UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

BEHAVIORAL MODELING WITH SYSTEMVERILOG FOR MIXED-SIGNAL VALIDATION

Par

Ben Yochret Sabrine

Proposition de mémoire présentée au

Département d'informatique et d'ingénierie

Pour l'obtention de diplôme de

Maîtrise

en sciences et technologie de l'information

Jury d'évaluation

Président du jury:

Examinateur interne:

Directeur: Prof. Larbi Talbi, PhD

Résumé

Aujourd'hui, la nécessité d'intégrer des circuits analogiques et numériques ensemble sur la même puce est devenue une exigence fondamentale dans les conceptions les plus récentes. Jusqu'à présent, les simulations analogiques et numériques étaient deux mondes différents et aucune interaction n'existait entre eux. La plupart des problèmes pratiques étaient liés à cet écart. Ici, nous essayons de mettre les deux conceptions dans le même environnement et de les simuler à travers un outil numérique afin d'avoir plus de visibilité et une simulation plus rapide. L'analyse temporelle des modèles de circuits analogiques ne détecte pas tous les problèmes qui pourraient apparaître lors de la validation ou après la fabrication et qui pourraient diminuer / modifier les performances du système. De plus, le temps de simulation de bas niveau pour les circuits analogiques est si important.

Notre contribution est tout d'abord de concevoir un modèle comportemental avec SystemVerilog qui fonctionne simultanément dans les domaines temporel et fréquentiel. Deuxièmement, nous concevrons un environnement de vérification / bancs d'essai Verilog qui fonctionnera simultanément dans les domaines temporel et fréquentiel. Notre solution consiste donc à mettre les deux conceptions dans le même environnement et à les simuler à travers un outil numérique afin d'avoir plus de visibilité et un fonctionnement plus rapide. Ce travail est réalisé en collaboration avec CIENA à OTTAWA.

Abstract

Todays, the need of integrating analog and digital circuits together on the same chip has become a fundamental requirement in most recent designs. So far, analog and digital simulations were two different worlds and no interaction existed amongst them. Most of the practical issues were related to this gap. We, herein, try to put both designs in the same environment and simulate it through a digital tool in order to have more visibility and faster running. The temporal analysis of analog circuits models does not detect all the issues that might appear during the validation or after the manufacturing and which might decrease/change the system performances. Moreover, the low-level simulation time for analog circuits is so important.

Our contribution is firstly to design a system Verilog behavioral model that runs simultaneously under both time and frequency domains. Secondly, we will design a system Verilog verification environment/testbenches that run simultaneously under both time and frequency domains. So, our solution is to put both designs in the same environment and simulate it through a digital tool in order to have more visibility and faster running. This work is realized at Ciena Corporation, Ottawa.

Acknowledgements

William Arthur Ward once said: “Feeling gratitude and not expressing it is like wrapping a present and not giving it.” I am grateful for this opportunity to give thanks to the people who have helped to bring this dissertation to fruition.

I would like to take this opportunity to thank my supervisor at UQO, Doctor Larbi Talbi. My manager at Ciena, Doctor Naim Ben-Hamida for giving me this opportunity to develop my theoretical skills within an industrial environment, Doctor Tahar Haddad and Doctor Sadok Aouini for their help and support, and finally thank all the people who had helped me to achieve my master.

Dedication

To my wonderful parents: Thank you for always being there, showering me with love, encouragement and affirmation. You have done the impossible by giving me roots as well as wings.

To my husband Hamza, for the patience and support he has shown throughout this master's degree.

To my daughter Sophia, that I love so much.

Contents

Table of figures

List of Tables

Acronyms

Application Specific Integrated Circuits	ASIC
Analog/Digital Converter	ADC
Analog Mixed Signal	AMS
Differential-algebraic equations	DAE
Digital/Analog Converter	DAC
Delta Sigma Modulator	DSM
Error Detector	ED
Frequency Divider	FD
Hardware Description Language	HDL
Kirchhoff Current Law	KCL
Kirchhoff Voltage Law	KVL
Loop Filter	LF
Numerical integration	NI
Phase Locked Loop	PLL
System on Chip	SoC
Switched Capacitor	SC
Synopsys Digital Simulator	VCS
Voltage Controlled Oscillator	VCO

Chapter 1

Introduction, Context and Problem statement

1. Introduction

Today, design verification of VLSI circuit is very important due to the high costs in of the manufacturing process. Therefore, the simulation remains the tool most used to validate this realization. In addition, the number of transistors per circuit has also undergone a significant increase: this number has increased from a few thousand to tens of millions in less than a decade. This increase makes the task of the simulators more and more difficult. Indeed, a simulator who can simulate ten thousand transistors in acceptable time, i.e. in less than one day, will not be able to simulate ten times more, because in this case the simulation time will take several days. Due to this problem, the increase in the speed of calculation of the machines is not enough, and that's why the algorithms of the simulators must be constantly improved.

Faced with these problems, the simulation in mixed digital analog mode is a good solution. Indeed, this solution allows to combine the precision of the analog simulation and the speed of the digital simulation. Of course, this is a compromise, because the designer does not have the same degree of information on the whole circuit, that's why that it is necessary to correctly decompose the circuit and to isolate the parts which need a fine simulation.

Therefore, the first utility of the simulation in mixed mode is that allows to simulate a VLSI circuit in its entirety (or almost), and to validate its behavior. Thus, modern applications of electronics such as telecommunications, high-definition television or signal processing, are increasingly turned to the circuits, part of which is composed of analog functions and the other of digital functions.

The main reasons for this orientation are, on the one hand, the analogue realization requires less transistors and therefore less space on silicon than its numerical equivalent, and on the other hand that analog electronics has made big progress in recent years , and thus the difference in quality between some analog functions and their numerical equivalent has become negligible for many applications.

Consequently, for these circuits the mixed simulation is necessary to validate the good functioning of the whole circuit and especially at the level of interaction between the two levels. It is for these two reasons that simulation in mixed mode knows such a trend currently [1].

2. Problem Statement

So far, analog and digital simulations were two different worlds and no interaction existed amongst them. Most of the practical issues were related to this gap.

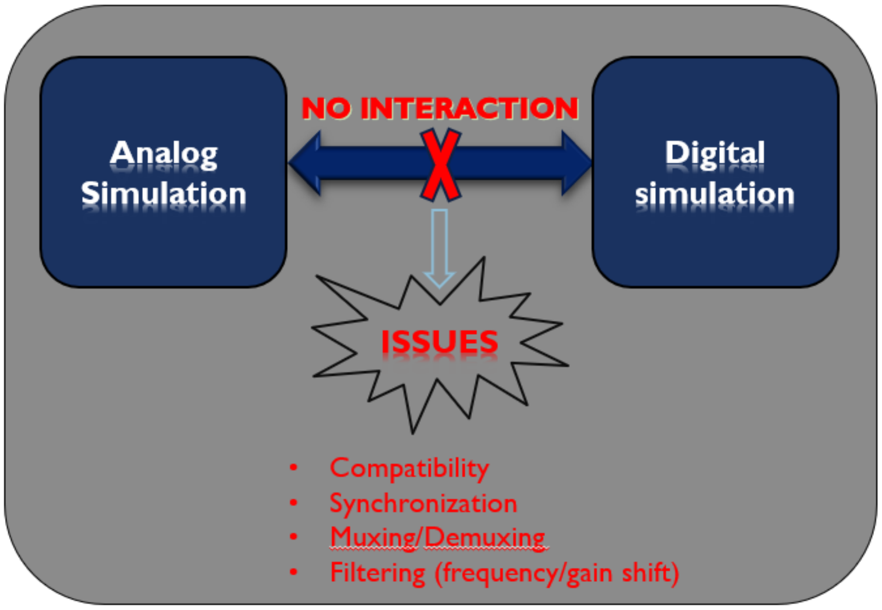


Figure 1: Problem statement

In fact, digital validation is interested to evaluate the logical function of a circuit. Consequently, it is important to test all the possible states that the design can go through to avoid any error that could be detected later. Digital synthesis plots the behavior model over simple digital gates combined with the constrained topology. Generally, digital blocks are coded using a hardware description language (HDL) like VHDL or Verilog and simulated in a digital simulator such as VCS or ModelSim [2].

Analog validation is interested to evaluate the characteristics of the output of the analog circuit according to a set of required performances. The validation of analog circuits is generally carried out with the SPICE tool which uses compact device models to evaluate the behavior of the circuit schematic [2].

The analog synthesis is much more difficult than digital synthesis due to the unconstrained topology and the complexity of the fundamental building blocks of analog circuits. The main reason of this problem is that analog designers are enabled to describe their design with a relatively high-level language which synthesize an implementation that is guaranteed to be functionally correct and have near optimal performance. Therefore, they must transform manually their design from concept to implementation and so the design process is much slower and more error prone than the design process of digitals circuits. We can say that the traditional method used for analog design allows for more creativity because they try to convert specifications to circuits, but it results in more errors, especially those resulting from miscommunication. These miscommunications errors prevent the whole system to run correctly when all the blocks are assembled, even though the blocks work properly when tested individually [3].

In such situation, when a complex digital chip can be designed correctly on the first try in few months while designing a complex analog chip can require 3-4 responses and more than one year to get right, we must find a compromise to overcome this kind of problem.

3. Goals and research Methodology

The main objective of our research is to design a system Verilog behavioral model that runs simultaneously under both time and frequency domains with a purely digital tool.
The fundamental challenges of our design are:

- Having both analog and digital parts of the design being simulated together.
- Monitor the interaction between both parts and detect any possible conflict.
- Rapidly validate all frequency related outputs within frequency plots.
- Reduce the simulation time.

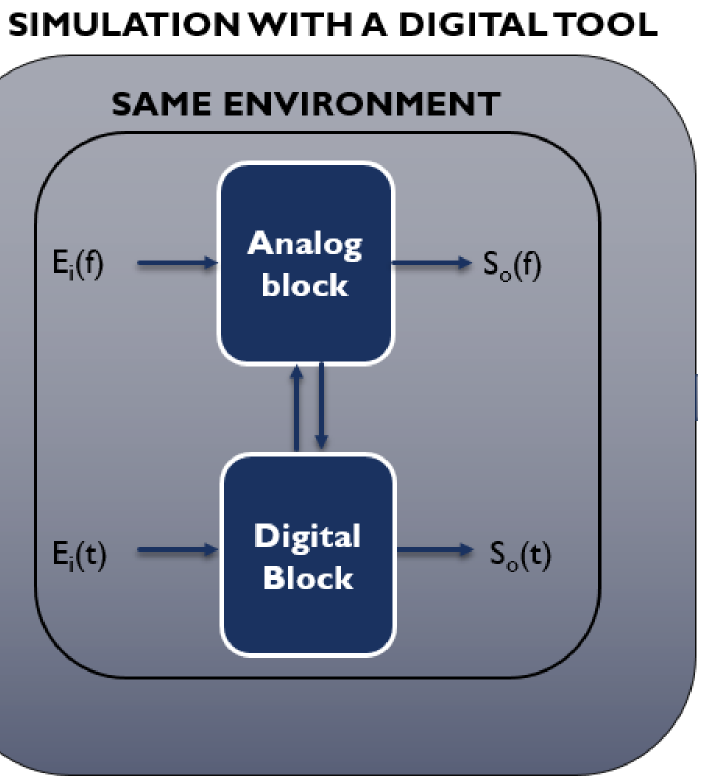


Figure 2: Proposed solution

Based on these challenges, this document is organized as follows:

Chapter 2 is a general overview of the different simulation levels for analog and digital circuits.

Chapter 3 is a review of the different approaches used for validating mixed systems (advantages and disadvantages) as well as the different simulators used.

Chapter 4 presents our proposed approach for developing behavioral models for analog circuits and mixed signal validation.

Chapter 5 presents a mixed system on which we will develop the behavioral model of a mixed system and validate it through the results obtained.

4. Contributions

The contribution of this research consists, on the one hand, to design a System Verilog behavioral model that operates simultaneously under time and frequency domains with a purely digital tool (VCS).

On the other hand, we will design a system Verilog verification environment/testbenches in order to detect the issues generated from interaction between analog and digital blocks before manufacturing and which might decrease/change the system performances.

5. Summary

After introducing the research context and presenting the problem statement in the first chapter, we will present, in the following chapter, an overview of different levels of simulation for digital and analog circuits.

Chapter 2

Overview of different levels of simulation

1. Introduction

Today, the need of integrating analog/digital blocks on the same chip has created a new challenge in modeling and simulation.

First, due to different approaches used in modeling and simulation to analyze analog and digital circuits, this causes problems in the simulation of the entire system. So, it is better to combine all the approaches in order to simulate the whole system with the appropriate speed/accuracy compromise.

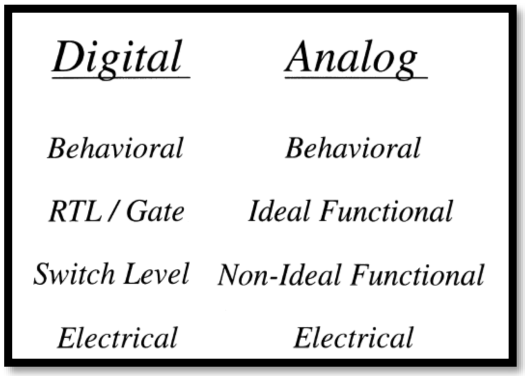
Secondly, the focus is on system-level design to deal with the complexity of large designs. In this context, the behavioral modeling and simulation are essential to the validation of an architecture proposed before the beginning of a detailed conception.

For many years, this approach has been very successful in the design of digital circuits by using standard hardware description languages to describe and simulate the behavioral modeling of the designs.

However, analog circuits are still designed and verified at the electrical level even though the simulation time is very important.

For mixed-signal simulation, it is important to create an analog modeling and simulation environment similar to the digital domain which allow to the designers to model components at the behavioral level and then perform system-level analog simulation.

With this approach, we will be able to validate at a high level the entire system's architecture in a reasonable amount of time. Therefore, based on specifications derived from the high-level simulation, the detailed design could be performed [4]. The following figure lists all the simulation's levels in the hierarchy for the two domains.



Digital	Analog
Behavioral	Behavioral
RTL / Gate	Ideal Functional
Switch Level	Non-Ideal Functional
Electrical	Electrical

Figure 3. Levels of simulation [4]

2. Simulation's levels for digital circuits

In this section we will describe the most detailed level to the higher level of abstraction.

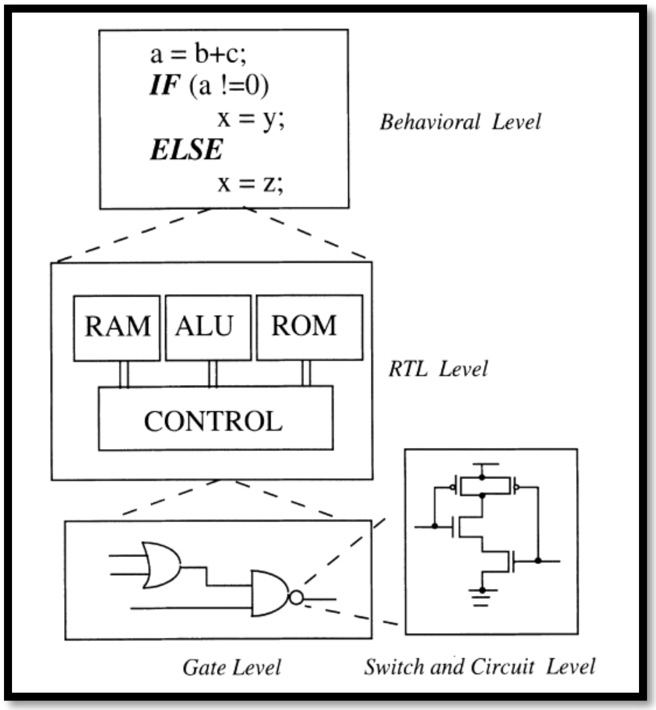


Figure 4. Levels of Abstraction in Digital Simulation [4]

2.1.Electrical Simulation

Electrical simulation or circuit level simulation provides huge details about the circuit. To model the dynamic characteristics of a circuit subjected to a set of input voltages, it is important to use ordinary nonlinear first order differential equations whose solutions are waveforms of voltage across pairs of circuit nodes and current waveforms through circuit elements [5].

The solution of this kind of nonlinear equation uses the Newton-Raphson method to convert them into linear equations. The disadvantage of this approach is that, for large circuits, the execution time of the linear solution is very high [6].

Several studies have been carried out to improve the performance of circuit simulators, note for example, timing simulation [7] [8] [9] which is a simplified form of circuit simulation based on relaxation, and tearing methods, which have been applied to linear [10] [11] [12] and non-linear equations [13].

2.2.Gate-Level Simulation

We use this kind of simulation when the electrical analysis is no longer cost effective due to complexity of the circuit. In logic simulation, transistors are grouped in logic gates and modeled at the gate level [14]. This form of simplification (macro-modeling) improve running time due to small numbers of models to be processed and the simplicity of arithmetic operations required to run each transistor group.

Therefore, instead of treating voltages and currents at signals nodes, we define discrete logic states and simple Boolean operations in order to determine the new logic value of each node.

2.3.Switch-Level Simulation

In this level of simulation, the whole system is simulated at transistor level instead of gate level [15] [16]. The transistors are modeled as gate-controlled switches and operate as follows: if the transistor is "ON," it is viewed as a closed switch and it may transfer a signal value from one node to another; if the transistor is "OFF," it is viewed as an open switch and is incapable of transmitting any signals through it.

The network is composed of a set of nodes connected by these switches, and the logic value at each node is determined using this idealized transistor model.

2.4.Register-Transfer Level Simulation

Register-Transfer Level (RTL) [17] simulation describes logic circuits at a higher level of abstraction. This form of simulation used for data path design, it’s used for description and simulation of the designs. The instruction describing the circuit operation includes a sequence of register transfers and arithmetic operations that are similar to data-flow descriptions.

Although RTL simulators are popular in computer design, but they do not usually provide information regarding races, hazards, illegal states or critical timing constraints.

2.5.Behavioral Level Simulation

Behavioral Level Simulation [18] [19] [20] allow the designers to define combinational and sequential blocks that can be used in system-level simulation. Structural blocks describe the interconnection between functional blocks. A behavioral block contains a detailed description of the operations to be performed on the inputs to produce the outputs of the block. The instructions describing the operations are usually written in a high-level language, typically a hardware description language (VHDL [21], system Verilog [22]), and then compiled into the simulator (ModelSim, VCS).

3. LEVELS OF SIMULATION FOR ANALOG CIRCUITS

As in the case of digital circuits, there are also levels of simulations for analog circuits. The different levels in the analog hierarchy are described in the following sections using the sampled data filter of figure 3.

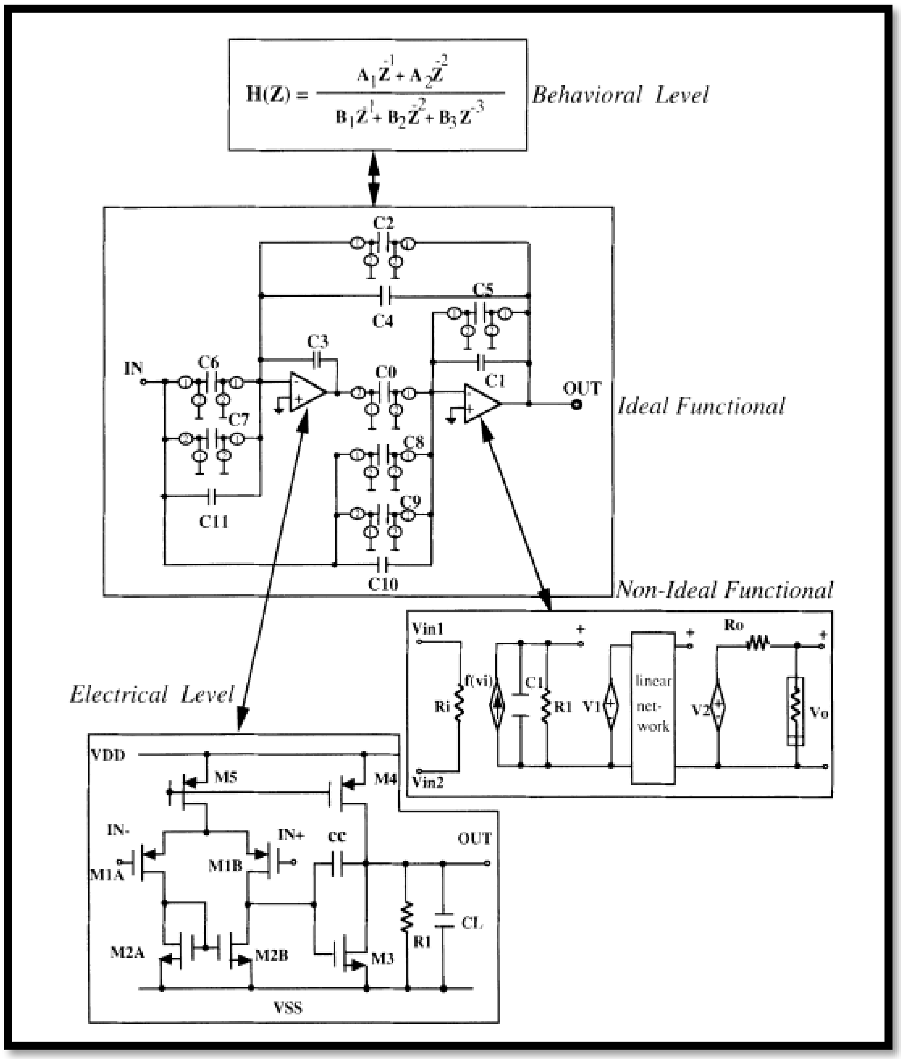


Figure 5. Levels of Abstraction in Analog Simulation [4]

3.1.Behavioral Simulation

When the function of a block is known but its detailed structure is undefined, we use the behavioral simulation [23][24][25][26][27]. At this level, individual blocks are described in terms of s-domain transfer functions, z-domain transfer functions, differential equations. For example, in the top of the figure 3 a behavioral level z-domain transfer function for a discrete-time filter. Multipliers, differentiators and integrators can be used to describe the interaction between behavioral blocks in terms of signal-flow diagrams.

3.2.Ideal Functional Simulation

Ideal Functional Simulation corresponds to the RTL level in digital circuits. This level allows the designer to quickly validate a proposed architecture using standard components before the details of the design are considered. At this level, designers use circuit components, such as ideal opamps, switches, integrators, and comparators [4].

3.3.Non-Ideal Functional Simulation

This level corresponds to the gate and switch levels in the digital domain that includes timing information and other electrical effects. At this level, we can find Complex models that include nonlinear properties, dynamic behavior, limiting effects and detailed input/output characteristics. It is similar to the ideal functional level except that the first order and second-order details are included in the models [4].

For example, in Figure 3, the macro model of the operational amplifiers includes a piecewise-linear gain, finite bandwidth and input/output resistance. The capacitances and resistances could also be included in the MOS switches when simulating the switched-capacitor filter circuit.

3.4.Electrical Simulation

This level corresponds directly to the same level in the digital domain. It is the most detailed level. the operational amplifiers in figure 3 are represented in terms of their MOS transistors with detailed models. In addition, frequency-domain analysis such as AC analysis, sensitivity, noise and distortion must be available to the analog designers.

4. Summary

In this chapter, we have presented the different levels of simulation for digital and analog circuits in order to know the advantages and disadvantages of each level of simulation and identify the importance of the high-level validation. The following chapter will be a review of three approaches that address mixed signal validation as well as the tools used and the challenges.

Chapter 3

Mixed-Signal Validation; OVERVIEW AND CHALLENGES

1. Introduction

System-level validation of mixed-signal SoC is challenging [3]. To better understand the need of mixed signal validation, we will discuss at the beginning of this chapter traditional analog and digital simulators and then we will compare three different approaches to validate mixed signals and finally we will show that behavioral modeling is the best solution for mixed-signal SoC validation.

2. Analog simulators

For analog circuit simulation, engineers use SPICE to verify their designs. It gives them the ability to compute the voltage and current values at each node for all instants of time. Kirchhoff's current law (KCL) and Kirchhoff's voltage law (KVL) equations are generated first for each node of the design. The presence of devices such as transistors in the design will set differential-algebraic equations (DAE's) as described below [28]:

$$\frac{dq(V,t)}{dx} = i(V,U,t) \tag{1}$$

Where V= V(t) is a vector of nodal voltages depending on time, U = U(t) is a vector of inputs, q(v,t) is a variable describing charge, and I(v,u,t) is a function describing the current. These kinds of equations can be solved by invoking NI (numerical integration methods) [2].

Second order Gear, trapezoidal, and/or BE (Backward Euler) are algorithms used first to apply time-discretization. An example where backward Euler integration is used is described: equation (2) is an algebraic transformation that yields to the system (1):

$$V_{t+\Delta t} = W_t + \Delta t.i(v_{t+\Delta t}, U_{t+\Delta t}, t+\Delta t) \tag{2}$$

Where Δt is the step time, $\boldsymbol{Vt}+\Delta\boldsymbol{t}$ is the vector of node voltages that needs to be solved at time t+Δt. In other words, finding the zeros of this nonlinear operator:

$$F(V)=0 \tag{3}$$

Where F is the equal to (2). Successive chords method or Newton-Raphson; are iterative algorithms that can be used to solve (3). Here is the gotten result if Newton-Raphson is used:

$$J_F(V^{i+1}) - J_F(V^i) = -F(V^i) \quad (4)$$

Where i is the iteration number and JF is the Jacobian matrix for operator F.
LU factorization is a useful method to solve equation (4). Some predefined values (noted by pv) are fixed as the difference between Vi and Vi+1and when the difference between them reach pv the loop breaks and stops. At every time step, all these informations are carried out during a transient simulation. SPICE simulators are known for their high accuracy to match designers' expectations which creates very detailed and high accurate models. This, in turn, means an expensive simulation [28].

3. Digitals simulators

After circuit validation at the transistor level, digitals simulators will evaluate the logical function of the circuit. A hardware description language (HDL) like VHDL or Verilog can be used to describe the design and then simulated under digitals simulator like VCS or ModelSim.

Digitals simulators are discrete time because they track any changes in Boolean values, hence, they evaluate the functional models when inputs change. Triggering events is an internal time-wheel mechanism in simulators that keeps track of when the input changes occur. When such an event occurs, the simulator starts computing the new outputs. There is another technique called selective trace technique which ignore in the simulation the periods without change in Boolean values until the next triggering event [29].

Figure 4. introduces the basic difference between both analog and digital simulator.

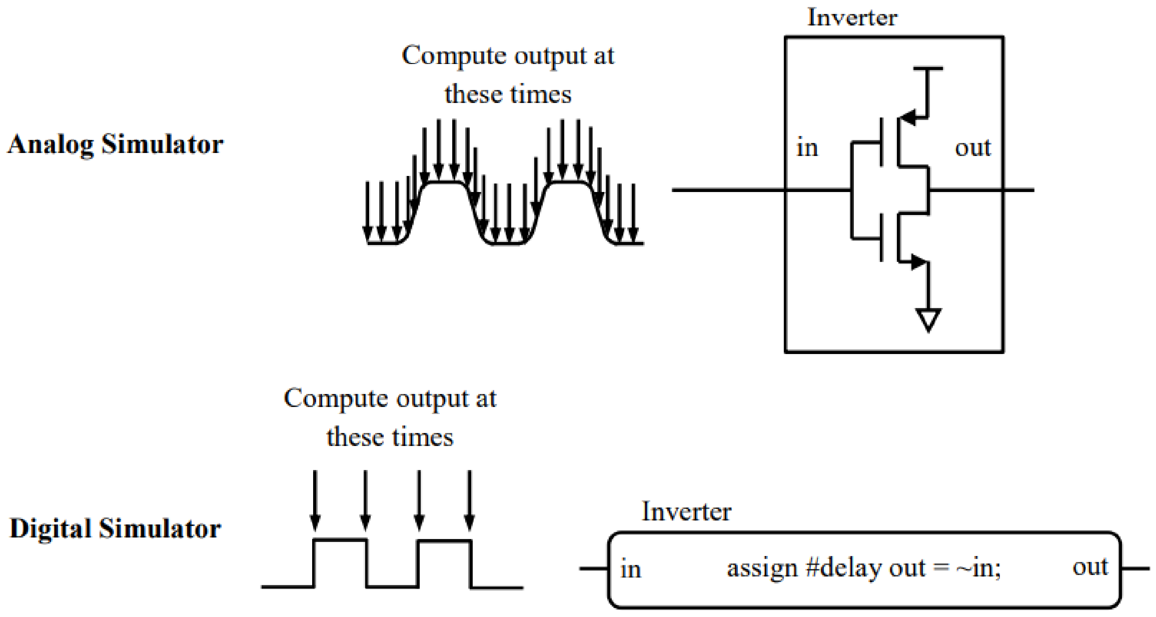


Figure 6. Difference between analog and digital simulators processing [2]

For the analog simulator (SPICE), it performs a very accurate analysis of the output waveform at each time by executing calculations. For the digital simulator, it calculates only the outputs of the inverter when the input changes its Boolean value. Thus, the analog simulator is much more demanding in terms of time and computation.

The choice of analog or digital simulators depends on the level of abstraction. If designers work at transistor or circuit levels, then they need analog simulators as input and output signals are continuous and performances metrics are computed based on signal interpolation.

Otherwise, if designers are working at gate or RTL level then digital simulators are best suitable as signals are presented by discrete states and number of inputs is large.

Even through, theoretically, it is possible to use analog simulators at any levels, in practice, it is not feasible as analog simulations are heavily CPU and memory usage because simulations rely on numerical solvers of nonlinear differential equations. Table 1 summarizes differences between both types of simulators.

Table 1. Analog Simulators versus Digital Simulators

Analog Simulators	**Digital Simulators**
Accurate metrics	Functional, logic coverage
Heavy Simulations	Fast Simulations
Continuous inputs and outputs	Input and output discrete states
Bidirectional	Unidirectional

4. Approaches to mixed-signal validation

In this section, we will compare three approaches for mixed-signal validation:

- ***Modified Simulators*** to run fast analog simulations.
- ***Macro-modeling*** to reduce model complexity in order speed up analog simulations.
- ***Behavioral modeling*** to run digital simulations with behavioral models of analog blocks.

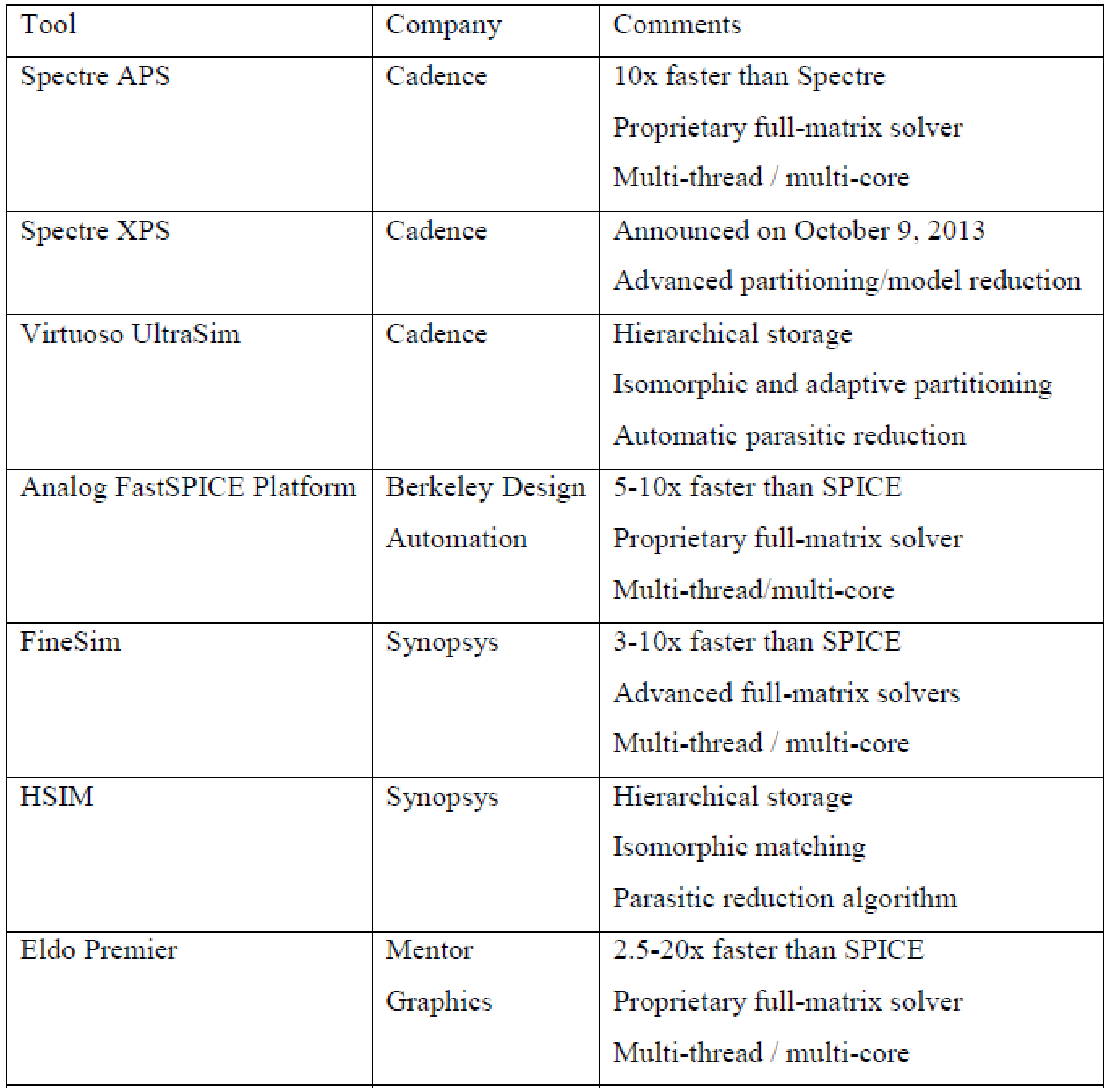
4.1. Modified simulators

We have shown in the two previous sections that the analog and digital simulators are two completely different worlds which makes mixed simulation very difficult to perform. In following section, we are going to go through some research on modified simulators which aim to adapt mixed-signal simulations.

4.1.1. Fast SPICE simulator

SPICE simulator uses a high accuracy mathematical engine to describe a model of an integrated circuit. This high accuracy leads to high complexity and large matrix size while solving the mathematical equations highlighted in the previous section. Therefore, all this results in a simulator with a complexity of O (n1.5-2) where n is the number of nodes in the circuit and a limited capacity of about 100,000 active elements (MOSFETs) [28]. With the high number of MOSFET devices (100 million to 1 billion) and parasitic devices, traditional SPICE is unable to perform a functional simulation of the full chip [28]. For this reason, fast SPICE intervenes with his three methodologies to speed simulation: Matrix-based, Graph-based and Circuit-based. The following table represents some extensions to existing analog simulators to support digital blocks and run simulations at transistor or/and gate levels.

Table 2. Modified Analog Simulators to support digital designs [30]



Tool	Company	Comments
Spectre APS	Cadence	10x faster than Spectre Proprietary full-matrix solver Multi-thread / multi-core
Spectre XPS	Cadence	Announced on October 9, 2013 Advanced partitioning/model reduction
Virtuoso UltraSim	Cadence	Hierarchical storage Isomorphic and adaptive partitioning Automatic parasitic reduction
Analog FastSPICE Platform	Berkeley Design Automation	5-10x faster than SPICE Proprietary full-matrix solver Multi-thread/multi-core
FineSim	Synopsys	3-10x faster than SPICE Advanced full-matrix solvers Multi-thread / multi-core
HSIM	Synopsys	Hierarchical storage Isomorphic matching Parasitic reduction algorithm
Eldo Premier	Mentor Graphics	2.5-20x faster than SPICE Proprietary full-matrix solver Multi-thread / multi-core

4.1.2. Piecewise-Linear Simulation

The Piecewise-Linear Simulation run the entire system rather than the full-blown differential algebraic system. With this form of simulator, we can run the macro-models of sub-circuits like operational amplifiers can be simulated instead of the full transistor level model [2]. As for digital circuits which are simulated in in their Boolean abstraction, analog circuits can be also simulated in a high-level piecewise linear abstraction [2].

4.1.3. Combining analog and digital simulator

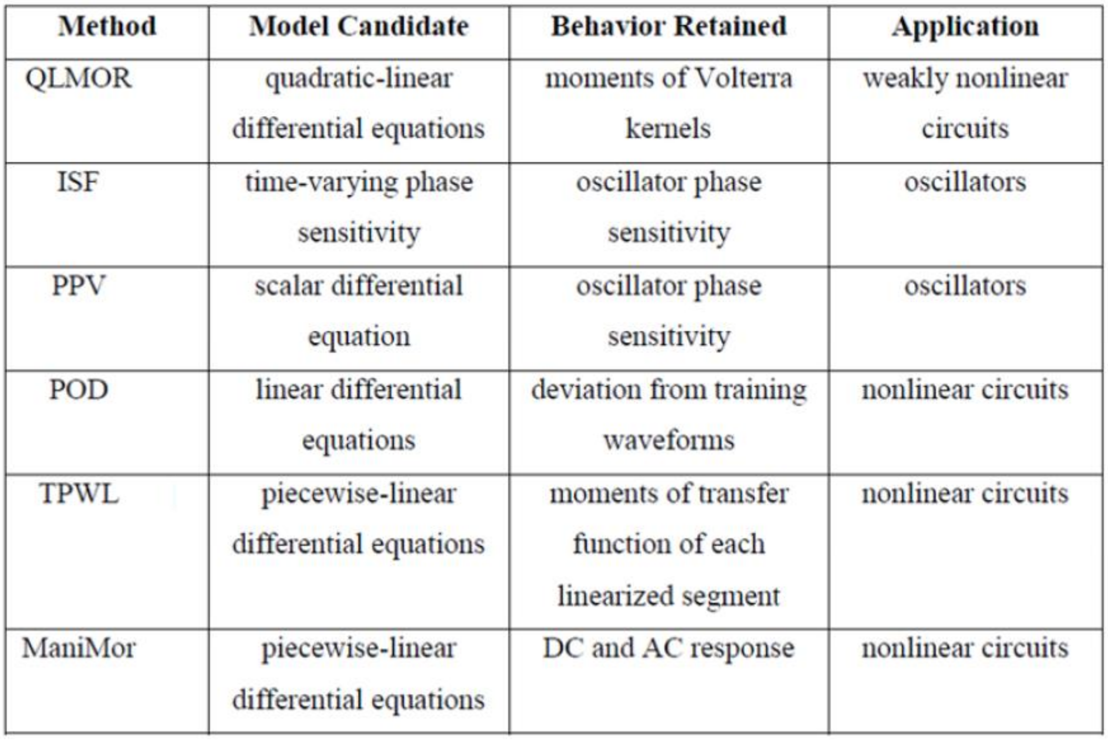
From the previous sections, fast SPICE is more accurate and efficient compared to basic SPICE due to his specified technologies that point to the needed circuit and its great ability to process complex circuits [28]. But it's still difficult to simulate both analog and digital circuits using an analog simulator (SPICE or fast SPICE).

Due to this problem some research has proposed to combine analog and digital simulators in order to allow each type of circuit to be analyzed by their respective engines. Authors in [31] and [32] have implemented an interface than stitch both simulators (analog and digital) with the aim to benefit from the best of two worlds: analog circuits will be solved accurately by the analog simulation engine, and at the same time, the digital simulation will be maintained at its usual high speed [33]. However, time of analog simulation slows down the top-level simulation of the whole system [34]. Therefore, another line of attack is needed.

4.2. Macro-modeling

Due to high simulation time of analog circuits, macro-modeling called also model reduction problem is used to find an alternative model (linear or nonlinear) possibly simpler in order to reduce simulation's time but maintaining the full system behavior. For example, we can use a transfer function of transistors rather than using the SPICE model. Table 3 provides a list of macro-models with filed application.

Table 3. Nonlinear macro-modeling algorithms [30]



Method	Model Candidate	Behavior Retained	Application
QLMOR	quadratic-linear differential equations	moments of Volterra kernels	weakly nonlinear circuits
ISF	time-varying phase sensitivity	oscillator phase sensitivity	oscillators
PPV	scalar differential equation	oscillator phase sensitivity	oscillators
POD	linear differential equations	deviation from training waveforms	nonlinear circuits
TPWL	piecewise-linear differential equations	moments of transfer function of each linearized segment	nonlinear circuits
ManiMor	piecewise-linear differential equations	DC and AC response	nonlinear circuits

4.3. Behavioral modeling

Previously viewed methods aim to keep the characteristics of the circuit in the description at the transistor level [35]. The functional validation of the entire system is not very demanding in terms of precision as in terms of simulation time [28], for this reason behavioral modeling is the best solution.

Behavioral models are written in a high-level language and require expert knowledge [35]. This approach is very important because it allow to validate the whole mixed-signal SoC and it can include checking in the behavioral models to facilitate the validation effort. It is primordial to note that this research is at system level, as illustrated in figure 7.

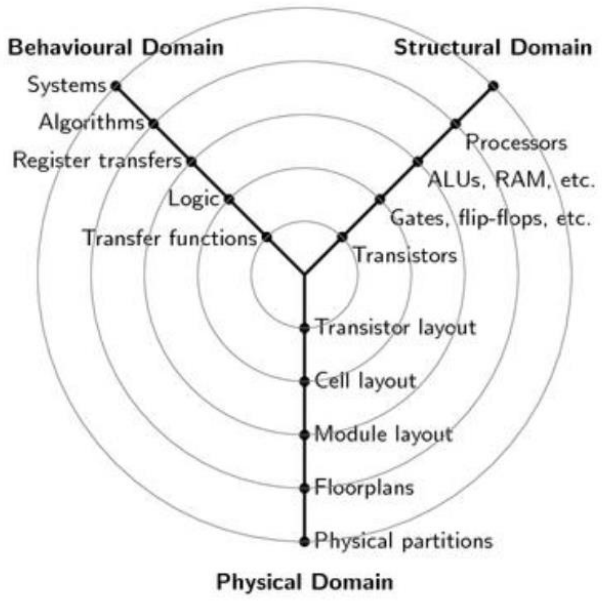


Figure 7. Gajski Kuhn Y-Chart

The industry response to this approach is many high-level languages such as Simulink/Matlab, SystemC-AMS, Verilog-A/Verilog-AMS/VHDL-AMS.

4.3.1. Simulink/Matlab

Simulink/MATLAB is a very used tool for system-level functional simulation and verification [36]. It is popular with its large library for derivation, transfer functions, filters, discrete time quantizes, mathematical operations and scope captures and visualization which make designers' task easier, indeed, they have to select the appropriate blocks, link them together, setting the simulation parameters and run [36].

A disadvantage of Simulink/MATLAB is that models can be only simulated under Matlab's simulator and it's not easy to connect it to the digital system written in Verilog.

4.3.2. SystemC-AMS

It is a C++ based language that extends SystemC (a digital circuits modeling language). We can find three Modes of Computation (MoCs):

- Timed Data Flow model (TDF): which allow to model data as time-sampled signals (discrete-time) [37].

- Linear Signal Flow model (LSF): it's similar to Simulink, it allows the modeling of continuous time systems as a set of linear algebraic equations [37].

- Electrical Linear Networks model (ELN): it allows the instantiating of the rest of the design. We can find current/voltage sources, resistors, capacitors, inductors, transmission lines, transformers, ideal switches, etc [37].

Unfortunately, as we have already seen in the previous section with Simulink/Matlab, SystemC-AMS cannot be easily connected to the digital system because has its own simulator.

Even though MATLAB/Simulink and SystemC-AMS are able of modeling continuous and discrete system, they lack the advantage of integration with digital system.

4.3.3. Verilog-A/Verilog-AMS/VHDL-AMS

Verilog-AMS/VHDL-AMS overcome the integration problem. They are able to handle both digital and continuous time analog signals and they can make the connection to a digital system relatively easy [38]. For example: delta-sigma modulators [39], gain amplifier [40], PLL's [38], …etc.

Verilog-A is an improved version of Verilog-HDL that takes care of analog functions (Z and Laplace transformations, integrations, delays…) [40] and it behaves the same way as SPICE simulators behave for the continuous system.

All the languages mentioned above use SPICE as simulator which execute orderly all the differential equations which slows down the simulation time.

4.3.4. Digital Verilog with Real

Basic algebraic capability and wires can be linked together using "Real" in Verilog simulators, and this have been implemented for the past few years in HDL-Languages (VHDL/Verilog/SystemVerilog) [41].

Sampled data systems integrate easily into the simulation environment because digital Verilog work with Boolean values and discrete events [42]. The following figure shows a sampled signal which is updated periodically.

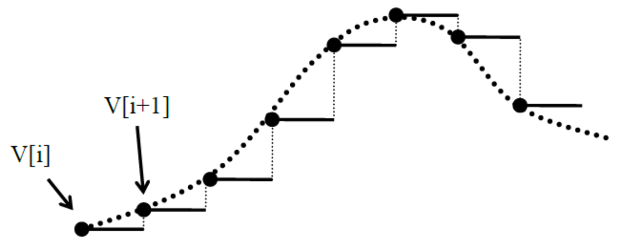


Figure 8. Sampled Data representation of continuous time signal [2]

Sampled data can be used to model some analog systems such as SC (Switched capacitor) and DSM (Delta Sigma Modulators) without complexity.

DSM first order is an example shown by Figure 9:

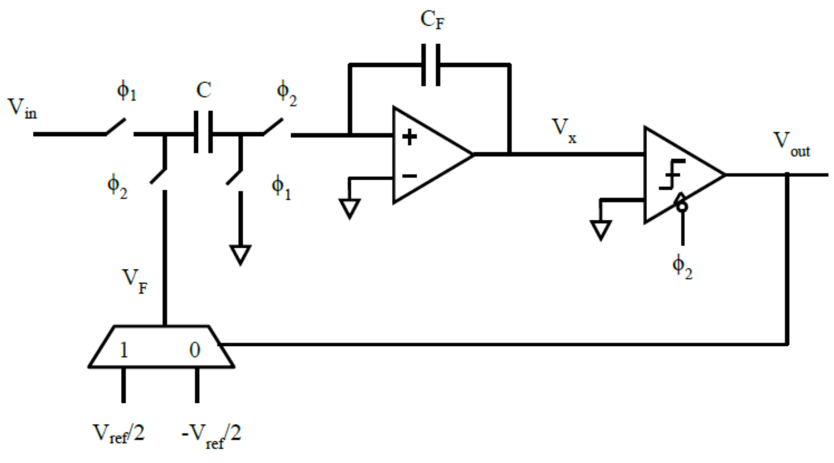


Figure 9. Switched capacitor implementation of 1er Order DSM [2]

Digital system is designed to sample directly the DSM's output and clock edge triggers all the nodes to settle which means a "synchronous system".

Figure 10 presents the corresponding signal flow graph for DSM 1st order; where Cf = C

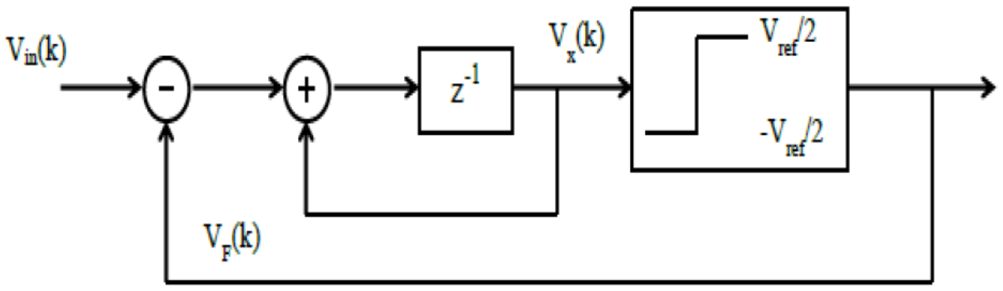


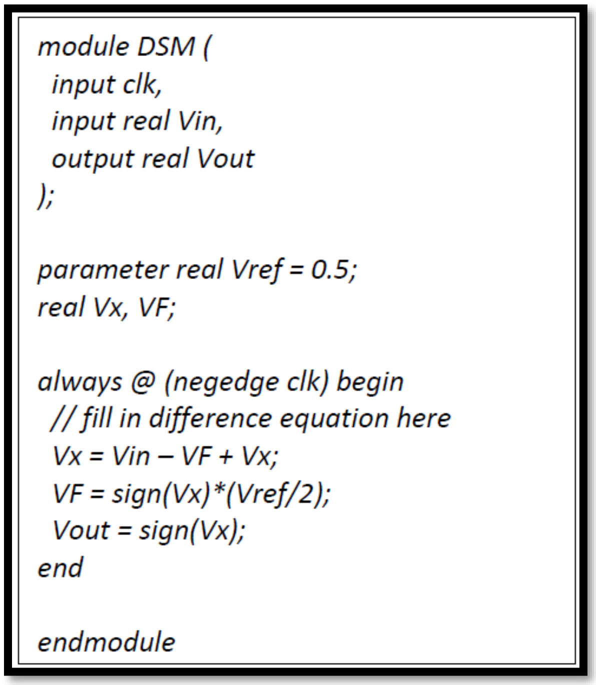
Figure 10. signal flow graph for 1er order DSM [2]

From the previous figures; system equation can be extracted as below [2]:

$$V_x(k+1) = V_{in}(k) - V_F(k) + V_x(k) \quad (5)$$

$$Where\ V_F(k) = \begin{cases} \frac{Vref}{2} . V_x(k-1) > 0 \\ -\frac{Vref}{2} . V_x(k-1) < 0 \end{cases}$$

For DSM example, system Verilog can be used to model it as described below [2]:



```
module DSM (
  input clk,
  input real Vin,
  output real Vout
);

parameter real Vref = 0.5;
real Vx, VF;

always @ (negedge clk) begin
  // fill in difference equation here
  Vx = Vin – VF + Vx;
  VF = sign(Vx)*(Vref/2);
  Vout = sign(Vx);
end

endmodule
```

Listing 1. Example model of first order delta-sigma modulator [2]

Delta sigma modulators and switched capacitor filters can be modulated either with MATLAB or with digital VERILOG languages with extensions of real numbers.

However, there are other transformation techniques that allow conversion of analog continuous-time systems to discrete-time systems. Note the bilinear transform [42], which converts analog transfer functions into difference equations in discrete time using a sufficiently high sampling rate (according to the Nyquist criterion). Several circuits have been modeled using Euler's advanced integration technique, note for example, digital PLLs [43], phase interpolators [44], switched capacitor amplifiers [41].

5. Mixed-signal Validation Challenge based on behavioral model

Among the approaches seen previously, verification method based on behavioral models simulated in digital simulators is the most common approach adopted in industry for the speed of simulation for mixed signal validation.

Our contribution is firstly to design a system Verilog behavioral model that runs simultaneously under both time and frequency domains. Secondly, we will design a system Verilog verification environment/testbenches that run simultaneously under both time and frequency domains. So, our solution is to put both designs in the same environment and simulate it through a digital tool in order to have more visibility and faster running.

The fundamental challenges of our behavioral design are:

- Having both analog and digital parts of the design being simulated together.
- Monitor the interaction between both parts and detect any possible conflict.
- Rapidly validate all frequency related outputs within frequency plots.
- Reduce the simulation time.

6. Summary

In this chapter we have given the difference in simulation tools and we have compared three different approaches to mixed-signal validation. While these three methods have advantages and disadvantages, behavioral modeling is the best solution that allows the validation at a high level the entire system's architecture in a reasonable amount of time. Therefore, based on specifications derived from the high-level simulation, the detailed design could be performed. To summarize the contribution of this work – namely providing some formalism to behavioral modeling – Chapter 4 will lay down general guidelines developed to writing behavioral models. Next, to see how these guidelines can be applied, we will discuss models for different types of circuits.

Chapter 4

Proposed approach for mixed signals validation

1. Introduction

Based on what was presented in the chapter 3, behavioral modeling seems to be the ideal approach which allows mixed-signal SoC validation in a very short time.

The purpose of this chapter is to identify the necessary elements for writing behavioral models for SoC validation. First, we will describe the need of choosing SYSTEM VERILOG as a modeling language. Then, the importance of circuit partitioning for the integration in a digital simulation environment. Next, since our simulation tool is purely digital, therefore, we will describe a method which allows to represent analog signals. Lastly, an approach, which allows to calculate the output of the models in an efficient manner, will be proposed.

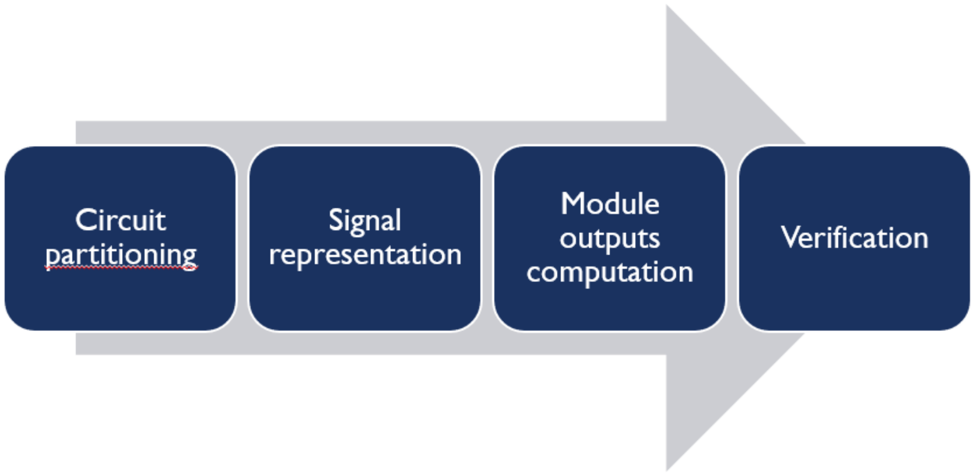


Figure 11: Methodology of the proposed solution

2. Modeling language

In Chapter 3, we saw that the simulation speed of Simulink and SystemC-AMS is similar at the system level but there is no resemblance between the written models and the actual physical implementation [45]. However, the digital languages (VHDL/Verilog/SystemVerilog) with real numbers capabilities can reduce the simulation run time compared to Verilog-A/Verilog-AMS/VHDL-AMS [44]. Moreover, with SystemVerilog, we no longer need to recreate the digital block of Soc in a different platform. For all the reasons mentioned above, SystemVerilog is our modeling language.

3. Circuit partitioning

The digital circuits are characterized by a unidirectional propagation of the signal; in other words, the output of one gate drives only the input of the next [29]. Therefore, the digital simulators are designed to cope with the unidirectional propagation of the signal and uses the fact that a gate's input changes cause it to re-evaluate only that gate.

However, the principle of unidirectionality does not exist in analog circuits. Non-unidirectional nodes usually occur when multiple current sources are summed or both current and voltage signals co-exist at a single point. Therefore, to deal with this inconvenience, digital simulator requires that analog designs must be partitioned into modules which are unidirectional and breaks any loop or feedback in analog design. The best example to explain this, is the current summing nodes. The figure 12 below shows a current DAC where the output of the current sources is current, while the feedback resistor drives a voltage back to the input node of the buffer. This causes conflict.

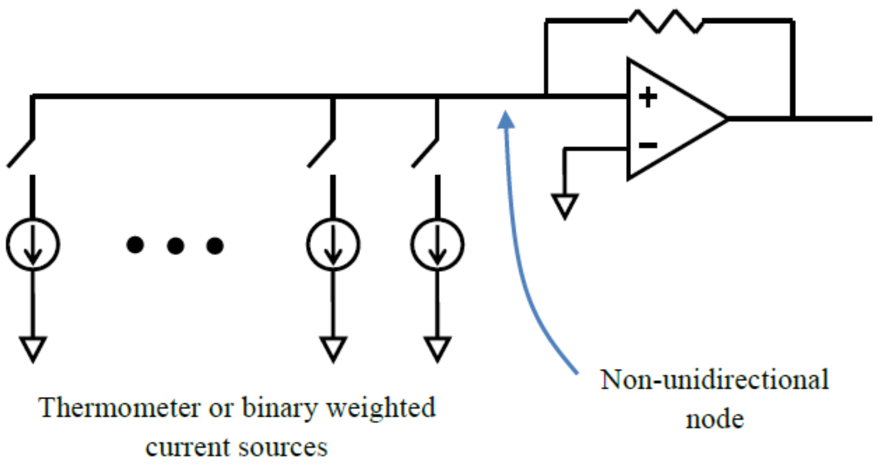


Figure 12. Current summing node in DAC [30]

For the DAC, the partitioning method consists in combining all the current sources in a block with a single current output. For the feedback resistance, it will be modeled as a block with a current input and a voltage output. For each combined block, we need a wrapper for checking the functional model. Figure 13 shows the partitioning procedure used for analog circuits.

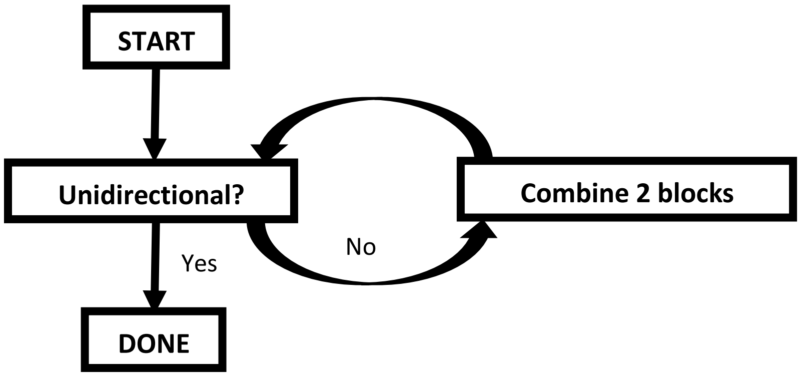


Figure 13. Circuit partitioning procedure

This partitioning operation is executed by an automatic program which detects the nodes with several current drivers or current / voltage conflict and hides these nodes until there are none left.

4. Signal representation

After the analog circuits are properly segmented into unidirectional blocks, the next challenge to overcome is the fact that digital simulators work with Boolean values and discrete events, while analog signals are continuous.

In this section we will explain the problems encountered during the representation of sampled data of analog signals and we will propose a piecewise linear (PWL) representation which facilitates the interaction with digital signals and then the XMODEL representation.

1
Sampled data representation
2
Piecewise linear representation (PWL)
3
XMODEL

Figure 14: Signal representation

4.1.Sampled data representation

We saw in chapter 3 that the sampled data representation of certain analog ‘synchronous’ circuits like delta-sigma and a switched capacitor filter fits perfectly in digital simulators with extensions of real number.

However, for ‘asynchronous’ analog circuits, the approach of sampled data representation is not as effective as in the case of synchronous circuits.

A classic example is a clockless comparator. As Figure 15 shows, a comparator toggles its output from low to high as its positive input increases in value beyond its negative input. Note that the crossing point is not in the same position for the continuous case and the sampled case and this is due to the lack of information on the signal shape between samples updates. It is possible to decrease this error by increasing the sampling frequency, but this will increase the simulation time. The fundamental issue here is that the digital simulator cannot interpolate between two signal updates until the second sample arrives.

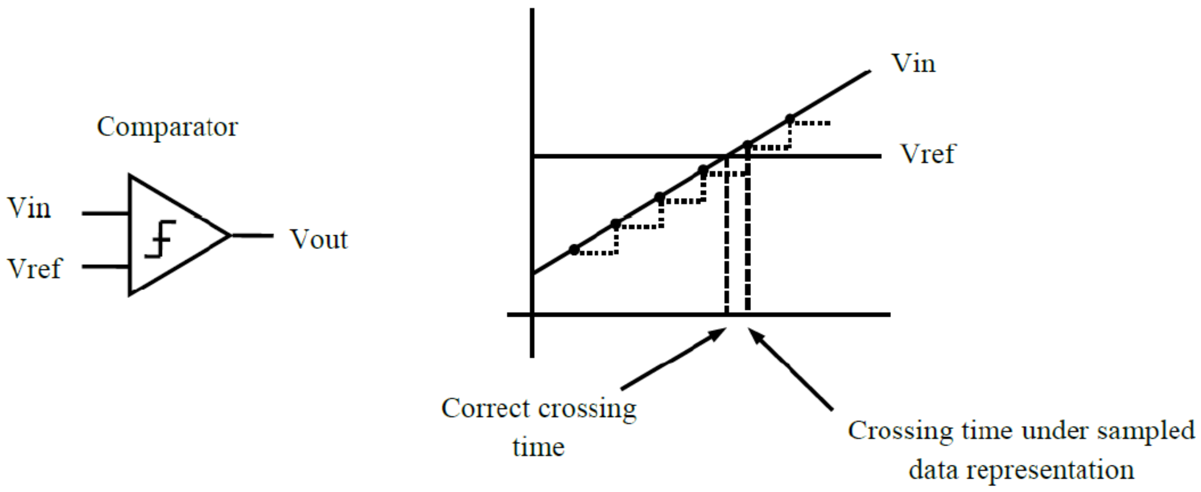
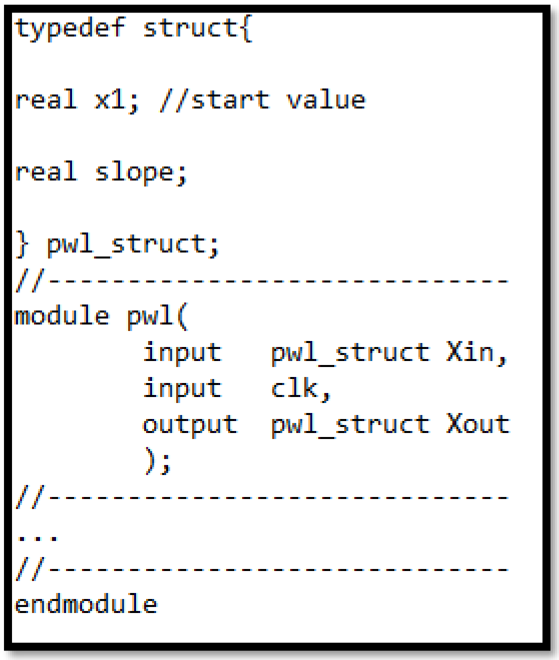


Figure 15. Sampled Data Representation of an analog comparator

4.2.Piecewise linear representation (PWL)

For asynchronous analog circuits, the representation of sampled data is no longer the best solution due to the lack of information on the signal shape between sample updates. To overcome this drawback, it is necessary to increase the samples in order to have more information; it is the representation of the piecewise linear signal (PWL).

This representation consists of describing the analog signals in a piecewise linear way, each linear segment begins at a value and continues on the slope. The following listing represents a SystemVerilog structure for a piecewise linear representation. This structure contains a starting value v1 of the signal and a slope, as well as how it can be defined as the type of input / output port of a module.



```
typedef struct{

real x1; //start value

real slope;

} pwl_struct;
//----------------------------
module pwl(
        input   pwl_struct Xin,
        input   clk,
        output  pwl_struct Xout
        );
//----------------------------
...
//----------------------------
endmodule
```

Listing 2. SystemVerilog structure for piecewise linear representation

The piecewise linear representation gives a certain freedom for the temporal instance of each signal sample and this through a change in one or the other element of the structure during each update. This method allows digital simulators to generate asynchronous events and capture the dynamics of circuits, thus solving the problem with the representation of sampled data identified in the previous section. The following figure illustrates the enabling capabilities of the PWL representation to model all types of circuits and interfaces.

Figure 16 shows the enabling capabilities of the PWL representation in general terms. It describes all the circuits and interfaces that can be modeled.

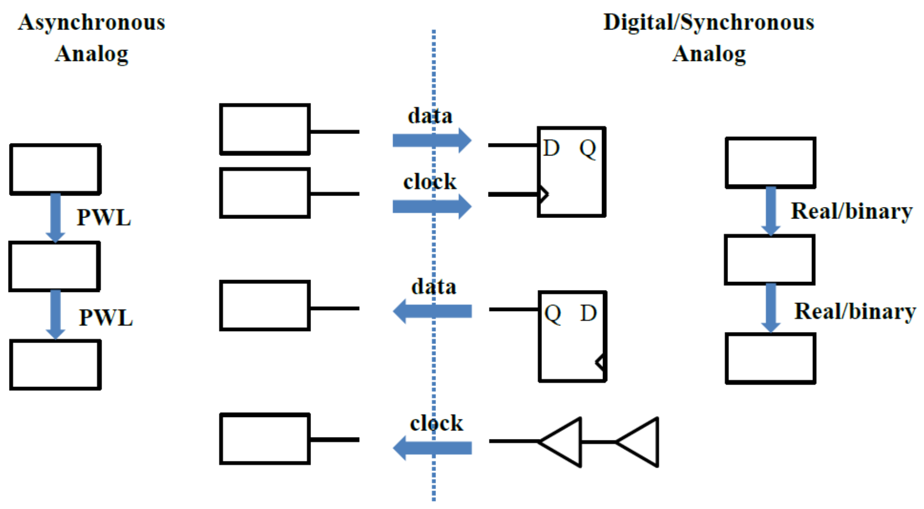


Figure 16. Communication between Analog/Digital components.

If a PWL data signal passes from the asynchronous analog domain to the digital domain, it needs a digital clock to sample it and the sampled value can be interpolated at the edge of the clock according to the slope of the signal. If the signal passes as a clock signal, there could be a delay or asymmetry of the clock signal (As in the case of the comparator discussed previously). To resolve the asynchronous crossover of the comparator, the slope information can be modified so that the simulator can interpolate.

If a clock / data signal passes from the digital domain to the asynchronous analog domain, its format must be converted into a PWL structure.

Finally, the representation of the PWL signal makes it possible to model a chain of asynchronous circuits. For a chain to exist, individual analog models must consume a PWL signal from the previous block and produce a PWL signal for the next block. The details of the required calculation will be the subject of the following section.

4.3.XMODEL

XMODEL is another approach to behavioral modeling [46] which consists of writing behavioral models which describe analog functions as almost linear filters in the s domain. The increase in data is obtained here through the equivalence between a time domain signal and its Laplace transform. The following equation represents waveforms x (t) with their equivalent Laplace domain expressions X (s):

$$x(t) = \sum_i c_i t^{m_i} e^{-a_i t} \quad \rightarrow \quad X(s) = \sum_i \frac{b_i}{(s+a_i)^{m_i+1}} \qquad (6)$$

The coefficients {ai, bi, mi} uniquely identify an analog signal. Any change of signal in the time domain will be translated by the change of the coefficients {ai, bi, mi}.

The output of a model can then be computed by multiplying the s-domain representation of the input with the s-domain transfer function of the system. Consequently, unlike the PWL representation, XMODEL does not require steps in time; however, reconstruction is necessary to view the waveforms in time domain.

The choice of representation depends on the type and complexity of the waveforms and the systems modeled.

5. Module Output Computation

The input and output of analog circuits are a multi-element, real-valued structure. The computation of the PWL output of a module due to a PWL input can be accomplished in three steps. First compute the continuous time domain response of a module due to a single linear segment on its input. Then based on the time constants of the system, a piecewise linear approximation is formed. Repeat step 1 and 2 with the next arriving input linear segment. Lastly, to be efficient, output updates that are within a certain error tolerance are removed.

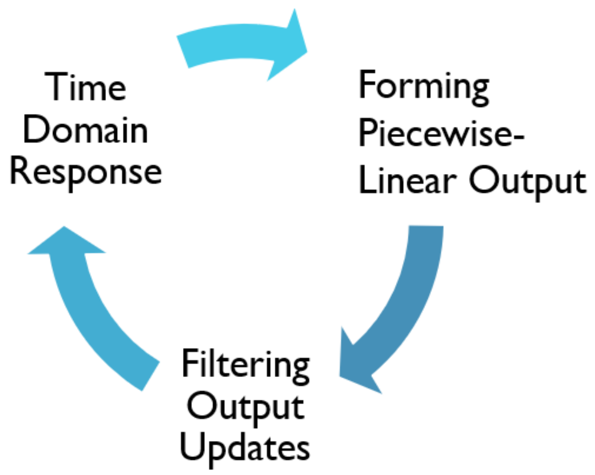


Figure 17: Steps for module output computation

5.1.Time Domain Response

A linear piecewise input is defined as a set of successive input but delayed and each is characterized by an initial value and a slope. When the system is stimulated by successive delayed inputs, a total transient response will be a trajectory traced by the evolution of the states of the system. Since the final states at the end of the current input segment are the initial states of the next input segment, and each input segment is in the same form (i.e. the value and the slope), simply examine the behavior of the system due to a single linear segment and repeat the calculation for the sequence of linear segments as they are updated.

Figure 18 shows the decomposition process of a piecewise linear segment. This decomposition consists in dividing the input element in a step of magnitude equal to the element of initial value of the structure PWL, and a ramp which starts at 0 but increases at the rate indicated by the element of slope of the structure PWL. For the final answer, it will be composed of the response to the step, the response to the ramp and the decay of the initial states of the system.

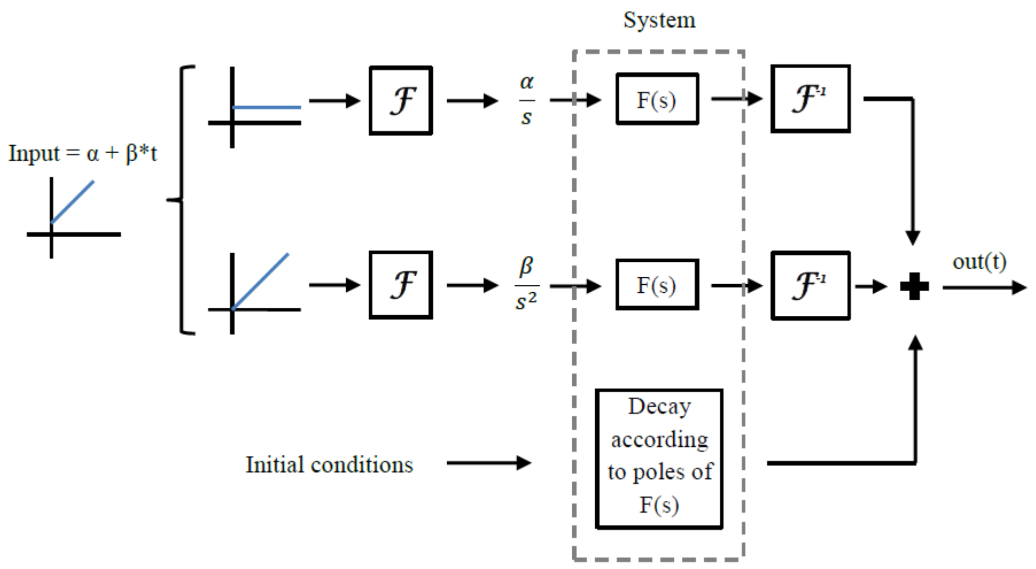
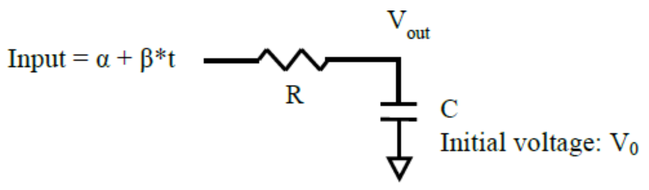


Figure 18. Output computing transformation: time and frequency domains

Consider a more concrete example (an RC circuit) described in the figure 19. The total output response to a linear segment on the input is a sum of exponentials. The last term describes the decrease in the initial state, i.e. the previous voltage on the capacitor drains slowly. The other terms constitute the response induced by the linear segment.



$$V_{out}(t) = \underbrace{\underbrace{\beta t + \beta RC(e^{-\frac{t}{RC}} - 1)}_{\text{Response to ramp}} + \underbrace{\alpha(1 - e^{-\frac{t}{RC}})}_{\text{Response to step}}}_{\text{Driven response}} + \underbrace{V_0 e^{-\frac{t}{RC}}}_{\text{Decay of initial voltage}}$$

Figure 19. Time domain response of an RC circuit

5.2.Forming Piecewise-Linear Output

After computing a module's continuous time output waveform when it is stimulated by a linear ramp, the next step is converting time output waveform stimulated by a linear ramp into piecewise linear segments. We will consider the same example seen above which the RC circuit with an element of slope 0 as input. Figure 20 shows the output waveforms of two systems with different rates of variation. Systems that change quickly require shorter linear segments. As shown in the following figure, piecewise linear segments length T1 of the first system is less than T2 of the second system.

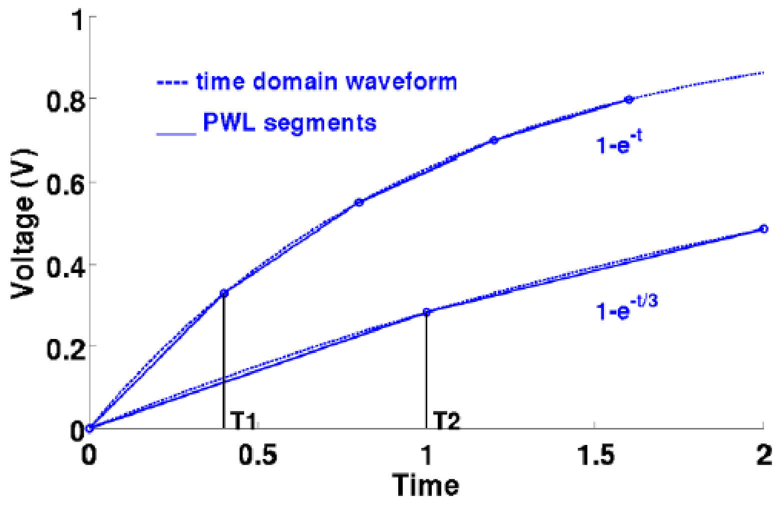


Figure 20. PWL segment length for systems with different time constants

The general idea is to estimate the length of time Δt after which the output response deviates too much from a linear ramp. After that, output immediately the segment describing the signal from the current instant until that time. Then, return after the expiry of Δt to compute the output waveform again to determine the next linear segment.

5.3.Filtering Output Updates

Filtering unnecessary output updates by comparing slopes of 02 consecutives outputs. Communication between blocks are through the propagation of an analog signal in piecewise linear format through a chain of analog modules. Each input update will result in one or more output updates which will become the input updates for the next block. Contrary to logic signals where it is very clear when an output changes, with continuous time signals it is not so clear. Thus, it is necessarily to limit output updates to avoid an unnecessarily large number of events.

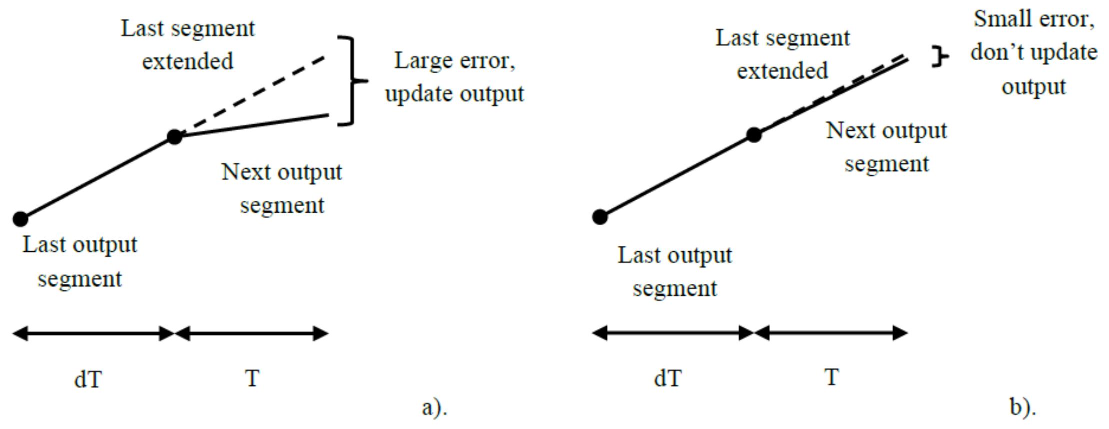
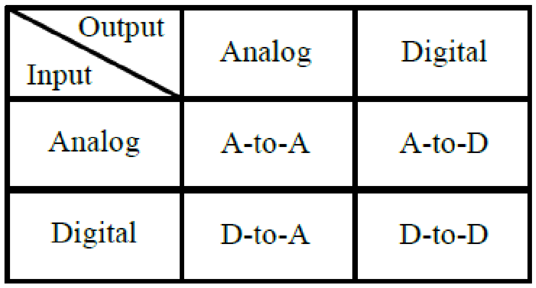


Figure 21: Filtering unnecessary output updates

6. Creating Analog Behavioral Models

After establishing the general approach for behavioral modeling, we will discuss in this section more details on the creation of models of analog circuits. Next, we will present the assertions that allow the verification of these models.

Since there are several kinds of analog circuits, we will categorize the analog functionality according to the input / output ports, as shown in the following figure.



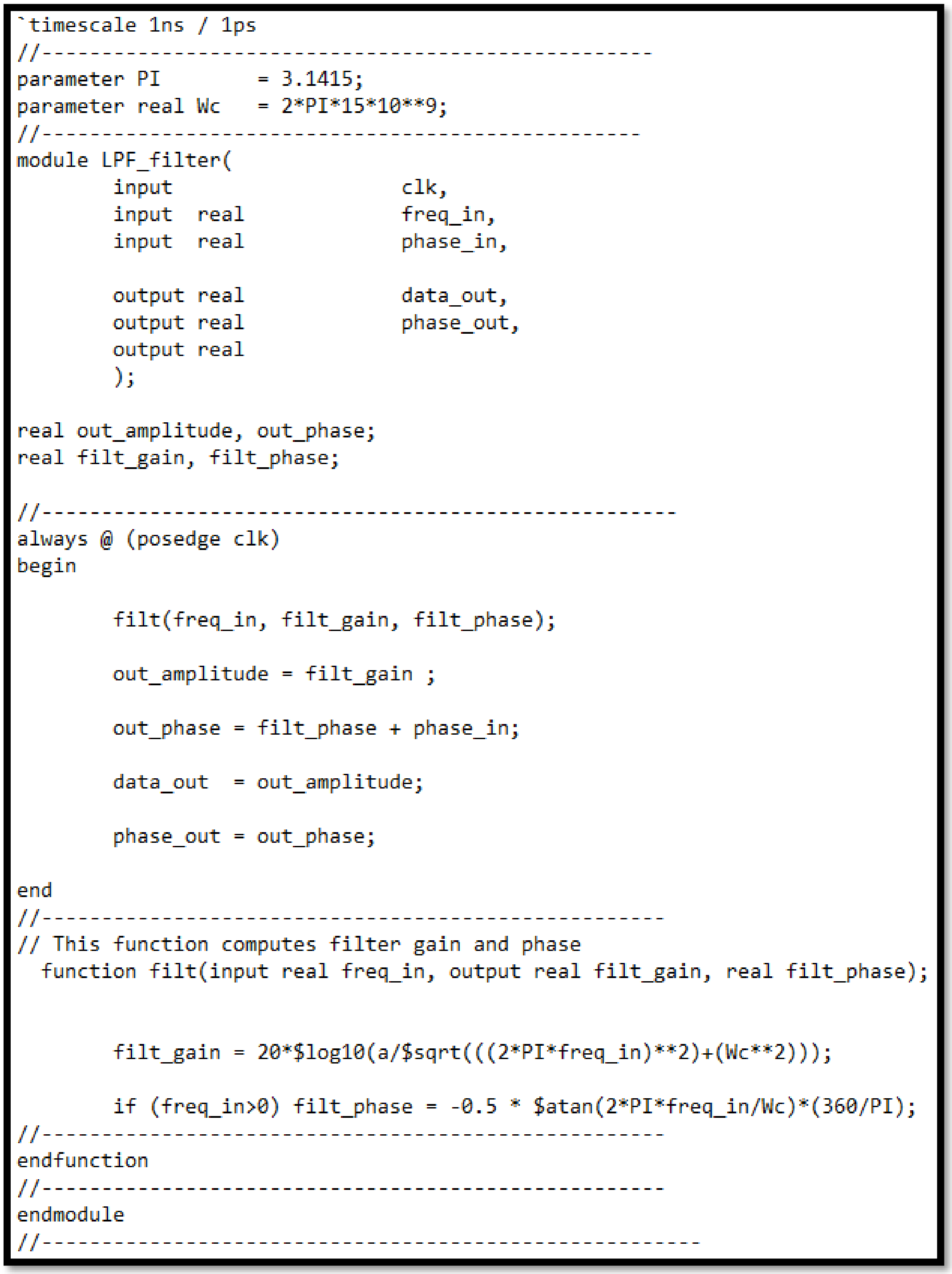
Input \ Output	Analog	Digital
Analog	A-to-A	A-to-D
Digital	D-to-A	D-to-D

Figure 22. Circuit categories based on input/output characteristic

6.1.A to A circuits

The best example to describe this kind of circuit are the filters since the analog input is filtered by a transfer function to produce an analog output. Other examples of circuits that fall into this category include linear voltage regulators, trans-impedance amplifiers (TIA), continuous time filters and programmable gain amplifiers (PGA).

The following listing is a behavioral model of a Low Pass Filter. Each circuit A to A is characterized by its transfer function. The equation presented in figure 19 is used in the case of a unipolar system. For higher order systems, a decomposition of the transfer function into a series of unipolar / zero transfer functions is necessary.



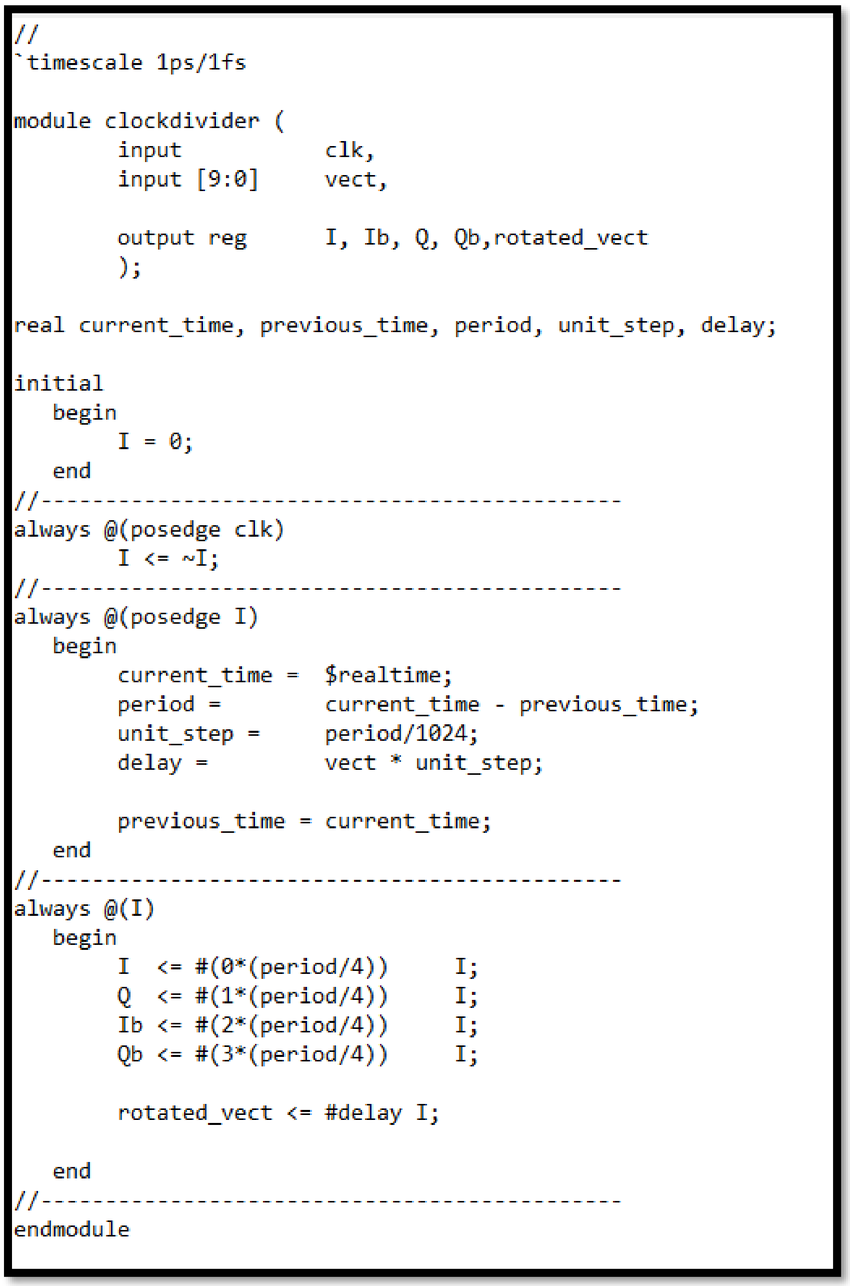
```
`timescale 1ns / 1ps
//-----------------------------------------------------
parameter PI        = 3.1415;
parameter real Wc   = 2*PI*15*10**9;
//----------------------------------------------------
module LPF_filter(
        input                   clk,
        input  real             freq_in,
        input  real             phase_in,

        output real             data_out,
        output real             phase_out,
        output real
        );

real out_amplitude, out_phase;
real filt_gain, filt_phase;

//-----------------------------------------------------
always @ (posedge clk)
begin

        filt(freq_in, filt_gain, filt_phase);

        out_amplitude = filt_gain ;

        out_phase = filt_phase + phase_in;

        data_out  = out_amplitude;

        phase_out = out_phase;

end
//----------------------------------------------------
// This function computes filter gain and phase
  function filt(input real freq_in, output real filt_gain, real filt_phase);


        filt_gain = 20*$log10(a/$sqrt(((2*PI*freq_in)**2)+(Wc**2)));

        if (freq_in>0) filt_phase = -0.5 * $atan(2*PI*freq_in/Wc)*(360/PI);
//----------------------------------------------------
endfunction
//----------------------------------------------------
endmodule
//-------------------------------------------------------
```

Listing 3. Behavioral model of a low pass filter

6.2.D to D circuits

These kinds of circuits are always analog, but they act as digital gates only within their working threshold and their inputs / outputs can be described by digital signals with an appropriate transition time. Therefore, circuits D to D are analog blocks which modify the clock signals or digital circuits which need a detailed description of their delays [7]. Examples of circuits D to D are the frequency dividers, duty cycle correction circuits and digital circuits under the influence of supply voltage fluctuations.

The following listing is a behavioral model of a Clock Divider. Les modèles d'écriture pour les circuits D-D nécessitent une bonne composition d'équation afin de quantifier le retard nécessaire. Cette modélisation ressemble à celle des circuits A-A; la différence est que le domaine de calcul est le temps et les informations analogiques sont exprimées comme un changement dans le temps plutôt que dans un changement de tension. Dans SystemVerilog, les retards sont notés par la directive #

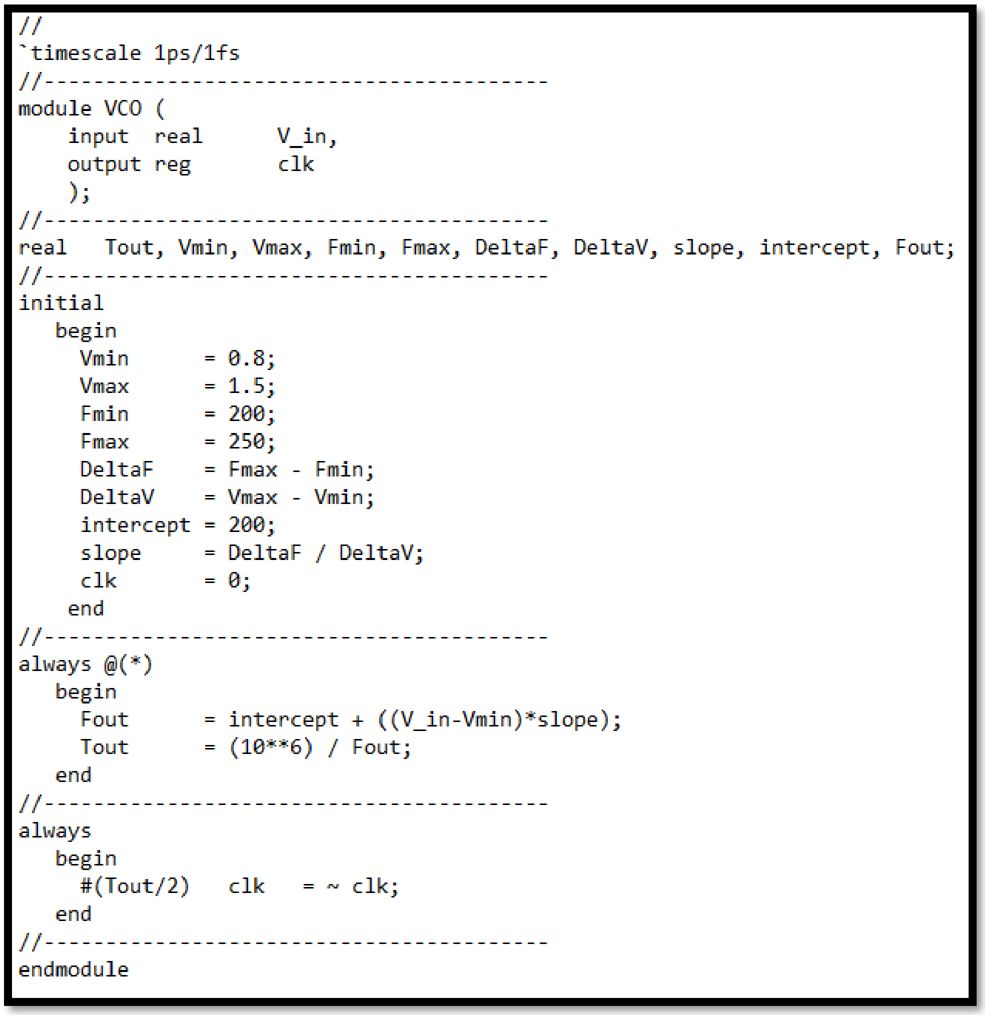


```
//
`timescale 1ps/1fs

module clockdivider (
        input           clk,
        input [9:0]     vect,

        output reg      I, Ib, Q, Qb,rotated_vect
        );

real current_time, previous_time, period, unit_step, delay;

initial
   begin
        I = 0;
   end
//---------------------------------------------
always @(posedge clk)
        I <= ~I;
//---------------------------------------------
always @(posedge I)
   begin
        current_time =  $realtime;
        period =        current_time - previous_time;
        unit_step =     period/1024;
        delay =         vect * unit_step;

        previous_time = current_time;
   end
//---------------------------------------------
always @(I)
   begin
        I  <= #(0*(period/4))     I;
        Q  <= #(1*(period/4))     I;
        Ib <= #(2*(period/4))     I;
        Qb <= #(3*(period/4))     I;

        rotated_vect <= #delay I;

   end
//---------------------------------------------
endmodule
```

Listing 4. Behavioral model of a clock divider

6.3.A to D circuits

The behavior of these types of circuits depends on the input waveform while the downstream block only depends on the transition time of the output generated by these blocks. An example of this kind of circuit is the Voltage Controlled Oscillator.

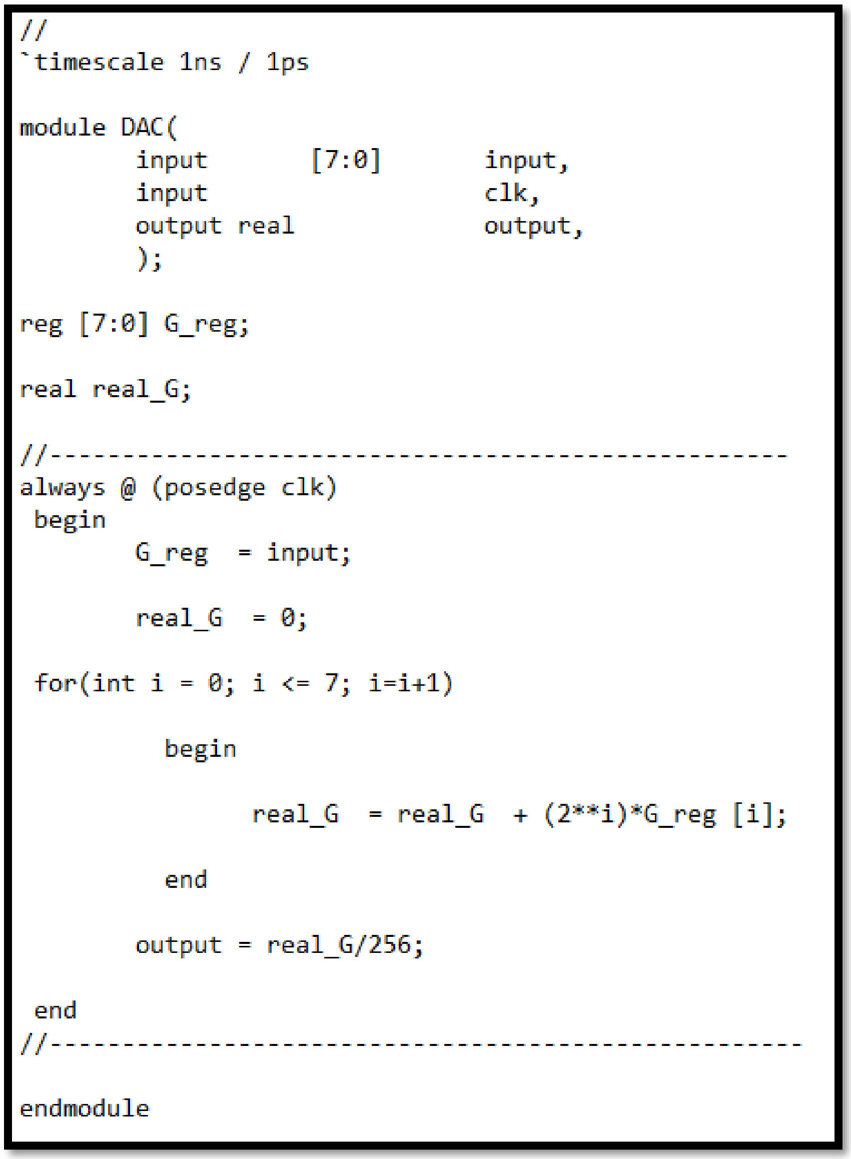


```
//
`timescale 1ps/1fs
//----------------------------------------
module VCO (
    input  real      V_in,
    output reg       clk
    );
//----------------------------------------
real   Tout, Vmin, Vmax, Fmin, Fmax, DeltaF, DeltaV, slope, intercept, Fout;
//----------------------------------------
initial
   begin
     Vmin      = 0.8;
     Vmax      = 1.5;
     Fmin      = 200;
     Fmax      = 250;
     DeltaF    = Fmax - Fmin;
     DeltaV    = Vmax - Vmin;
     intercept = 200;
     slope     = DeltaF / DeltaV;
     clk       = 0;
    end
//----------------------------------------
always @(*)
   begin
     Fout      = intercept + ((V_in-Vmin)*slope);
     Tout      = (10**6) / Fout;
   end
//----------------------------------------
always
   begin
     #(Tout/2)   clk    = ~ clk;
   end
//----------------------------------------
endmodule
```

Listing 5. Behavioral model of a Voltage control oscillator (VCO)

6.4.D to A circuits

The behavior of circuits D to A is the opposite of that of circuits A to D. It depends on the change on their inputs to produce perfectly analog outputs. The following example shows the behavioral model of a digital to analog converter.



```
//
`timescale 1ns / 1ps

module DAC(
        input       [7:0]       input,
        input                   clk,
        output real             output,
        );

reg [7:0] G_reg;

real real_G;

//-----------------------------------------------------
always @ (posedge clk)
 begin
        G_reg  = input;

        real_G  = 0;

 for(int i = 0; i <= 7; i=i+1)

          begin

                real_G  = real_G  + (2**i)*G_reg [i];

          end

        output = real_G/256;

 end
//------------------------------------------------------

endmodule
```

Listing 6. Behavioral model of a converter digital to analog

7. Behavioral model verification

After writing the behavioral model of a circuit, it is important to design a system Verilog verification environment/testbenches in order to check the correct functioning of the circuit.

A test bench is a virtual platform containing the design to be tested (UUT) and virtual wires connected to the UUT inputs and outputs. It allows to verify the correctness of a design to be implemented in hardware. He has three main purposes:

- Generate stimulus for simulation
- Apply this stimulus to the module under test and collect output responses

- Compare output responses with expected values

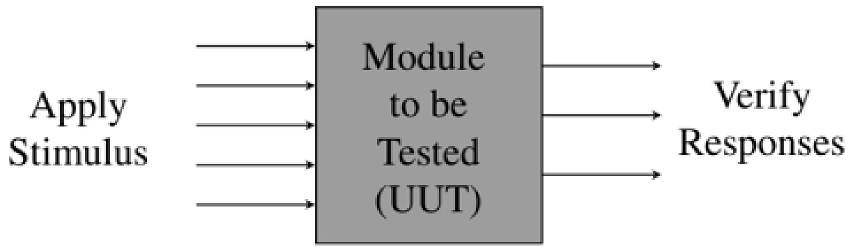
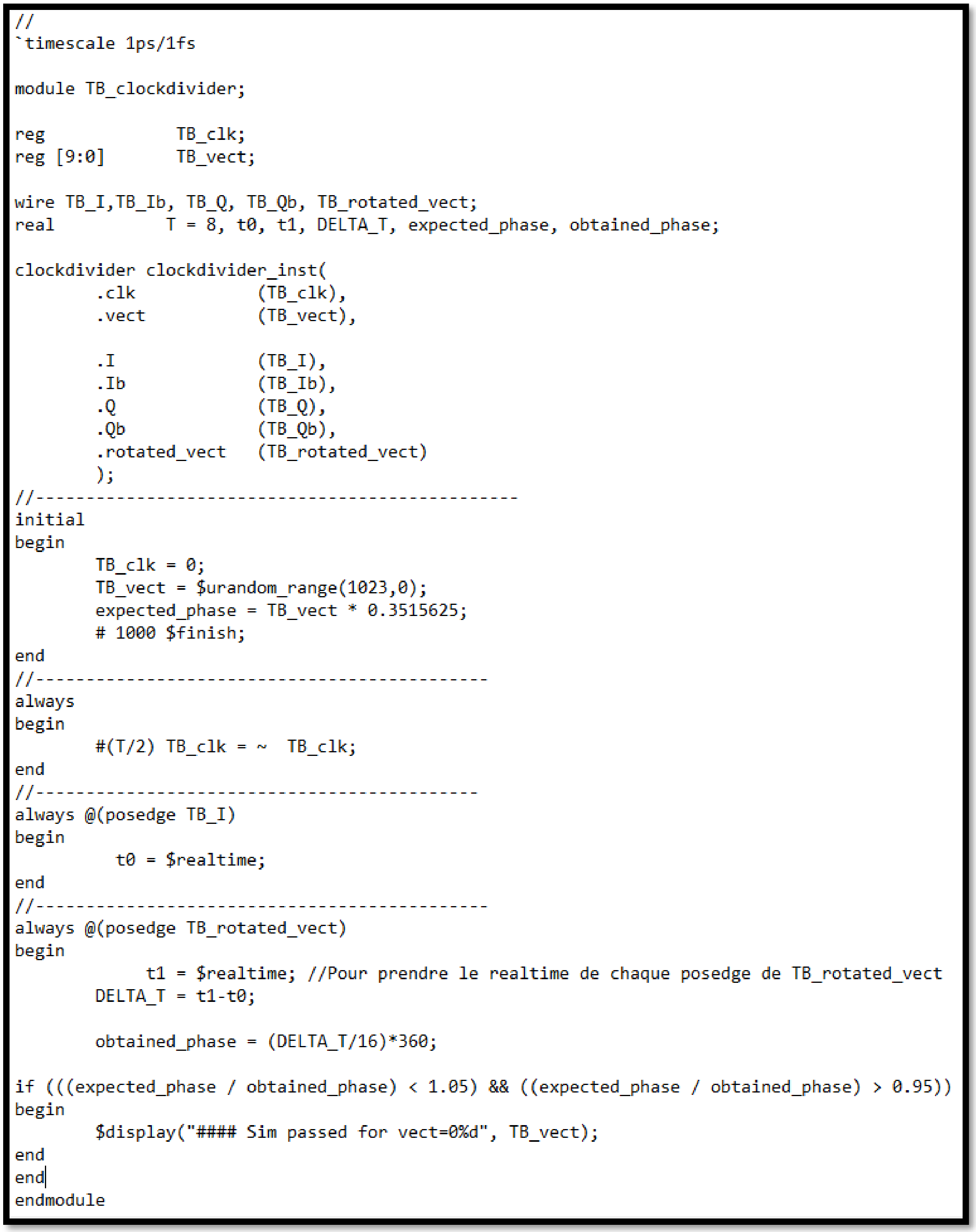


Figure 23: Test bench

The following listings are test benches designed to verify the behavioral model of a VCO and a Clock divider.

```
//
`timescale 1ps/1fs
//-------------------------------------------
module TB_VCO ();
//-------------------------------------------
real       TB_V_in, toto;
wire       TB_clk;

//-------------------------------------------
VCO TB_VCO (
    .V_in       (TB_V_in),
    .clk        (TB_clk)
    );
//-------------------------------------------
initial
begin
repeat (20)
  begin
        toto = $urandom_range(800, 1500);
        TB_V_in = toto/1000;
        # 100000;
  end
        $finish;
end
//-------------------------------------------
endmodule
```

Listing 7. Behavioral model verification of a VCO



```
//
`timescale 1ps/1fs

module TB_clockdivider;

reg             TB_clk;
reg [9:0]       TB_vect;

wire TB_I,TB_Ib, TB_Q, TB_Qb, TB_rotated_vect;
real           T = 8, t0, t1, DELTA_T, expected_phase, obtained_phase;

clockdivider clockdivider_inst(
        .clk            (TB_clk),
        .vect           (TB_vect),

        .I              (TB_I),
        .Ib             (TB_Ib),
        .Q              (TB_Q),
        .Qb             (TB_Qb),
        .rotated_vect   (TB_rotated_vect)
        );
//------------------------------------------------
initial
begin
        TB_clk = 0;
        TB_vect = $urandom_range(1023,0);
        expected_phase = TB_vect * 0.3515625;
        # 1000 $finish;
end
//---------------------------------------------
always
begin
        #(T/2) TB_clk = ~  TB_clk;
end
//--------------------------------------------
always @(posedge TB_I)
begin
          t0 = $realtime;
end
//---------------------------------------------
always @(posedge TB_rotated_vect)
begin
             t1 = $realtime; //Pour prendre le realtime de chaque posedge de TB_rotated_vect
        DELTA_T = t1-t0;

        obtained_phase = (DELTA_T/16)*360;

if (((expected_phase / obtained_phase) < 1.05) && ((expected_phase / obtained_phase) > 0.95))
begin
        $display("#### Sim passed for vect=0%d", TB_vect);
end
end
endmodule
```

Listing 8. Behavioral model verification of a Clock Divider

8. Summary

In this chapter we have seen the three objectives of behavioral modeling for mixed-signal validation. The first objective consists in partitioning the analog circuits into unidirectional blocks in order to adapt to the digital simulation environment. The second objective concerns the representation of analog signals with the piecewise linear approximation. Finally, the third objective concerns the choice of the appropriate linearity domain which makes it possible to quickly evaluate the outputs of a module. At the end of the chapter, we have demonstrated the use of these techniques in the creation of behavioral models for different types of analog circuits.

Chapter 5

Case study of a mixed system and experimental results

1. Introduction

In this chapter, we present the theoretical model and experiment results used to implement the behavioral model of mixed system following the methodology presented in chapter 4. As it is going to be detailed, the behavioral model is analyzed in time domain as well in frequency domain.

2. Block diagram of mixed system

The following figure represents the mixed system on which we will develop our behavioral model with SystemVerilog and we will simulate it with a purely numerical tool which is the VCS.

This mixed system is essentially composed of two blocks. The first block is a digital block which contains two digital / analog converters which convert the digital inputs into real values and two registers in order to record the information. It also contains an analog / digital converter which takes the actual values that come from the analog block and will convert them into binary values. Regarding the analog block, it is composed of an amplifier, a high pass filter and an RMS block which calculates the rms value of the filter output.

The inputs and outputs of the analog block are text files of 600 rows and two columns that represent the gain and phase of the input and output signal. For the numeric block, its input is a 600-line text file containing 8-bit numeric values and its output is an 8-bit numeric value.

In the following paragraph we will detail the specifications of each block. These specifications must be observed when developing the behavioral model of the whole system.

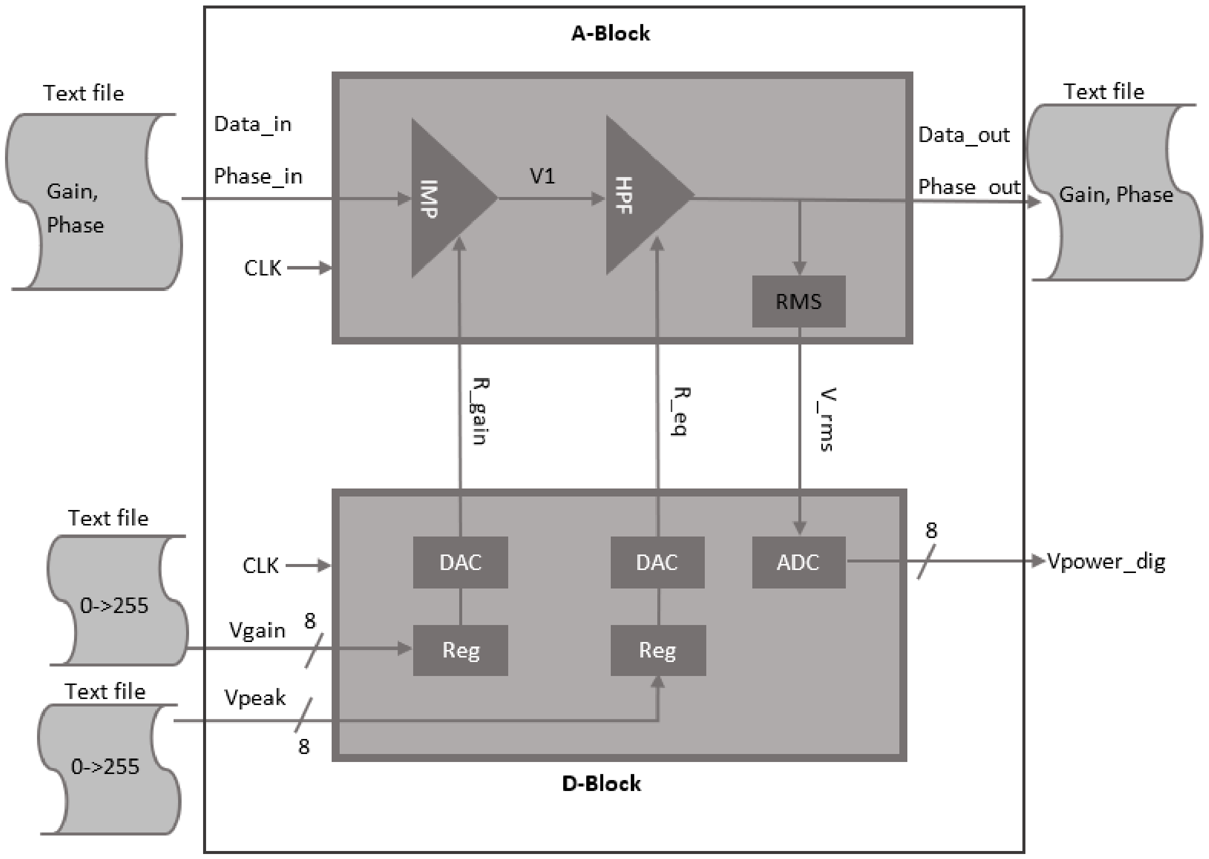


Figure 24. Mixed system block diagram

3. Analog block model specifications

3.1. Analog Input data

The input data is a file containing the spectral content of a signal in 100MHz steps from 0MHz to 60 GHz. Each bin is expressed by using two Cartesian terms i.e. real and imaginary. For a simple initial case, the output impedance of the driver providing this signal spectrum is ignored. The text file is a matrix of 2 columns (real and imaginary) by N columns where N = 600.

3.2. Control Input Data

Control input data consists of Vgain and Vpeak which are voltages which update on the rising edge of the digital block and remain fixed for that digital clock cycle. Vgain and Vpeak are 8 bits vectors ie they change from 0 to 255.

3.3. Analog Behavioral model:

The analog behavioral model has two cascaded sections and one parallel section. These are described below:

3.3.1. Gain Section:

The gain section applies a universal scaling multiplier to each bin based on an input voltage called Vgain. Gain will be 10dB per volt where the input scales from 0dB to 10dB as the input voltage Vgain changes from 0V to 1V. The relationship between input voltage and the gain is linear. This function can be satisfied by a matrix multiplication ie the input 2xN matrix multiplied by a scalar.

3.3.2. Filter Section:

The filter section provides a peaked high pass filter function to the bins from the Gain section. For initial simplicity sake, the peaking will be linear wherein the initial bin sees no multiplication, but each successive bin sees 0.01dB x Vpeak more gain than the previous bin. The input voltage Vpeak scales from 0V to 1V. For simplicity sake, no delay is modeled ie no phase rotation of bins. As we saw in gain section, the function of the filter can be represented by a matrix multiplication. The cut-off frequency is fixed at 15GHz.

3.3.3. Power Detector Block:

The peak detector hangs off the output of the Filter Section. The Power Detector Block sums the total power in all the bins and presents this power as an output voltage This voltage is named V_rms.

3.4.Analog Output Data

The analog output data is a file containing the spectral content of the Filter Section which should be in Cartesian form having signal bins from 0MHz to 60GHz in 100MHz steps.

3.5.Control Output Data

The control output data consists of the Power Detector output block which is in volts.

4. Digital block model specifications

4.1.Digital Input Data

the digital input data consists of a data file, updated on each clock cycle of the clock driving the digital block. This data file contains 8bits which represent Vgain and 8bits which represent Vpeak.

4.2.The digital block

The digital block contains the following

- A register clocked by CLK to capture the Vgain value and a register clocked by CLK to capture Vpeak.
- An ideal 8 bits ADC to convert Vgain to R_gain and an ideal 8 bits ADC to convert Vpeak to R_eq.
- An ideal 8 bits DAC to convert V_rms to Vpower_Dig.
- A register clocked by CLK to capture Vpower_Dig.

4.3.Digital Output Data:

The digital output data is the output of the Vpower_Dig register.

5. Simulation's objective

On each cycle of CLK, the digital block is to be operated, resulting in new values of Vgain and Vpeak. At the end of each DigClock cycle, the analog functional blocks sequentially carry out their calculations and generate an analog output matrix that and a single voltage output that is called Vpower.

Assuming you can assign the above matrix values to the symbol pins of a behavioral blocks in a Verilog simulation, this can all be run in one simulation. There are no funny clocking issues to contend with because while the I/O are vectors, they are unchanging and require one single mathematical operation on rising edge of DigClock.

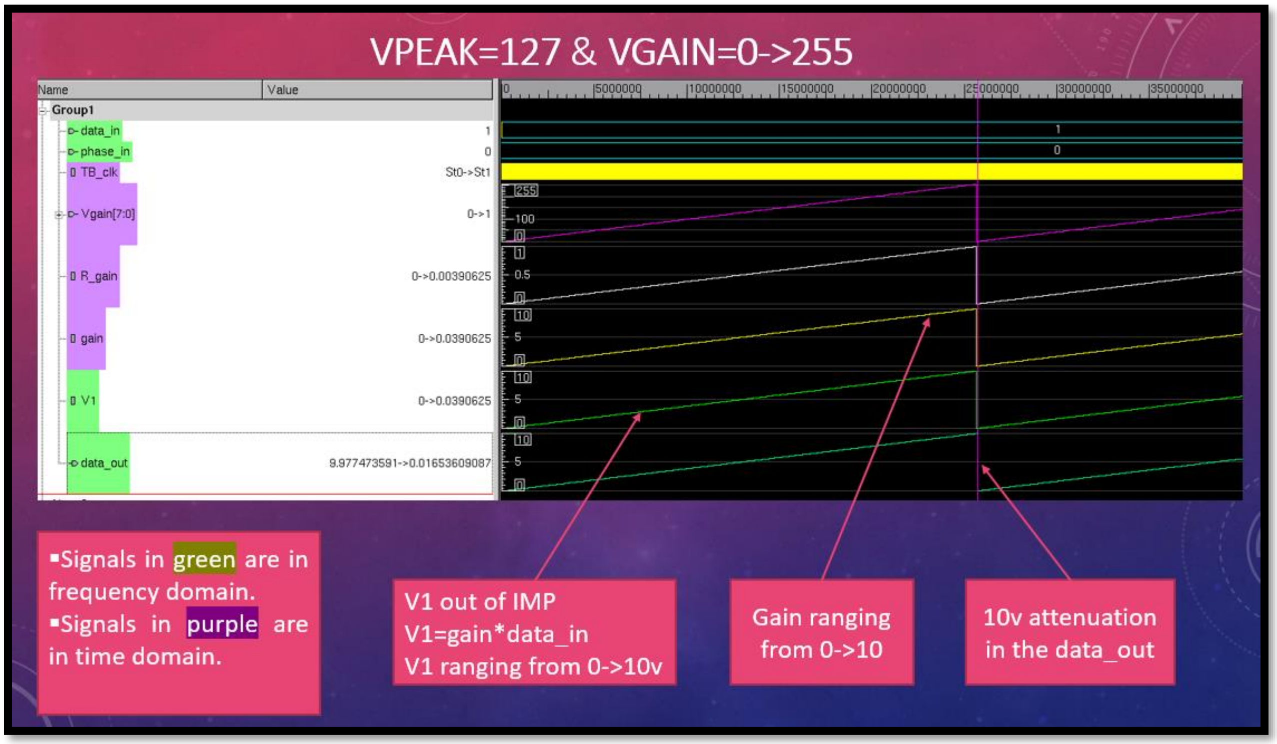
6. Simulation results

6.1. Validation of amplifier operation

To validate the operation of the amplifier we designed a test bench in which we varied the input voltage Vgain from 0 to 255(i.e Vgain on 8 bits) while Vpeak is set to 127.

On each rising edge of the clock, the test bench reads the content of the input Vgain file of the digital block and the content of the input data file of the analog block and writes the result in an output data file.

Since initially the maximum gain is fixed at 10v and the input amplitude is fixed at 1v, the variation of Vgain between 0 and 255 allows us to visualize the variation of the output of the amplifier V1 between 0 and 10V as shown the figure below.


VPEAK=127 & VGAIN=0->255

Name	Value
Group1	
data_in	1
phase_in	0
TB_clk	St0->St1
Vgain[7:0]	0->1
R_gain	0->0.00390625
gain	0->0.0390625
V1	0->0.0390625
data_out	9.977473591->0.01653609087

▪Signals in green are in frequency domain.
▪Signals in purple are in time domain.

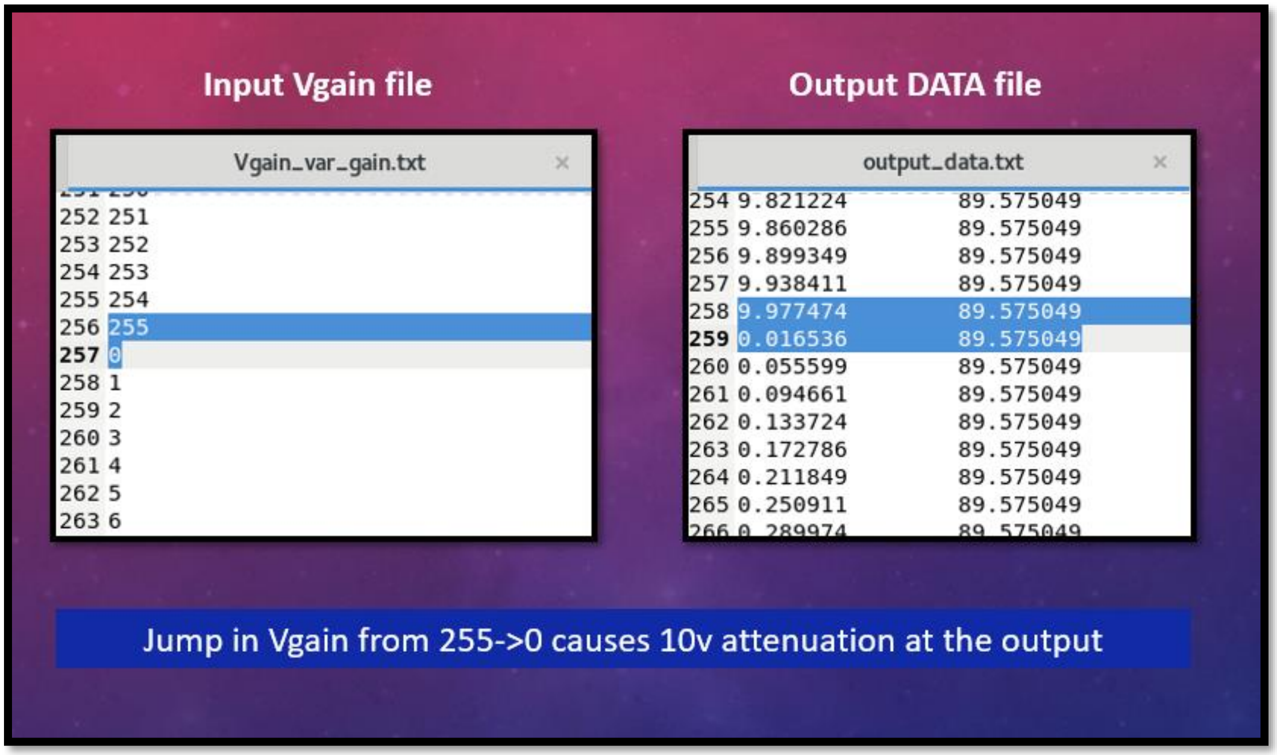
V1 out of IMP
V1=gain*data_in
V1 ranging from 0->10v

Gain ranging from 0->10

10v attenuation in the data_out

Figure 25: Amplifier's simulation with VCS

The figure below allows to check if the variations of the results obtained in the output file are synchronized with the variations of the input Vgain file. In addition, it allows us to check the compatibility between the results obtained with VCS and the data of the text files. The verification is done at the Vgain jump from 255 to 0.



```
252 251
253 252
254 253
255 254
256 255
257 0
258 1
259 2
260 3
261 4
262 5
263 6
```

```
254 9.821224         89.575049
255 9.860286         89.575049
256 9.899349         89.575049
257 9.938411         89.575049
258 9.977474         89.575049
259 0.016536         89.575049
260 0.055599         89.575049
261 0.094661         89.575049
262 0.133724         89.575049
263 0.172786         89.575049
264 0.211849         89.575049
265 0.250911         89.575049
266 0.289974         89.575049
```

Jump in Vgain from 255->0 causes 10v attenuation at the output

Figure 26: Inputs and outputs in text files

In the simulation with VCS, we notice that there is an attenuation of 10v in data_out. This is confirmed through the output text file which shows the same attenuation rate when Vgain jumps from 255 to 0.

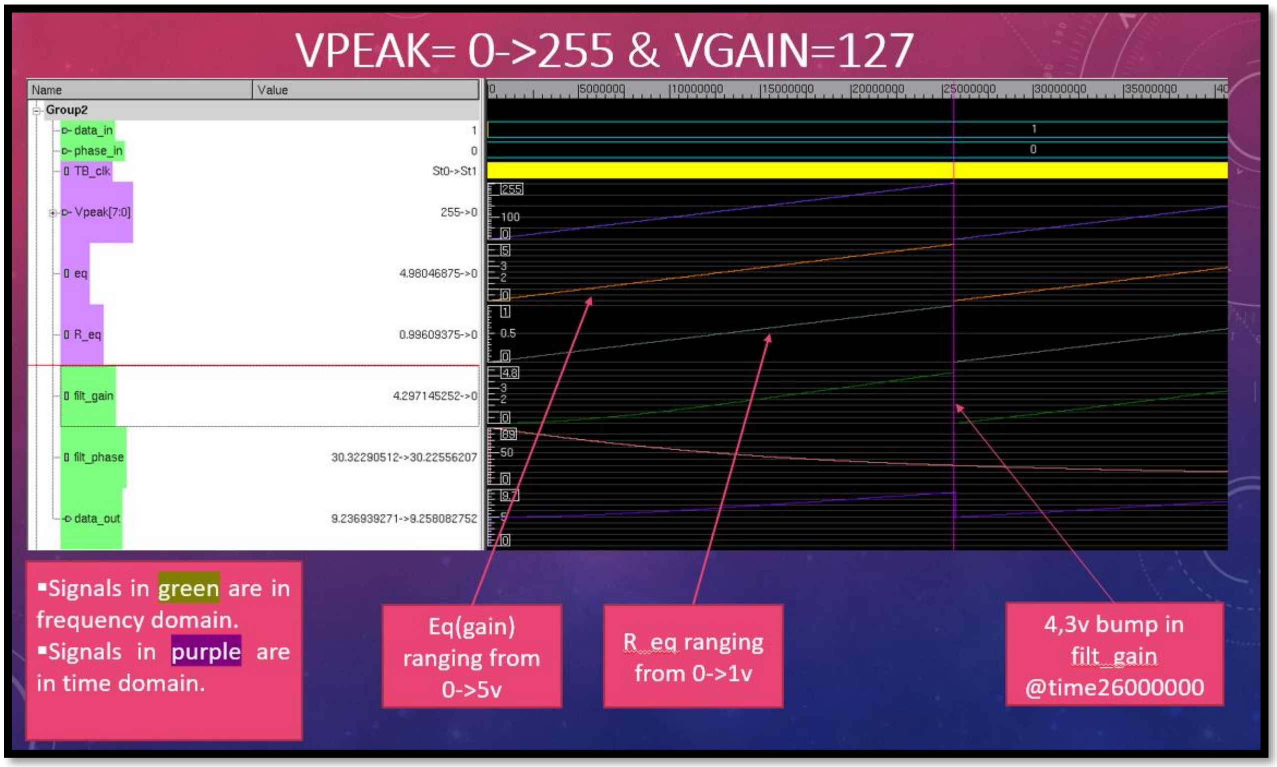
Since the results obtained correspond perfectly to those obtained with the VCS tool, we can then say that the behavior of our amplifier meets the requested specifications.

6.2.Validation of equalizer operation

Our equalizer is essentially composed of an amplifier and a high pass filter. To validate the operation of the equalizer, we must first validate the operation of the amplifier and then validate the operation of the high pass filter. For this we have set Vgain to 127 and we varied Vpeak from 0 to 255.

The test bench works in the same way as for the amplifier, on each rising edge of the clock he reads the content of the input Vpeak file of the digital block and the content of the input data file of the analog block and writes the result in an output data file.

A first validation is done at the gain level of the equalizer. Initially the maximum eq_gain is fixed at 5v. So, when Vpeak is ranging from 0 to 255 we can see in figure 27 that Eq is ranging from 0 to 5v.


VPEAK= 0->255 & VGAIN=127

Name	Value
Group2	
data_in	1
phase_in	0
TB_clk	St0->St1
Vpeak[7:0]	255->0
eq	4.98046875->0
R_eq	0.99609375->0
filt_gain	4.297145252->0
filt_phase	30.32290512->30.22556207
data_out	9.236939271->9.258082752

- Signals in green are in frequency domain.
- Signals in purple are in time domain.

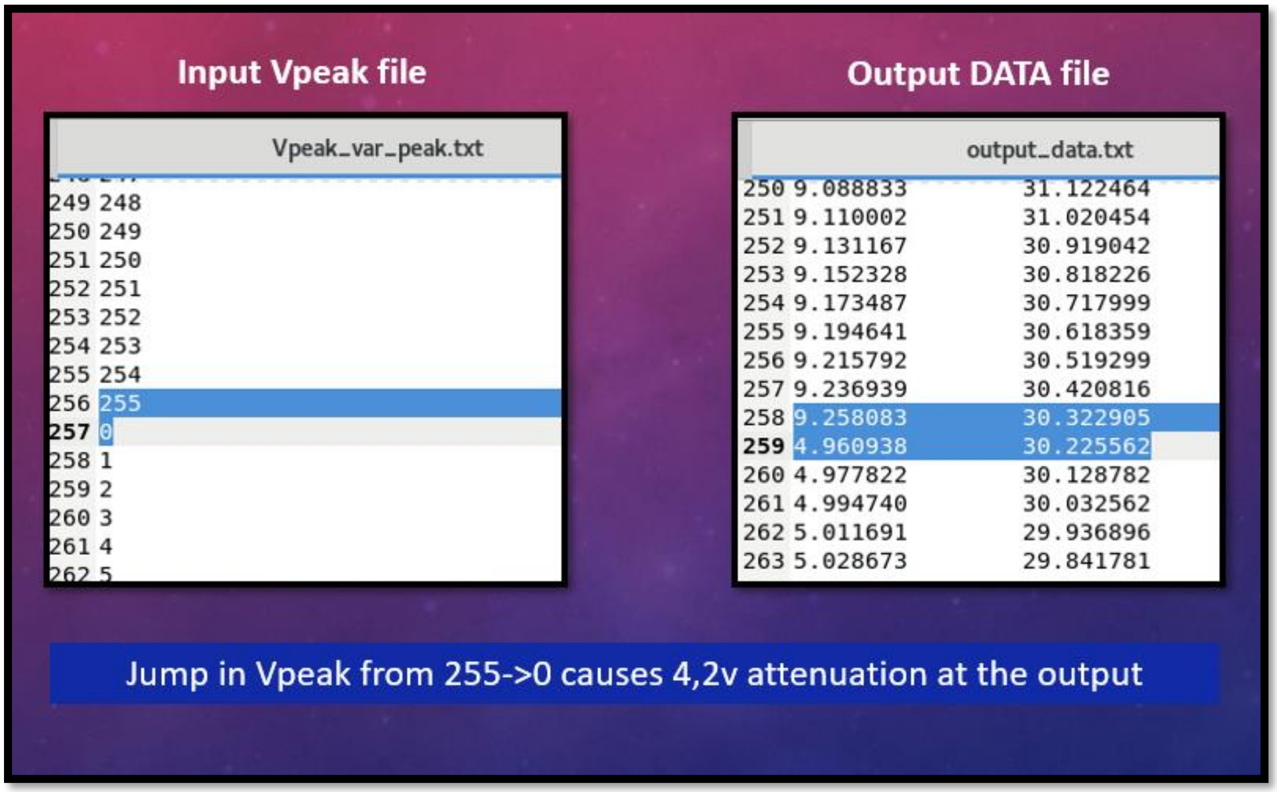
Eq(gain) ranging from 0->5v

R_eq ranging from 0->1v

4,3v bump in filt_gain @time26000000

Figure 27: Equalizer's simulation with VCS (1^{ST} validation)

The figure below allows to check if the variations of the results obtained in the output file are synchronized with the variations of the input Vpeak file. The verification is done at the Vpeak jump from 255 to 0. We see here that this jump corresponds perfectly to an attenuation of 5V in the output file.



```
249 248
250 249
251 250
252 251
253 252
254 253
255 254
256 255
257 0
258 1
259 2
260 3
261 4
262 5
```

```
250 9.088833    31.122464
251 9.110002    31.020454
252 9.131167    30.919042
253 9.152328    30.818226
254 9.173487    30.717999
255 9.194641    30.618359
256 9.215792    30.519299
257 9.236939    30.420816
258 9.258083    30.322905
259 4.960938    30.225562
260 4.977822    30.128782
261 4.994740    30.032562
262 5.011691    29.936896
263 5.028673    29.841781
```

Jump in Vpeak from 255->0 causes 4,2v attenuation at the output

Figure 28: Inputs and outputs in text files

The second check is made at the high pass filter. To perform such validation, we must first know the behavior of a high pass filter. A high pass filter is a filter that lets high frequencies pass and attenuates low frequencies, that is, frequencies below the cutoff frequency.

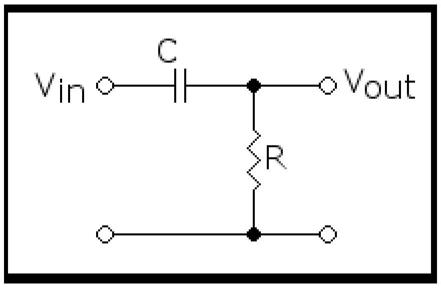


Figure 29: Diagram of a high pass filter

Figure 29 represents the diagram of a high pass filter. Its transfer function is as follows:

$$H(j\omega) = \frac{v_o}{v_i} = \frac{jRC\omega}{1 + jRC\omega} \tag{7}$$

In this equation, *j* is a complex number, such that $j^2 = -1$, and ω is the circuit pulsation or radial frequency, expressed in rad / s. As the cutoff frequency of an RC circuit is:

$$f_c = \frac{1}{2\pi RC} \text{ ou } \omega_c = \frac{1}{RC} \tag{8}$$

We find with the observable physical quantities used in the diagrams of Bode (figure 30):

- The gain in decibels:

$$G_{dB}(\omega) = 20 \cdot \log |H(j\omega)| = 20 \cdot \log\left(\frac{\omega}{\omega_c}\right) - 10 \cdot \log\left(1 + \left(\frac{\omega}{\omega_c}\right)^2\right) \tag{9}$$

- The phase in radians:

$$\begin{aligned}\phi(\omega) = \arg H(\omega) &= \frac{\pi}{2} - \arg\left(1 + j\frac{\omega}{\omega_c}\right) \\ &= \frac{\pi}{2} - \arctan\left(\frac{\omega}{\omega_c}\right)\end{aligned} \tag{10}$$

We can then distinguish two ideal situations:

- When $\omega \ll \omega_c$:

$$G_{dB} \sim 20 \cdot \log\left(\frac{\omega}{\omega_c}\right) \text{ et } \phi \simeq 90 \tag{11}$$

- When $\omega \gg \omega_c$:

$$G_{dB} \simeq 0 \text{ et } \phi \simeq 0 \tag{12}$$

- When $\omega = \omega_c$:

$$G_{dB} = \text{-3 dB.} \tag{13}$$

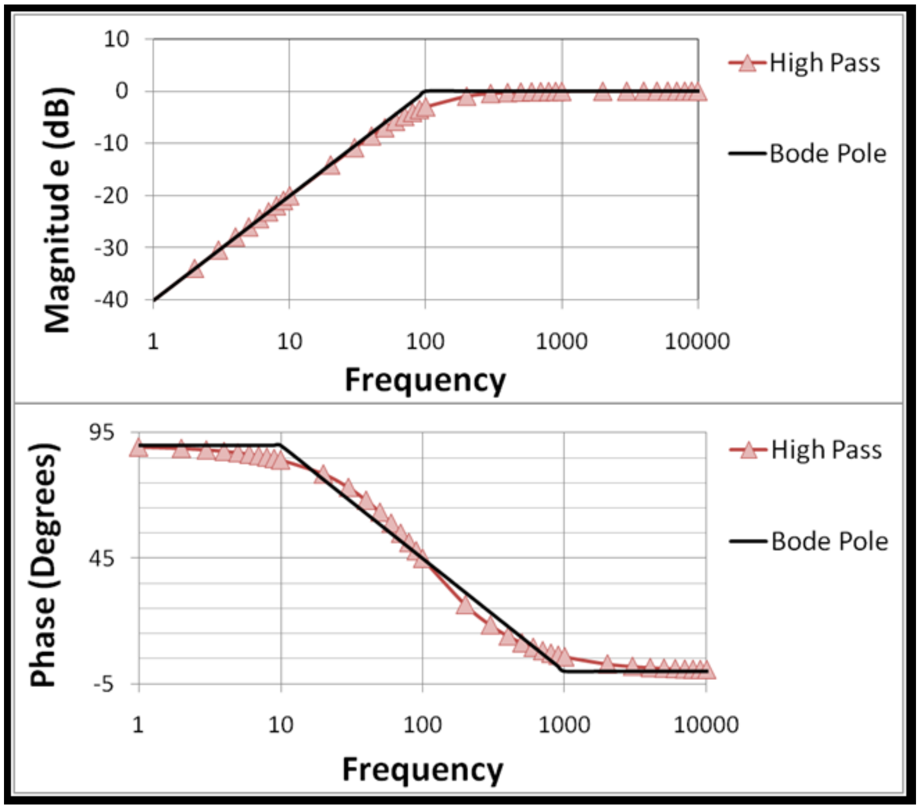


Figure 30: Bode diagram of a high pass filter (1st order system)

After having presented a reminder of the high filter operation, we will now compare the results obtained with VCS with those expected.

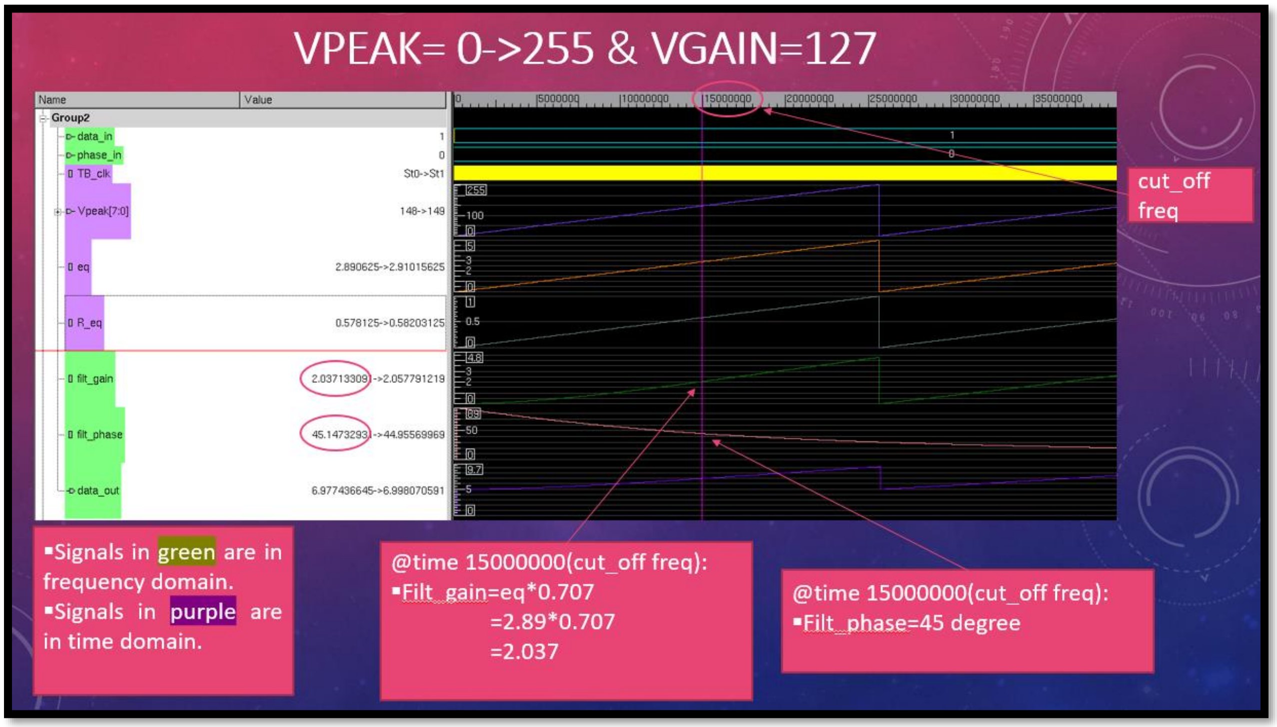
The following figure shows that when Vpeak varies from 0 to 255, we notice that the wave form of filt_gain and filt_phase correspond perfectly to the wave form of a high pass filter shown in figure 29. We also notice that at the cutoff frequency, which is 15Ghz, we have the output phase of the filter is 45 degrees and the output gain is calculated as follows:

At time 15000000(cut_off freq):

- Filt_gain=eq*0.707

  =2.89*0.707

  =2.037


VPEAK= 0->255 & VGAIN=127

```
Name                Value
Group2
  data_in           1
  phase_in          0
  TB_clk            St0->St1
  Vpeak[7:0]        148->149
  eq                2.890625->2.91015625
  R_eq              0.578125->0.58203125
  filt_gain         2.03713309->2.057791219
  filt_phase        45.1473293->44.95569969
  data_out          6.977436645->6.998070591
```

cut_off freq

▪Signals in green are in frequency domain.
▪Signals in purple are in time domain.

@time 15000000(cut_off freq):
▪Filt_gain=eq*0.707
=2.89*0.707
=2.037

@time 15000000(cut_off freq):
▪Filt_phase=45 degree

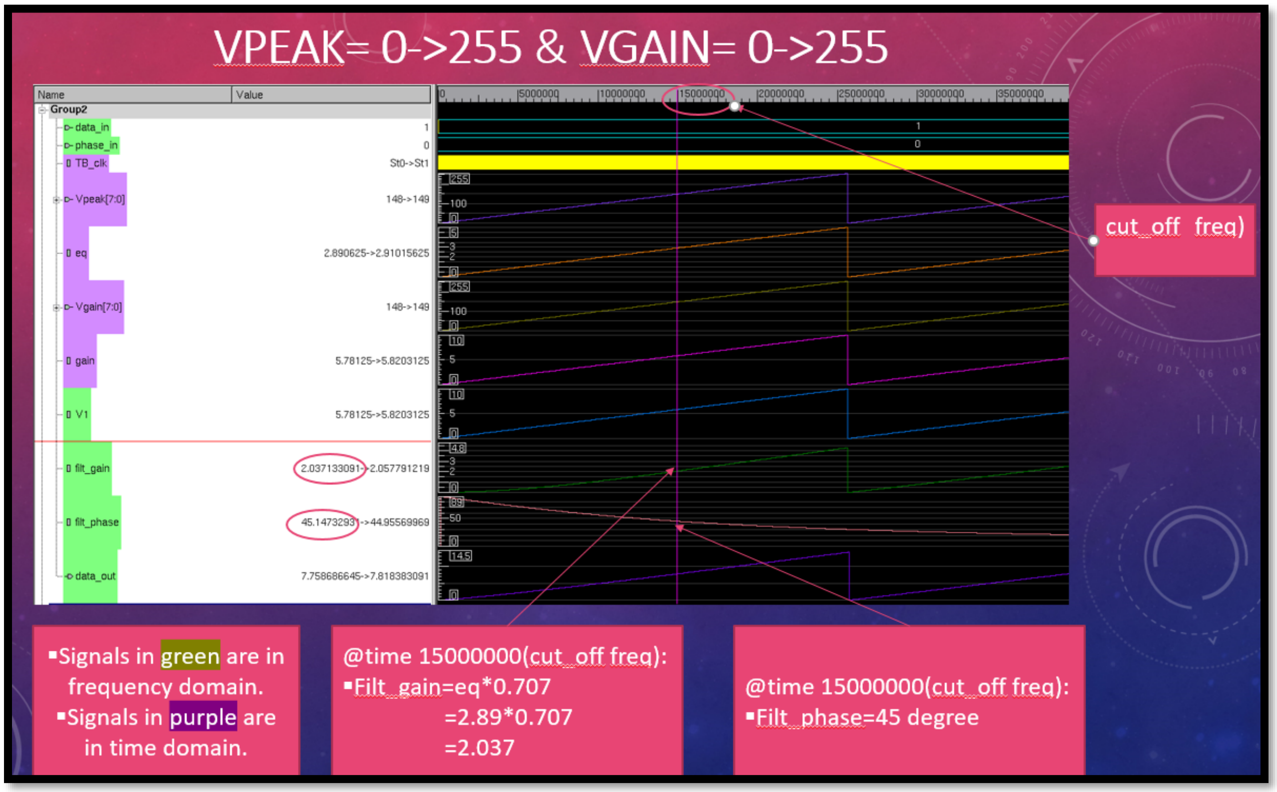
Figure 31: Equalizer's validation with VCS (2nd validation)

Even if our simulation tool (VCS) is purely digital, we have managed to perform a mixed simulation of digital signals (TB_clk, Vpeak, Eq) and analog signals (data_in, phase_in, filt_gain, filt_phase, data_out). This mixed simulation allowed us to validate the functioning of the equalizer.

6.3.Whole system validation

To validate the functioning of the whole system we designed a test bench in which we varied the input voltage Vgain from 0 to 255 and Vpeak from 0 to 255 because Vgain controls the amplifier and Vpeak controls the equalizer.

On each rising edge of the clock, the digital block will look to convert the digital values of Vgain and Vpeak into analog values so that they can be processed by the analog block. After converting the numerical values, the analog block reads the inputs of Data_in and Phase_in and it will sequentially perform the operations of the amplifier as well as that of the high pass filter. When complete, the output values will be written to a text file.



VPEAK= 0->255 & VGAIN= 0->255

Name	Value
Group2	
data_in	1
phase_in	0
TB_clk	St0->St1
Vpeak[7:0]	148->149
eq	2.890625->2.91015625
Vgain[7:0]	148->149
gain	5.78125->5.8203125
V1	5.78125->5.8203125
filt_gain	2.037133091->2.057791219
filt_phase	45.1473293->44.95569969
data_out	7.758686645->7.818383091

▪Signals in green are in frequency domain.
▪Signals in purple are in time domain.

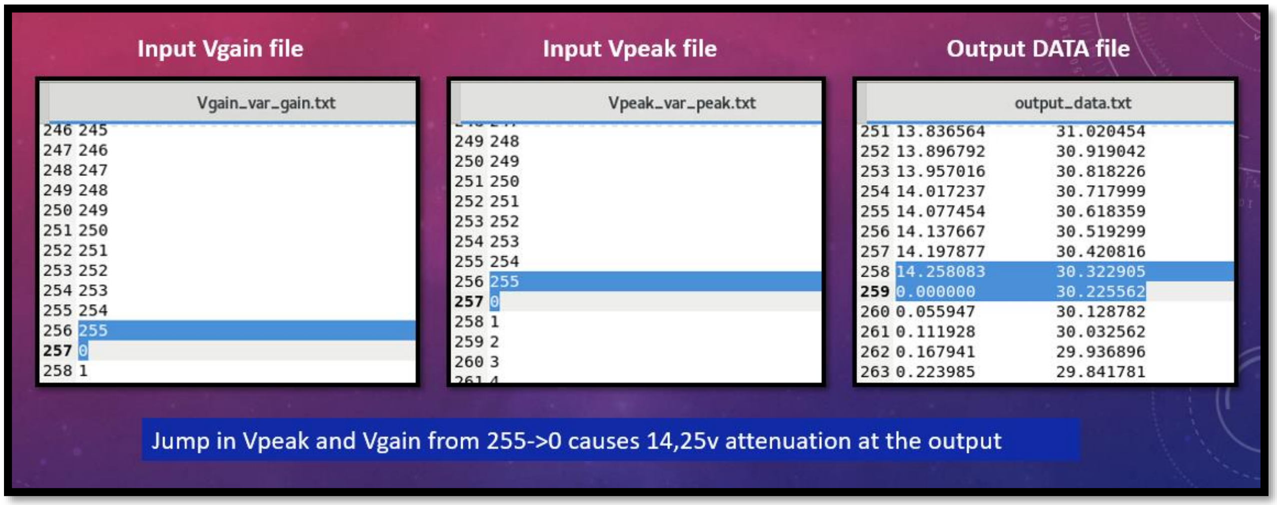
@time 15000000(cut_off freq):
▪Filt_gain=eq*0.707
=2.89*0.707
=2.037

@time 15000000(cut_off freq):
▪Filt_phase=45 degree

Figure 32: Whole system's validation with VCS (1^{ST} validation)

A first validation is made at the level of the behavior of the high pass filter. We notice here that even the two inputs of Vpeak and Vgain vary at the same time, the high pass filter retains its behavior and the validation of this is done through the verification of its gain and its phase at the cutoff frequency which is 15 Ghz. the VCS simulation of the figure above shows that at the cutoff frequency; the phase is 45 degree and the gain 2.037 (calculated detailed above).

A second validation is made at the level of the data synchronization between the results obtained with VCS and those in the output text file.


Input Vgain file
Input Vpeak file
Output DATA file
Jump in Vpeak and Vgain from 255->0 causes 14,25v attenuation at the output

Vgain_var_gain.txt

```
246 245
247 246
248 247
249 248
250 249
251 250
252 251
253 252
254 253
255 254
256 255
257 0
258 1
```

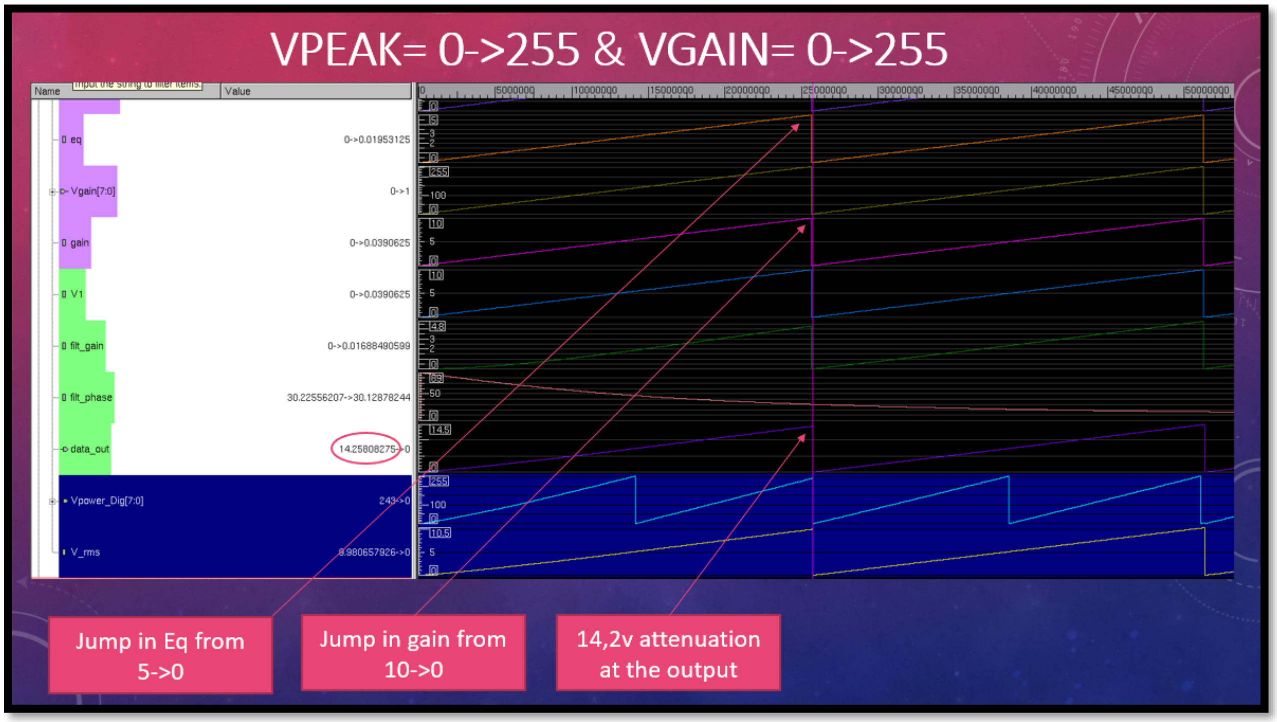
Vpeak_var_peak.txt

```
249 248
250 249
251 250
252 251
253 252
254 253
255 254
256 255
257 0
258 1
259 2
260 3
```

output_data.txt

```
251 13.836564     31.020454
252 13.896792     30.919042
253 13.957016     30.818226
254 14.017237     30.717999
255 14.077454     30.618359
256 14.137667     30.519299
257 14.197877     30.420816
258 14.258083     30.322905
259 0.000000      30.225562
260 0.055947      30.128782
261 0.111928      30.032562
262 0.167941      29.936896
263 0.223985      29.841781
```

Figure 33: Data out attenuation in the output data file


VPEAK= 0->255 & VGAIN= 0->255
Jump in Eq from 5->0
Jump in gain from 10->0
14,2v attenuation at the output

Name	Value
eq	0->0.01953125
Vgain[7:0]	0->1
gain	0->0.0390625
V1	0->0.0390625
filt_gain	0->0.01688490599
filt_phase	30.22556207->30.12878244
data_out	14.25808275->0
Vpower_Dig[7:0]	243->0
V_rms	9.980657926->0

Figure 34: Whole system's validation with VCS (2nd validation)

Indeed, the variation of Vgain from 0 to 255 will cause a variation of the gain of the amplifier from 0 to 10. Similarly, the variation of Vpeak from 0 to 255 will cause a variation of the gain of the equalizer from 0 to 5. We saw previously that when Vgain jumps from 255 to 0 results in a 10v attenuation in the output value and Vpeak's jump from 255 to 0 results in an attenuation of

4.2v, therefore when both inputs jump at the same time this will cause an attenuation of 14.2v in the output data. this can be validated by reading the data in the simulation with VCS (Figure 33) and the data written in the output text file (figure 32). We notice here that there is a perfect correspondence between the two data. Consequently, thanks to a mixed visualization of analog and digital signals, we were able to validate the overall behavior of our mixed system.

7. Conclusion

In this chapter we have shown that the validation of mixed circuits is possible with a digital tool and this through a system Verilog behavioral model that runs simultaneously under both time and frequency domains. We succeeded in monitoring the interaction between both parts (Digital and analog blocks) and we rapidly validated all frequency related outputs within frequency plots which considerably reduced the simulation time.

Conclusion and Future Perspectives

Today, the validation of mixed signal SoCs is a very important task given the great interaction between analog and digital circuits. Indeed, after having studied the different levels of simulation and the numerous solutions proposed by other researchers for validation with mixed signals, we proposed behavioral modeling approach of mixed system (analog / digital) simulated through an environment purely digital. The method described in chapter 4 provides an overview of the whole system behavior as well as the different interactions between analog and digital blocks and this in a very reduced simulation time compared to analog simulators (SPICE). Even if the validation is carried out in a digital event environment, it remains precise, fast and faithful to the characteristics of the circuits which interest the designers. In chapter 5 we presented an example of a mixed circuit that we validated with a purely numerical tool (VCS). Thanks to the mixed simulation we were able to validate the behavior of the entire system in record time.

In future work, integration of the framework within the standard Universal Verification Methodology (UVM) is unavoidable step as the UVM sis widely accepted and used in industry. This integration will mainly require re-organizing the System-Verilog code into an oriented objects hierarchy according the concepts and principals of UVM.

Bibliography

[1] Dominique Rodriguez, *Description et simulation mixte analogique-numérique: analyse de VHDL analogique, réalisation d'un simulateur mixte*, Joseph Fourier-Grenoble University, February 1994

[2] Sabrina Liao, Verilog Piecewise linear behavioral modeling for mixed-signal validation, Stanford University, Doctor of Philosophy report, May 2014

[3] Ken Kundert, *A Formal Top-Down Design Process for Mixed-Signal Circuits*, Designer's Guide Consulting, Inc

[4] Resve Saleh, Shyh-Jye Jou, A. Richard Newton, *MIXED-MODE SIMULATION AND ANALOG MULTILEVEL SIMULATION*, University of Illinois, University of California, 1994

[5] L.W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," Electronics Research Laboratory Rep. No. ERLM520, University of California, Berkeley, May 1975.

[6] A. R. Newton, A. Sangiovanni-Vincentelli, "Relaxation-based Circuit Simulation", IEEE Trans. on Elec. Dev., Vol. ED-30, No.9, Sept. 1983,pp.1184-1207.

[7] B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS – An MOS Timing Simulator," IEEE Trans. Circ. and Sys., Vol. 22, 1975, pp. 901-909.

[8] A. R. Newton, "The Simulation of Large-Scale Integrated Circuits", Ph.D. dissertation, University of California, Berkeley, ERL Memo. ERL-M78/52, July 1978.

[9] G. De Micheli, " New Algorithms for the Timing Analysis of MOS Circuits" Master Report, University of California, Berkeley, 1980.

[10] A. Sangiovanni-Vincentelli, L. K. Chen and L. O. Chua, "A New Tearing Approach-Node Tearing Nodal Analysis", International Symposium on Circuits and Systems, 1977, pp. 143-147.

[11] P. Yang, I. N. Hajj and T. N. Trick, "SLATE: A Circuit Simulation Program with Latency Exploitation and Node Tearing", International Conference on Circuits and Computers, October 1980.

[12] K. A. Sakallah, "Mixed Simulation of Electronic Integrated Circuits", Ph.D. dissertation, Carnegie-Mellon University, DRC-02-07-81, Nov. 1981.

[13] N. B. G. Rabbat, A. Sangiovanni-Vincentelli and H. Y. Hsieh, "A Multilevel Newton Algorithm with Macromodelling and Latency for the Analysis of Large-Scale Nonlinear Circuits in the Time Domain", IEEE Trans. on Circ. and Sys., Vol. CAS-26, Sept. 1979, pp. 733-741.

[14] M. A. Breuer and A. D. Friedman, **Diagnosis and Reliable Design of Digital Systems,** Computer Science Press, 1976.

[15] R. E. Bryant, "An Algorithm for MOS Logic Simulation", LAMBDA, 4th Quarter 1980, pp. 46-53.

[16] V. Rao, D. Overhauser, I. Hajj, T. Trick, Switch-level Timing Simulation of MOS VLSI Circuits, Kluwer Academic Publishers, Boston, MA., 1989.

[17] M. A. Breuer, Ed., **Digital System Design Automation: Languages, Simulation and Data Base,** Computer Science Press, 1975.

[18] D. Hill, "Multilevel Simulator for Computer-Aided Design", Ph.D. dissertation, Dept. of Elec. Eng., Stanford University, 1980.

[19] B. Infante, A. Sanders, E. Lock, "Hierarchical Modeling in a Multi-level Simulator", International Conference on Computer-Aided Design, Santa Clara, CA. 1984, pp. 39-41.

[20] A. Insinga, "Behavioral Modeling in a Structural Logic Simulator", International Conference on Computer-Aided Design, Santa Clara, CA. 1984, pp. 42-44.

[21] IEEE Computer Society, **IEEE Standard VHDL Language Reference Manual,** The Institute of Electrical and Electronic Engineering, New York, PUBL. NO: IEEE Standards Coordinating Committee 20, 1988.

[22] D. E. Thomas and Philip R. Moorby, The Verilog Hardware Description Language, Kluwer Academic Publishers, Boston, 1991.

[23] Rakesh Chadha, Chandramouli Visweswariah and Chin-Fu Chen, "M3-A Multilevel Mixed-Mode Mixed *ND* Simulator", IEEE Transactions on Computer-Aided Design, Vol. 11, No.5, May 1992, pp. 575- 585.

[24] G. Gielen, E. Liu, A. Sangiovanni-Vincentelli and P. Gray, "Analog Behavioral Models for Simulation and Synthesis of Mixed-Signal Systems", European Design Automation Conference, 1992, pp. 464-468.

[25] J. Singh and R. Saleh, "iMACSIM: A Program for Multi-Level Analog Circuit Simulation", International Conference on Computer-Aided Design, 1991, pp. 16-19.

[26] M. Rumsey and J. Sackett, "An ASIC Methodology for Mixed Analog-Digital Simulation", 26th ACM/IEEE Design Automation Conference, 1989, pp. 618-621.

[27] M. Vlach, "Modeling and Simulation with Saber", IEEE ASIC Design Conference, 1990, pp. T -11.1-T -11.9.

[28] M. J. Rewieński, "A Perspective on Fast-SPICE Simulation Technology," in *Simulation and Verification of Electronic and Biological Systems*, Netherlands: Springer, 2011, pp.23-42.

[29] M. Zwolinski, K. G. Nichols, A. D. Brown and M. Awan, “A Mixed-Mode Circuit Simulator,” in *Proc. UK IT Conference*, 1990, pp. 390-393.

[30] Qian, Kun. ‘*Variability modeling and statistical parameter extraction for CMOS devices*. PHD Dissertation. UC Berkeley, 2015.

[31] H. Fleurkens and P. Buurman, “Flexible Mixed-Mode and Mixed-Level Simulation,” in *Proc. Circuits and Systems, 1993 IEEE International Symposium on,* 1993, pp. 2137-2140.

[32] A. R. W. Todesco, and T. H.-Y. Meng, “Symphony: A Simulation Backplane for Parallel Mixed-Mode Co-Simulation of VLSI Systems,” in *Proc. DesignAutomation Conference, 33rd ACM Annual,* 1996, pp. 149-154.

[33] N. Abdallah, P. B. Sabet, and A. Greiner, “On the Design of Mixed-Mode Simulators for Modern VLSI Circuits,” in *Proc. Circuits and Systems, IEEE 38th Midwest Symposium on*, 1995, vol. 2, pp. 1168-1171.

[34] C.-J. R. Shi, “Mixed-Signal System-on-Chip Verification Using a Recursively- Verifying-Modeling (RVM) Methodology,” in *Proc. Circuits and Systems, 2010 IEEE International Symposium on,* 2010, pp. 1432-1435.

[35] C. Gu, “Algorithmic Nonlinear Macromodeling: Challenges, Solutions and Applications in Analog/Mixed-Signal Validation,” in *Proc. Custom Integrated Circuits Conference, 2013 IEEE,* 2013, pp. 1-8.

[36] G. Zheng, S. P. Mohanty, and E. Kougianos, “Design and Modeling of a Continuous-time Delta-Sigma Modulator for Biopotential Signal Acquisition: Simulink Vs. Verilog-AMS Perspective,” in *Proc. Computing Communication & Networking Technologies, 2012 IEEE 3rd International Conference on*, 2012, pp. 1- 6.

[37] Open SystemC Initiative (2010, Oct. 02), *SystemC AMS extension User’s Guide* [Online], Available: http://www.systemc.org/downloads/standards

[38] P. Gang, “Behavioral Modeling and Simulation of Analog/Mixed-Signal Systems Using Verilog-AMS,” in *Proc. Information, Computing and Telecommunication, 2009 IEEE Youth Conference on,* 2009, pp. 383-386.

[39] Y. Wang, C. Van-Meersbergen, H.-W. Groh, S. Heinen, “Event Driven Analog Modeling for the Verification of PLL Frequency Synthesizers,” in *Proc. Behavioral Modeling and Simulation Workshop, 2009 IEEE International,* 2009, pp. 25-30.

[40] X. Lai, Y. Zhang, Y. Li, and X. M. Liu, “Behavioral Modeling of Electronic Circuit Module with Verilog-A Language,” in *Proc. ASIC, 2001 IEEE 4th International Conference on,* 2001, pp. 155-158.

[41] A. Prodic, and D. Maksimovic, “On Behavioral Modeling of a Mixed-Signal Analog to Digital Converter,” in *Proc. Computers in Power Electronics, 2002 IEEE Workshop on,* 2002, pp. 100-105.

[42] D. Dumlugol, and D. Webber, “Analog Modeling Using Event-Driven HDL’s,” in *Proc. VLSI Design, 1994 IEEE 7th International Conference on,* 1994, pp. 53-56.

[43] T. J. Wen, and T. Kwasniewski, “Phase Noise Simulation and Modeling of ADPLL by SystemVerilog,” in *Proc. Behavioral Modeling and Simulation Workshop, 2008 IEEE International,* 2008, pp. 29-34.

[44] C. V. Kashyap, and C. S. Amin, “Raven: A Tool for Automatic Generation of Analog Behavioral Models from Schematics,” presented at *Frontiers in Analog Circuit Synthesis and Verification,* Snowbird, Utah, 2011.

[45] K. J. Kerns, M. Bhattacharya, S. Rudnaya, and K. Gullapalli, “Automatic, Hierarchy-Independent Partitioning Method for Transistor-Level Circuit Simulation,” U.S. Patent 8 060 355, Jan. 29, 2009 (pending).

[46] J.-E. Jang, M.-J. Park, D. Lee, and J. Kim, "True Event-Driven Simulation of Analog/Mixed-Signal Behaviors in SystemVerilog: A Decision-Feedback Equalizing (DFE) Receiver Example," in *Proc. Custom Integrated Circuits Conference, 2012 IEEE,* 2012, pp. 1-4.