Co-simulation Techniques for Mixed Signal Circuits

Tudor Timisescu

Technische Universität München

***Abstract* – As designs grow more and more complex, there is increasing effort spent on verification. Most bugs are found at the interactions between blocks, so it is absolutely necessary to start testing as soon as possible in the development cycle. Behavioral modeling is a technique which allows for higher simulation speeds, as well as early system level verification.**

***Index terms* – Functional verification, mixed signal, behavioral modeling.**

I. INTRODUCTION

The development of multi-million gate circuits poses increasing challenges due to ever more complex requirements (both physical and functional) to the need to integrate a larger and larger amount of proprietary blocks, IP modules and even software applications. To reach these ambitious goals and still meet time-to-market requirements, it is mandatory to not waste precious development time on re-spins, which prompts the need for efficient and effective verification practices able to find bugs early on in the design process, way before committing the chip to silicon.

II. FUNCTIONAL VERIFICATION

Functional verification is the task of proving that a design conforms to its specification and operates as required. Some statistics: it is estimated that around 71% of IC re-spins are due to functional bugs and about 47% are due to incomplete or incorrect specifications. Moreover, the effort required to verify a complex state of the art SoC takes at least

60 – 70% of the whole development time [1].

Pre-silicon development consists mostly of the interaction between four teams: architecture development is in charge of formulating specifications for the IC, partitioning the future system into components and working out the functional relations between them; the analog and digital design teams implement the architecture defined beforehand into analog and digital blocks realizable on silicon; finally, the verification team has the task of proving that the implementation of the chip was done resulted in a product conforming to the specifications. The basic interactions between these teams are summarized in Figure 1 - IC Development Flow.

Not only is the design of analog blocks very different from the design of digital blocks, but also the way verification is done. While analog circuits are validated to operate correctly in a wide variety of environmental conditions and subject to different process variations, digital circuits on the other hand are required to work under a wide variety of input vectors. The way verification is approached is also fundamentally different: analog blocks use traditional directed testing, while new digital blocks are verified using random constrained stimulus generation.

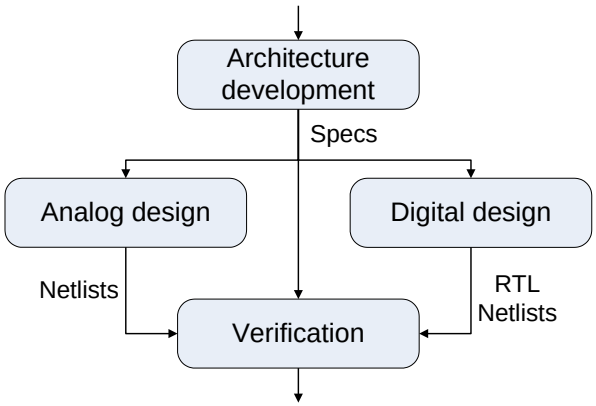

Figure 1 - IC Development Flow

A. Digital verification

Modern digital verification relies on using hardware verification languages (HVLs) like SystemVerilog or e. These allow for an object oriented approach at a higher level of abstraction, eliminating the need to think only at the signal level. Stimulus is generated randomly out of the allowed input space allowing it to reach corner cases a human test writer would probably omit. As such, the testbenches must implement automatic checking and there exist also clear metrics to evaluate the results by. While this undoubtedly increased the time spent on testbench development in the early stages of the verification process, the effort pays off in the end, as all tests share the verification environment as opposed to writing directed tests, where a new testbench must be built basically from scratch for each one [2]. Figure 2 - Directed vs. Random Constrained Verification shows a qualitative graph comparing the two methodologies in terms of “speed”.

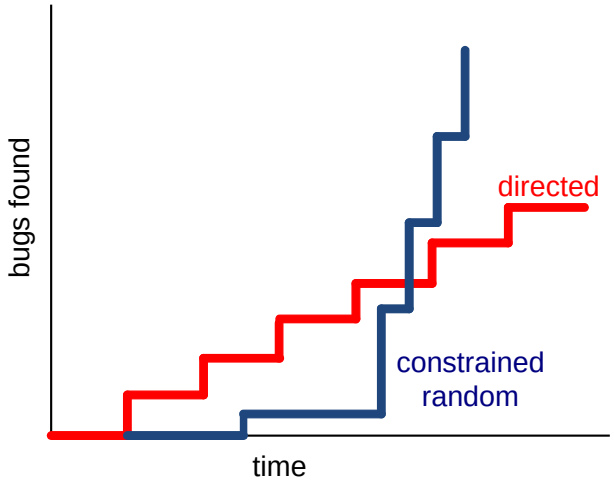

Figure 2 - Directed vs. Random Constrained Verification

The entire process is done according to a thorough verification plan based on the specification, which requires no white box knowledge of the circuit. There are different methodologies available such as the e Reuse Methodology (eRM), the Verification Methodology Manual (VMM), the Open Verification Methodology (OVM) and the new Universal Verification Methodology (UVM) which facilitate reuse of verification components and interoperability between different verification IP providers.

The tools on which digital verification is done are logic simulators, such as QuestaSim, VCS and Incisive Unified Simulator. These are event driven and offer high simulation speeds compared to analog simulations at the transistor level.

B. Analog verification

Analog designs still require thinking in terms of various physical quantities such as voltage, current, temperature, charge, etc. This leads to having to manually define input stimuli, which limits the number of different testcases that can be executed. Likewise, the need for visual inspection of waveforms and file dumps of the results decreases the number of test runs that can be performed while also being prone to human errors.

There is not really any notion of verification IP, so each circuit has its own testbench built basically from scratch and requires detailed knowledge of its inner workings.

The tools used in simulations are called SPICE simulators which solve the differential equations of the circuit. Some examples are Nanosim, Ultrasim and Questa-ADMS. These tools require large amounts of simulation time, but a trade-off can be made between this and the accuracy of the results.

III. MIXED SIGNAL SYSTEM SIMULATION

Today's designs are not purely analog or digital, but contain blocks that are either fully analog (voltage regulators, amplifiers, sensors, etc.), fully digital (registers, arithmetic blocks, FSMs, etc.) or contain a mixture of the two which is

functionally deeply intertwined (ADCs and DACs are the most eloquent examples). Such systems are called mixed signal systems (Figure 3 - Example of a Mixed Signal System shows a small sample of such a mixed signal system, where the yellow blocks are purely analog, the blue blocks represent digital logic and the green blocks are mixed analog-digital blocks).

It is not sufficient to simulate each component individually to achieve a working system, as most bugs are usually found at the interfaces and interactions between sub-blocks.

These types of systems must be simulated using analog tools and their verification inherits all the weaknesses of analog verification, mainly the "touch and feel approach" and the huge simulation times required. Besides these drawbacks, the biggest problem is that this verification step can only be done in a late stage of the development cycle, so finding too many bugs here could be fatal for the time-to-market.

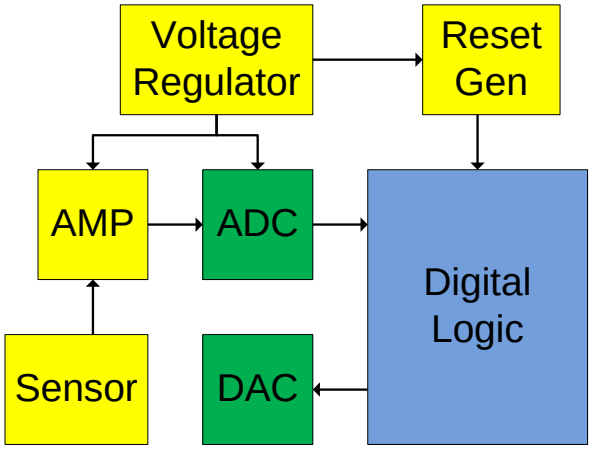

Figure 3 - Example of a Mixed Signal System

A. Analog model based verification

A possible solution to two of these problems, namely the large simulations times and the availability only very late in the development process, is using behavioral models in place of transistor based analog blocks. These models would only capture the interesting and essential behaviors of the blocks and not details such as variations with temperature, process, supply voltage, etc. Using them also removes the need to solve differential equation, giving the possibility of using logic simulators. Early system integration and testing becomes a reality as well and some of the advantages of digital verification become accessible (developed methodologies for the digital interfaces, reuse, etc.).

The languages of choice for modeling are pure behavioral VHDL (most widely used) or Verilog AMS and VHDL AMS. The latter are used for also implementing structural models of any type of analog system (even, for example, mechanical systems [3]), but with the downside of requiring more advanced simulators.

A testbench developed to use analog models can not only be used for early simulations, but also for late full system integration testing. Blocks can be replaced with their implemented counterparts as they become available, to do the final top level verification using

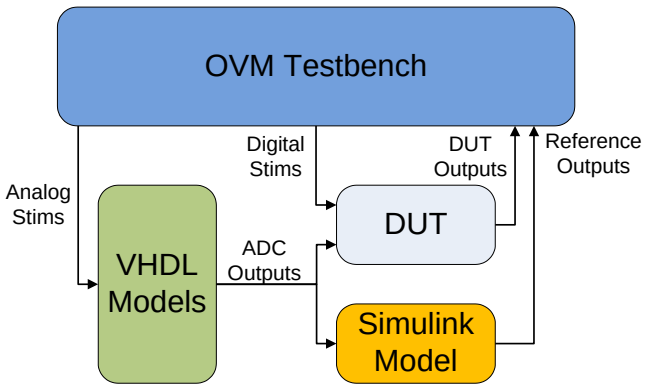

Figure 4 – D-BAP Sensor Verification Environment

only a smaller number of tests to achieve sign-off.

B. System level reference models

In the digital verification section, it was mentioned that the testbench must be self-checking. This requires the verification team to come up with a high level model for the inner workings of the system. Such a model should already exist, developed by the system level team as an executable specification for early architecture validation and to be provided to the customer to refine the specifications.

These models are usually written in plain C, SystemC, Simulink or maybe a mixture of all these and more. As they already capture the intended behavior of the IC, reusing this for verification purposes removes the need for double work and error prone translation from one language to another.

C and SystemC are supported by all major simulators (but the solutions are usually proprietary and cannot be moved from one tool to another). SystemC provides the additional bonuses of transaction level modeling support, integration of hardware and software components and the extensive methodologies already available (for example, the Open SystemC Initiative) [4].

Matlab and Simulink models must use separate products for co-simulation (such as the EDA Simulator Link). These provide interfaces only at signal level and the licenses for them usually cost a lot of money (not counting the license costs for Matlab, Simulink and any required toolboxes). Communication with the HDL/SPICE simulator can be done either via shared memory on the same machine or via a network.

IV. EXAMPLE

As a practical example to illustrate the concepts described in this paper, an overview of the verification environment for one of the new Digital Barometric Air Pressure Sensors from Infineon Technologies is given. A simplistic block diagram is given in Figure 4 – D-BAP Sensor Verification Environment.

The DUT contains the entire digital core of the IC (color white). The analog parts, such as the sensors, voltage regulator, reset generator, EEPROM, ADC, etc. are modeled using behavioral

VHDL (color green). A reference model for the pressure and temperature signal processing paths is available in the form of a Simulink model (color yellow). The OVM testbench (color blue) contains the bulk of the verification environment: test control, driving the inputs (both digital and analog), monitoring the interfaces for data transactions, modeling the rest of the digital core, and coverage collection. Analog stimuli are passed through the VHDL models and are converted to digital signals representing the ADC outputs. These are then sent to both the DUT and the Simulink model and their responses are compared.

V. CONCLUSIONS

It has been mentioned in this paper that the majority of bugs are found at the system level. Unfortunately, the need to have all these blocks ready before such simulations can be performed means that they can only be done late in the development cycle. A method to allow early testing is using behavioral blocks to speed up testbench development and the availability of a top level integration for the DUT.

Future trends include multi-language testbenches (such as the Universal Verification Methodology), using such testbenches for transistor level simulations and promoting analog VIP reuse.

REFERENCES

1. **Bonfini, G., Chiavacci, M. and Pescari, E.** Verification of Mixed-Signal Systems. [Online] http://www.embeddedstar.com/articles/2005/11/article20051107-1.html.

2. **Spear, Chris.** *SystemVerilog for Verification - A Guide to Learning the Testbench Language Features.* s.l. : Springer, 2008.

3. **Watkins, Steve and Wong, Kam.** Hardware Design for Software People in the SoC Era. *Mixed Signal Modeling with Vanilla VHDL and Verilog* . [Online] http://www.bluepc.com/mixpap.html.

4. **Open SystemC Initiative.** Open SystemC Initiative - About SystemC. *Open SystemC Initiative.* [Online] http://www.systemc.org/community/about_systemc/.