Eine Methode zur Verifikation von Mixed-Signal-ASIC

Anselm Breitenreiter*, Jesús López†, Pedro Reviriego‡, Daniel González† und Milos Krstic*
*IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany
{breitenreiter|krstic}@ihp-microelectronics.com
†Arquimea Ingenieria S.L.U., Leganés, Spain
{jlopez|dgonzalez}@arquimea.com
‡Escuela Politécnica Superior, Universidad Antonio de Nebrija, Madrid E-28040, Spain
previrie@nebrija.es

***Zusammenfassung*—Es wird eine Verifikationsmethode für Mixed-Signal-ASIC vorgestellt, die den verschiedenen Abstraktionsebenen im Entwicklungsprozess gerecht wird. Durch eine modulare Testbench in SystemVerilog und Cosimulation mit MATLAB wird umfassende Systemverifikation bei einfacher Handhabung und geringer Simulationslaufzeit gewährleistet.**

I. Einleitung

Mit steigender Komplexität von Mixed-Signal-ASIC wird deren Entwicklung stetig anspruchsvoller. Zeitgemäße Methoden des Systementwurfs umfassen mehrere Abstraktionsebenen, angefangen bei der Spezifikation, bis hin zur taktgenauen Darstellung. Das Besondere am Mixed-Signal-Entwurf ist, dass alle Abstraktionsebenen sowohl dem digitalen als auch dem analogen Verhalten gerecht werden müssen. Die größte Herausforderung stellt hierbei die Verifikation dar, welche einerseits vertikal integriert sein soll, d.h. das verschiedene Abstraktionsebenen nahtlos miteinander verbunden sind und Verifikation auch ebenenübergreifend möglich ist. Andererseits spielt die Simulationslaufzeit eine Rolle, insbesondere Analog-Mixed-Signal-Simulationen (AMS) die sehr tiefe Ebenen einschließen sind oft zu zeitintensiv. Besondere Aufmerksamkeit sollte den Schnittstellen zwischen der analogen und digitalen Welt gelten um sicherzustellen, dass sie auch auf hohen Abstraktionsebenen korrekt repräsentiert sind.

Im Folgenden wird eine Methode zur Verifikation vorgestellt, die die verschiedenen Abstraktionsebenen widergespiegelt und das analoge Verhalten durch direktes Einbinden von high-level MATLAB-Modellen abbildet. Dabei wird umfassende Verifikation bei geringer Laufzeit im Vergleich zur AMS-Simulation erreicht. Zudem wird eine einheitliche Schnittstelle zur Testspezifikation und -datengenerierung geschaffen in die die Systemanforderungen unabhängig von der adressierten Abstraktionsebene übertragen werden können.

Die Grundlage bildet eine Testbench-Architektur nach [1], die in SystemVerilog [2] implementiert wurde. Sie ist vielfach erprobt [3], [4] und liegt auch umfangreichen Frameworks wie VMM [5] oder UVM [6] zugrunde, welche geringen Wartungsaufwand und hohe Wiederverwendbarkeit versprechen. Da diese Frameworks allerdings mit höherem initialen Aufwand verbunden sind und um volle Flexibilität zu wahren, wird hier nicht darauf zurückgegriffen. Ein zur vorgestellten Methode vergleichbarer Ansatz ist in [7] beschrieben. Im Gegensatz dazu sind hier die Schnittstellen zwischen Prüfling und Testbench jedoch rein digital, sodass die Testbench unabhängig vom analogen Verhalten ist. Desweiteren soll hier näher auf Aspekte der Cosimulation und Testkonfiguration eingegangen werden.

In Abschnitt II wird die Verifikationsmethode im Allgemeinen vorgestellt, woraufhin in Abschnitt III die Umsetzung am Beispiel einer Ethernet PHY vorgeführt und Testergebnisse präsentiert werden. Eine abschließende Betrachtung findet in Abschnitt IV statt.

II. Methode zur Verifikation von Mixed-Signal-ASIC

Die Testbench stützt sich auf ein modulares Konzept, dass durch objektorientierte Programmierung in SystemVerilog direkt umgesetzt werden kann. Alle Konfigurationsparameter werden aus einer zentralen Testspezifikation abgeleitet, welche ebenso die Vorgaben für eine zufällige Erzeugung von Testdaten liefert. Anstatt den digitalen Schaltungsteil allein zu simulieren werden die MATLAB-Modelle durch Cosimulation zur Laufzeit evaluiert.

A. Testbench-Architektur

Abbildung 1 zeigt die Struktur der Testbench. Auf unterster Ebene, der *Signal Layer*, befindet sich der Prüfling. Dessen Schnittstellen werden auf Bit- und Taktebene aus der nächsthöheren Ebene, der *Command Layer*, stimuliert und gelesen. Darüber bildet die *Functional Layer* die erste Abstraktion. Hier werden Sequenzen über mehrere Takte und Bits zu Transaktionen zusammengefasst und es findet der Vergleich von simuliertem und erwartetem Wert statt. In der darauffolgenden *Scenario Layer* wird ein gesamter Testlauf betrachtet, beispielsweise bestehend aus dem Setzen einer Gerätekonfiguration und der anschließenden Übertragung einer Reihe von Datenframes mit bestimmten Eigenschaften.

B. Testspezifikation und Generierung von Testdaten

Die Testspezifikation gibt den Rahmen für den gesamten Testablauf vor. Wo sie auf hoher Abstraktionsebene beispielsweise die Anzahl der zu testenden Pakete und deren mögliche

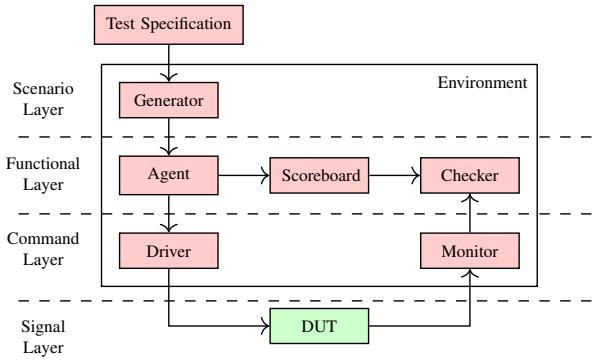

Abbildung 1. Die Testbench-Architektur zur Verifikation digitaler Systeme nach [1].

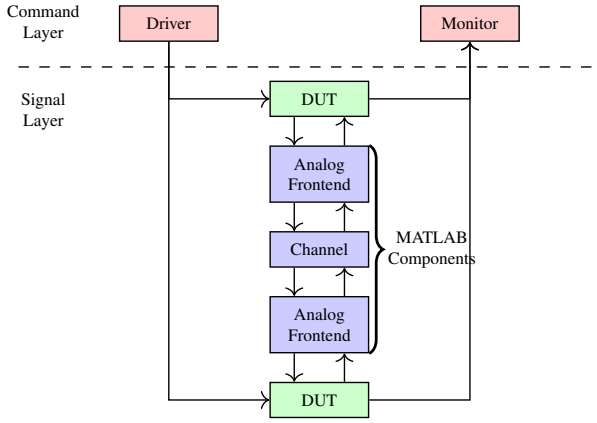

Abbildung 2. Erweiterung der *Signal Layer* um Modelle analoger Komponenten. Beispielhaft wird hier eine gesamte Übertragungsstrecke in die Testbench eingebunden.

Länge definiert kann sie ebenso Grenzen für den Jitter einzelner Signale setzen und die analogen Modelle parametrisieren. Dazu wird sie durch die gesamte Testbench propagiert und jedes Testbenchmodul wertet die jeweils benötigten Datenfelder aus. Zur Generierung der zufälligen Testdaten werden SystemVerilogs *Random Constraints* verwendet.

C. Erweiterung um analoge Komponenten

Um die Modelle der analogen Komponenten in die Verifikation zu integrieren wird die *Signal Layer* erweitert. Abbildung 2 zeigt die Verifikation einer gesamten Übertragungsstrecke mittels beschriebener Testbench-Architektur. Für die Integration der MATLAB-Komponenten wurden zwei Optionen untersucht:

1) Cosimulation über einen TCP-Socket: Bei diesem Ansatz wird die Berechnung der analogen Funktionen zur Laufzeit in MATLAB durchgeführt, der Datenaustausch mit dem HDL-Simulator findet über einen TCP-Socket statt. Der Vorteil dieses Ansatzes ist, dass der volle Funktionsumfang von MATLAB zur Verfügung steht und MATLAB beispielsweise für die visuelle Darstellung der analogen Signale genutzt werden kann. Fehlersuche und Änderungen des MATLAB-Codes können direkt vorgenommen werden.

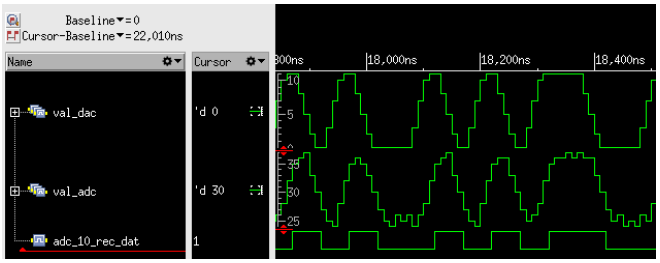

Abbildung 3. Gezeigt sind die simulierten Signale der Ethernet PHY. Dargestellt sind das digitale Signal am Eingang des analogen Frontends auf Senderseite, sowie das Signal am Eingang des Digitalteils auf Empfängerseite und schließlich das rekonstruierte binäre Signal.

2) Codegenerierung aus MATLAB: Eine weitere Möglichkeit besteht darin, aus MATLAB-Funktionen C-Code zu erzeugen, der aus SystemVerilog über das *Direct Programming Interface* (DPI) aufgerufen werden kann. Die Berechnungen sind so schneller, allerdings muss für jede Änderung der C-Code neu erzeugt und kompiliert werden. Da die Entwicklung der digitalen, analogen und Testbench-Komponenten meist einen iterativen Prozess darstellt, erweist sich dieser Ansatz als zu zeitintensiv, solange die Entwicklung noch nicht abgeschlossen ist. Die höhere Portabilität bietet allerdings Vorteile sobald die Testumgebung Dritten zur Verfügung gestellt werden soll.

III. Testergebnisse am Beispiel der Ethernet PHY

Als Beispiel soll die Anwendung der beschriebenen Verifikationsmethode auf eine Ethernet PHY präsentiert werden. Diese besteht aus einem Digitalteil und einem analogen Frontend. Zwei PHYs sind über einen analogen Kanal miteinander verbunden. Abbildung 3 zeigt simulierte Signalverläufe.

IV. Fazit

Es wurde eine Methode zur Verifikation von Mixed-Signal-ASIC vorgestellt, die aufgrund ihres modularen mehrschichtigen Ansatzes Testvorgaben auf unterschiedlichen Abstraktionsebenen direkt erfassen kann. Dabei können diese leicht erweitert oder geändert werden, ohne Änderungen an der Testbench selbst vornehmen zu müssen. Weiterführend soll die Methode um eine einheitliche Unterstützung für *SystemVerilog Assertions* sowie *Functional Coverage* ergänzt werden, die den mehrschichtigen Ansatz abbildet, damit sowohl der Systemzustand verfolgt, als auch die Testabdeckung auf jeder Abstraktionsebene überwacht wird.

Danksagung

Diese Veröffentlichung entstand im Rahmen des SEPHY Projekts als Teil des Forschungsprogramms Horizon 2020 der Europäischen Union, Grant Agreement No. 64024.

Literatur

[1] C. Spear, *SystemVerilog for Verification, Second Edition: A Guide to Learning the Testbench Language Features*, 2nd ed. Springer Publishing Company, Incorporated, 2008.

[2] "Ieee standard for systemverilog–unified hardware design, specification, and verification language," *IEEE Std 1800-2012 (Revision of IEEE Std 1800-2009)*, pp. 1–1315, Feb 2013.
[3] Y. J. Oh and G.-Y. Song, "Simple hardware verification platform using systemverilog," in *TENCON 2011 - 2011 IEEE Region 10 Conference*, Nov 2011, pp. 1414–1417.
[4] Z. Zhou, Z. Xie, X. Wang, and T. Wang, "Development of verification envioronment for spi master interface using systemverilog," in *Signal Processing (ICSP), 2012 IEEE 11th International Conference on*, vol. 3, Oct 2012, pp. 2188–2192.
[5] J. Bergeron, E. Cerny, A. Hunter, and A. Nightingale, *Verification Methodology Manual for SystemVerilog*. Springer US, 2005.
[6] *Universal Verification Methodology (UVM) 1.2 Class Reference*, Accellera Systems Initiative (Accellera), 2014.
[7] X. Yang, X. Niu, J. Fan, and C. Choi, "Mixed-signal system-on-a-chip (soc) verification based on systemverilog model," in *System Theory (SSST), 2013 45th Southeastern Symposium on*, March 2013, pp. 17–21.