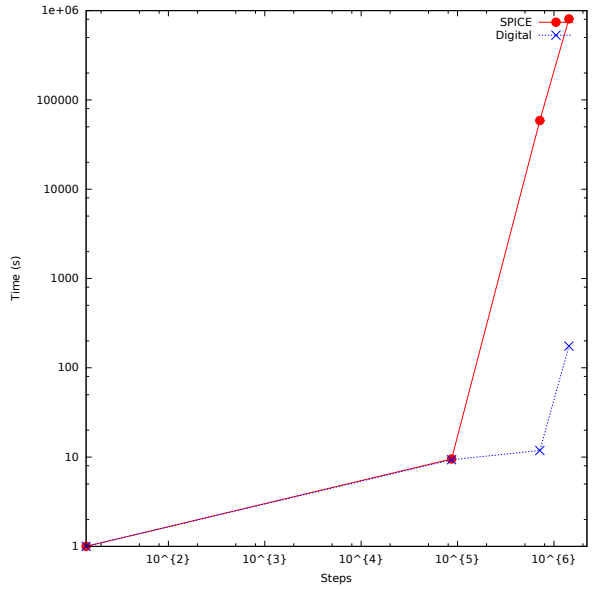


Fig. 8: ADC simulation time.

## VI. Conclusion and Future Work

Our results have demonstrated that digital abstractions of analog circuits can significantly decrease the simulation time during a mixed-signal verification process. Taking as example the ADC circuits, the time spent to simulate 9-bit ADC circuits is $1,300\%$ higher when compared to an 8-bit ADC, depicting an exponential growth when increasing the ADC resolution by a single bit. The study presented in this work shows the advantages of use ALIAS to validate mixed-signal circuits while supplying realistic analog inputs to digital blocks. As future work we will improve and include new features in our tool in order to provide more circuits to test mixed-signal verification environments.

## Acknowledgments

We would like to thank CAPES, CNPq, and FAPEMIG for the financial support.

## References

[1] H. D. Foster, "Trends in functional verification: A 2014 industry study," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2015.

[2] A. L. P. Marciano, "Alias: Analog circuit abstractions for digital systems verification," Master's thesis, Federal University of Minas Gerais, apr 2015.

[3] L. W. Nagel and D. O. Pederson, *SPICE: Simulation program with integrated circuit emphasis*. Electronics Research Laboratory, College of Engineering, University of California, 1973.

[4] M. F. Wagdy and Q. Xie, "Comparative adc performance evaluation using a new emulation model for flash adc architectures," in *Circuits and Systems, 1994., Proceedings of the 37th Midwest Symposium on*, vol. 2, pp. 1159–1163 vol.2, Aug 1994.

[5] R. Harjani, A. Rutenbar, and L. R. Carley, "Oasys: A framework for analog circuit synthesis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 8, no. 12, pp. 1247–1266, 1989.

[6] B. Kaminska, K. Arabi, I. Bell, P. Goteti, J. L. Huertas, B. Kim, A. Rueda, and M. Soma, "Analog and mixed-signal benchmark circuits-first release," in *Test Conference, 1997. Proceedings., International*, pp. 183–190, Nov 1997.

[7] R. Kondagunturi, E. Bradley, K. Maggard, and C. Stroud, "Benchmark circuits for analog and mixed-signal testing," Master's thesis, University of Kentucky, 1999.

[8] J. F. Hudson, "Piecewise linear topology," *New York*, 1969.

[9] I. LTspice and D. Models, "Linear technology," 2013.

[10] S. Williams, "Icarus verilog," 2006.