Verification of Mixed Signal System Design accelerated by a new Diagnostic Tool-Kit

D. Dammers, F. Tissafi Drissi, M. Giroud, D. Schollän, L. M. Voßkämper
mems@dolphin-integration.com
Dolphin Integration / SAXO
Bismarckstr. 142a, 47057, Duisburg

Abstract

Time-to-market is still required to decrease in state-of-the-art design processes. Reliable verification of complete mixed-signal systems, i.e. logic and mixed signal electronics and systems with their attached peripherals, such as sensors and actuators, is needed to be performed quickly to fit in the enhanced design process. While the verification of the logic part, mostly implemented in Verilog and VHDL, has gained a speed increase through the use of Accellera's Property Specification Language (PSL), the analog part suffers from not being supported by this language. To speed up the verification of the analog part (SPICE, Verilog-A(MS), VHDL-AMS), innovation is needed. Based on the PSL principle, this paper presents an innovative solution for reliable and efficient validation of mixed-signal circuit design which commonly makes use of multiple languages for modeling. As demonstrator, the electrical validation of an embedded memory design is used as starting-point.

Keywords: mixed-signal design rule check, verification rule check, detectors, specification, assertion.

1 Introduction

In general, the standardized Property Specification Language (PSL, IEEE 1850) is often used in HDL development design flows for digital hardware design verification. VHDL and Verilog model source code is enriched by the PSL language instructions, as comments in the initial approach, integrated into the languages in upcoming revisions of the standards, or as verification units associated with the models. These instructions allow performing dedicated specification rule checks during simulation. Unfortunately, the PSL language is restricted to event driven aspects.

However, decreased production cycles of mixed-signal systems heavily call for advanced mechanisms in the verification of complete system simulations, i.e. to additionally take into account analog and mixed-signal aspects. This paper describes a new solution to automatically check complete mixed-signal designs using a library of detectors implemented in VHDL-AMS. The application example shows the electrical and timing verification of memory write cycles with help of voltage detectors in a mixed-signal circuit and a tighten integration in the design flow.

The new solution and the methodology[1] behind it are described in chapter "The Observer Tool-Kit". In chapter "The Demonstrator", a typical application of the solution is presented. Inside view of a detector is shown in chapter “Implementation” exemplarily on a voltage-detector. "Conclusion" summarizes the advantages of the proposed methodology for mixed signal circuit design. And, in chapter "Outlook", we explain in which fields the observer solution brings benefits to mixed-signal circuit designers and system integrators.

2 The Observer Tool-Kit

The vision that led to the tool-kit implementation was to improve design reliability and security through automated design verifications, thereby bypassing tedious and time-consuming manual waveform analysis. It consists of a VHDL-AMS detector library and a graphical user interface to set up the detectors and add them to the testbench.

Detectors are built to observe specific system characteristics, e.g. to measure currents, voltages, frequencies, slopes, delays, jitters etc., on the condition not to influence the system behavior in the simulation. So the detectors are passive "observers" with respect to the circuit. But they are active with respect to the designer: the detectors announce online, during simulation, when signals violate specification rules and write these events in report files for further analysis. Depending on the severity level set up for the alerts, the simulation can be aborted, paused or continued.

While implementing the detectors, special focus was set on being compatible with any possible applications, for instance through parameterization in order to be adaptable to different specifications. Besides the analog ports for attaching the detectors to the analog/mixed signal circuit nets to be observed, every detector has a logic input port to attach an enable signal and a logic output port to supply a trigger signal to the system. So, the user is able to build more complex specification rule checkers with help of the basic detectors already defined in the library.

The standardized hardware description language VHDL-AMS was chosen to build the library of detectors. On one hand the standard language ensures that designers using diverse simulators supporting multiple languages can benefit from these developments assembled using this library. On the other hand VHDL-AMS offers “assertion statements” to report misbehavior and control of the simulator.

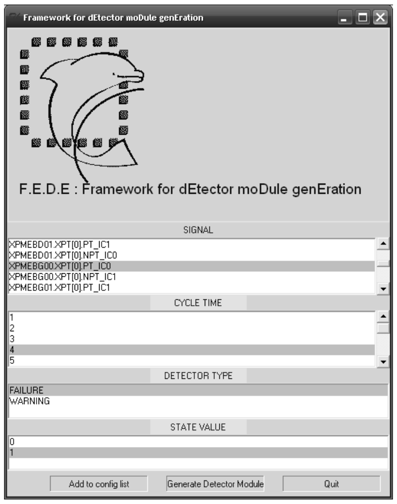
The advantage of using a library of detectors, rather than of using a special language, is that already verified model libraries remain untouched. Detectors can easily be placed inside the schematics of the design to verify and, of course, compounded specification rule checkers can be reused independently from any device/circuit model.

[1] The Detector methodology described in the following was developed and partly co-financed in the frame of the EU supported, regional funded project EMSIG (Development and transfer platform for the industrialization of mixed-signal circuits, FKZ 005-0604-0020). European Funds for Regional and Structural Development, regional project Ziel 2 Gebiet

3 The Demonstrator

To demonstrate the advantages of this new methodology, we apply voltage detectors on a mixed-signal SRAM (Static Random Access Memory) design for electrical validation of write operations at transistor level. Note: Of course, observing read operations is also possible, but for the demonstration of the advantages write operation check is sufficient. The detectors have to check if the voltage of each bit cell in the memory represents the logic value "1" or "0" at a specific time corresponding to the data input.

The observer toolkit offers two possibilities to integrate the detectors in the testbench. The first is a post generation and the more convenient, since it allows adding a detector network easily to an existing circuit without modifying it. In other words: At first the testbench has to be created. Then the preparation of the list of schematic signals is necessary, since the graphical user interface (GUI) uses this list to generate and set up the detector circuit. Figure 1 shows, the GUI with the available signals of the testbench in the upper pane. Here the designer is able to choose the signals to observe, such as the clock signal, the bit cell voltages, etc. The "CYCLE TIME" pane allows setting the number of read/write cycles. The "DETECTOR TYPE" pane allows setting the type of message level "Failure" or "Warning". Depending on this, the simulation will be continued, paused or aborted. The last setting is the "STATE VALUE", which allows setting the observation if the signal goes above or below the specification. The left button adds the selected configuration to the detector circuit setup. The middle button finally generates the detector circuit. The right button is used to close the GUI. In summary: the GUI avoids error prone manual detector circuit creation through generation via a few mouse-clicks. Currently, the GUI-option is restricted to be used with the simulator SMASH[2].


Framework for dEtector moDule genEration
F.E.D.E : Framework for dEtector moDule genEration
Add to config list
Generate Detector Module
Quit

SIGNAL
XPMEBD01.XPT[0].PT_IC1
XPMEBD01.XPT[0].NPT_IC0
[✓ XPMEBG00.XPT[0].PT_IC0]
XPMEBG00.XPT[0].NPT_IC1
XPMEBG01.XPT[0].PT_IC1

CYCLE TIME
1
2
3
[✓ 4]
5

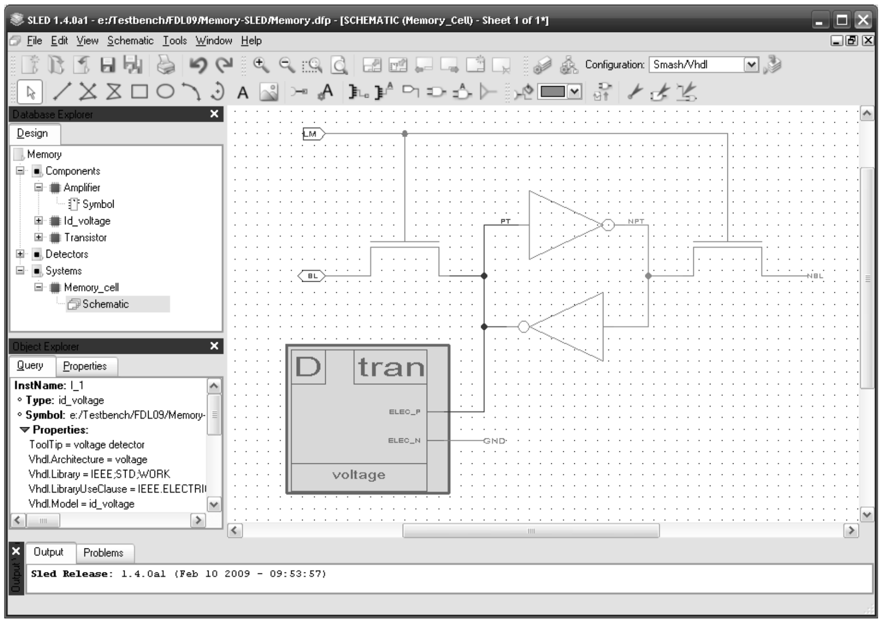
DETECTOR TYPE
[✓ FAILURE]
WARNING

STATE VALUE
0
[✓ 1]

Figure 1: Detector integration GUI

[2] Single kernel, mixed-signal and multi-language simulator by Dolphin Integration

The second possibility is the common way to add the detector directly through connecting the detector symbol to the device to observe via a schematic editor in the test-bench schematic. The screenshot of Figure 2 shows the bit cell schematic with attached voltage detector in the schematic editor SLED[3]. Needless to say that both options lead to the same simulation results.



Configuration: Smash/Vhdl

```
InstName: I_1
° Type: id_voltage
° Symbol: e:/Testbench/FDL09/Memory-
▼ Properties:
  ToolTip = voltage detector
  Vhdl.Architecture = voltage
  Vhdl.Library = IEEE;STD;WORK
  Vhdl.LibraryUseClause = IEEE.ELECTRI
  Vhdl.Model = id_voltage
```

```
Sled Release: 1.4.0a1 (Feb 10 2009 - 09:53:57)
```

Figure 2: Detector integration, schematic editor SLED

In Figure 3, the functionality diagram of the bit cell is shown. The clock signal CK and the DATA INPUT are digital signals, whereas PT is an analog signal and represents the boolean value of the memory cell, where V > VDD/2 is "true" and V < VDD/2 is "false", whereby VDD is the supply voltage. It has to be checked if the value of PT has the proper value at the end of a write cycle, i.e. at t1 the write operation starts and at t2 the write operation has to be finished; at t2 the detector is triggered to check if PT is above VDD/2. The detector can abort, pause or continue the simulation, depending on the severity level of the alerts, if it detects a wrong value at the end of the write cycle.

The complete testbench of this design contains an electrical modeling of a memory instance. The size of memory is 64 Kbit. We use 8 bit cells to perform the electrical validation of the complete testbench of this design which represents an electrical modeling of a memory instance.

By comparison to the conventional simulation, the fault free simulation with instantiated detectors needs 14.26% more time for one complete verification run on an Intel P4 3GHz Windows PC with 1GB SDRAM using the simulator SMASH release 5.12.1.

[3] **S**chematic **L**ink **ED**itor, EDA tool of Dolphin Integration http://www.dolphin-integration.com

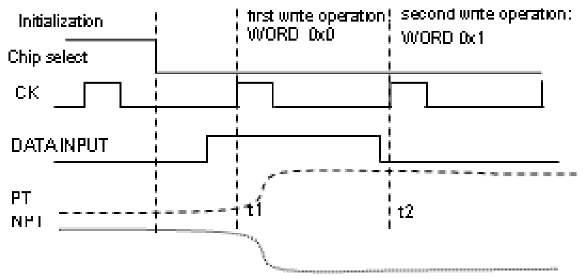
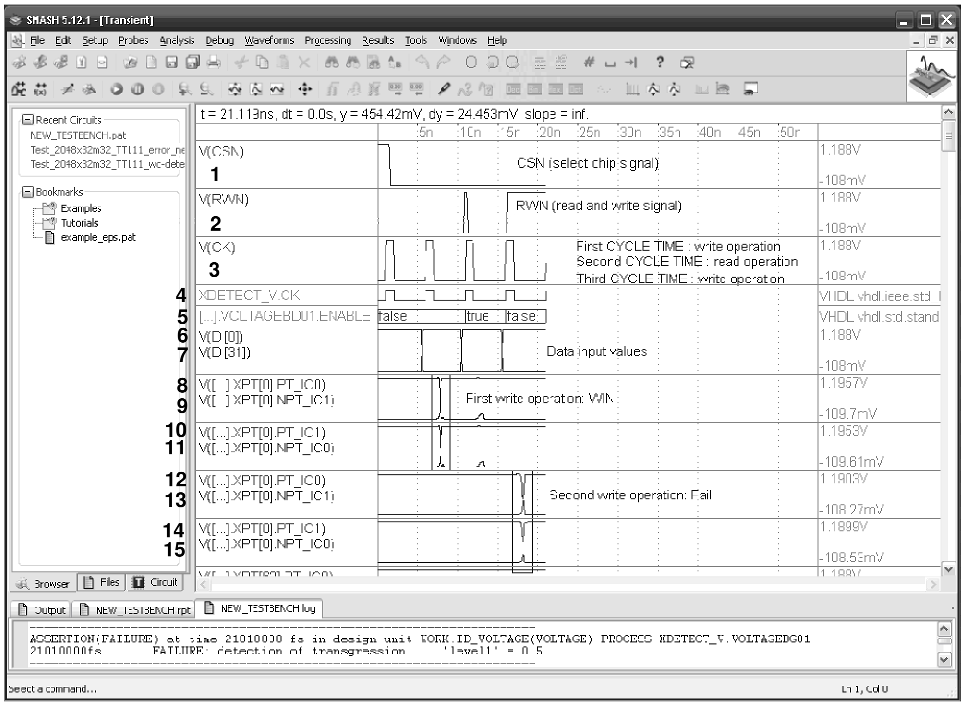


Figure 3: Bit cell functionality

The actual simulated signals are displayed in the transient simulation window online while the simulation is in progress. Figure 4 shows the simulation of the memory design in the simulator SMASH. The first write cycle after initialization is correctly done (see traces 8 to 11). On the second write cycle, a failure occurs (see traces 12 to 15). The detector reliably detects the misbehavior and aborts the simulation immediately as defined previously. At the same time, the error is announced on the screen at the lower pane of the simulator and logged separately for later off-line analysis in the detectors report file (see Listing 1).


SMASH 5.12.1 - [Transient]
File Edit Setup Probes Analysis Debug Waveforms Processing Results Tools Windows Help
t = 21.113ns, dt = 0.0s, y = 454.42mV, dy = 24.453mV slope = inf.
V(CSN) 1
CSN (select chip signal)
V(RWN) 2
RWN (read and write signal)
V(CK) 3
First CYCLE TIME : write operation
Second CYCLE TIME : read operation
Third CYCLE TIME : write operation
4 XDETECT_V.CK
5 [...].VOLTAGEBDU1.ENABLE
false true false
6 V(D[0])
7 V(D[31])
Data input values
8 V([...].XPT[0].PT_IC0)
9 V([...].XPT[0].NPT_IC1)
First write operation: WIN
10 V([...].XPT[0].PT_IC1)
11 V([...].XPT[0].NPT_IC0)
12 V([...].XPT[0].PT_IC0)
13 V([...].XPT[0].NPT_IC1)
Second write operation: Fail
14 V([...].XPT[0].PT_IC1)
15 V([...].XPT[0].NPT_IC0)
Browser Files Circuit
Output NEW_TESTBENCH.rpt NEW_TESTBENCH.log
Select a command...

```
--------------------------------------------------------------------------------
ASSERTION(FAILURE) at time 21010000 fs in design unit WORK.ID_VOLTAGE(VOLTAGE) PROCESS XDETECT_V.VOLTAGEDG01
21010000fs       FAILURE: detection of transgression     'level1' = 0.5
--------------------------------------------------------------------------------
```

Figure 4: Transient simulation of an erroneous write cycle

The first trace of the simulation shows the chip select signal CSN, which is “1” during the initialization cycle. Trace 2 shows the RWN signal, which is “1” for read operation and “0” for write operation. The analog clock signal and the digital clock signal are shown in trace 3 and 4. As example of a detector enable signal, the enable signal of the bit cell 2 is shown in trace 5. V(DI[0]) (trace 6) shows the data input value for the bit cells XPMEBD00.XPT[0] (trace 8) and XPMEBG00.XPT[0] (trace 12), whereas the data input value for the bit cells XPMEBD01.XPT[0].PT_IC1 (trace 10) and XPMEBG01.XPT[0].PT_IC1 (trace 14) is shown in V(DI[31]) (trace 7).

A second testbench is used to demonstrate a detector alert where the write operation fails on the second write operation. The fault simulation lasts 2 minutes, 46 seconds and 593 ms CPU time from start to abortion on the above mentioned PC using the same release of SMASH. Compared to the simulation over the full period of eleven cycles, the time saved for checking the design is 12 minutes, 18 seconds and 969 ms. Since it is difficult to estimate the time used by the designer to observe all signals, this time is neglected, but not forgotten in this consideration. But, significantly more important than the saved time, detectors, in opposite to a tired designer, always detect unfailingly the fault!
In the lower pane of the simulator (Figure 4), the detector alert of a specification violation caused by bit cell XPMEBG01.XPT[0] is shown (trace 14). The corresponding log-file is shown in Listing 1.

```
** Detector Message Report **
21010000fs  FAILURE: transgression at XPMEBG01.XPT[0] 'level1'=0.5
```

Listing 1: Detector log-file

The configurable detector messages allow easy and reliable location of faulty circuit elements. For further analysis or in order to document the simulation runs, the designer has the possibility to enable the detector log-file, where it is also possible to configure the log-text. The log-files are simple text files in order to be accessible from any other application. This means that detectors facilitate the observation of important signals in the circuit, so that design productivity/security is noticeably increased.

4 Implementation

The GUI was developed in Perl/Tk. To use it no further preparations are necessary then the generation of the list of schematic signals. The shown voltage detector is one piece of the Detector library implemented in VHDL-AMS. The advantage of this library, in opposite to add assertion checks every time again in new “device-model” source code, is the reusability. We use separate models for detectors and since the detector models possess an enable input for activation and a trigger output they can be assembled to build more complex detectors. Currently, the library comprises detectors to observe voltage, current, power, resistance, conductance, voltage at a specific time, current at a specific time, delay, frequency, frequency jitter, ratio, slew rate, rise time, fall time and generic edge. These manifold detectors make use of VHDL assertion statements to announce their alerts and report them.

Special care was set to not influence the systems behavior while implementing the detectors as demonstrated in Listing 2, where a part of the implementation of the voltage detector in VHDL-AMS is shown. One can see that the model has two electrical terminals, "elec_p" and "elec_n", where only the across quantity i.e. the voltage v_test, is used to detect if v_test is above a limit, see "PROCESS (v_test'ABOVE(level1), v_test'ABOVE(level2),Enable) IS". I.e. the quantities are used for "measurement"; no manipulation will be performed. PROCESS(transgression) generates the assertion message, as well as the report file. Depending on the severity level, the simulation will be continued, paused or aborted, see line "message_type : SEVERITY_LEVEL := ERROR".

```
7.     ENTITY id_voltage IS
8.      GENERIC level1 : VOLTAGE := 0.0; level2 : VOLTAGE := 0.0;
9.        detector_type : DETECTION_TYPE := Range_in;
10.       message_type  : SEVERITY_LEVEL := ERROR,
11.       transgression_message: STRING := " V out of range";
12.       rearranged_message   : STRING  := "V back in range";
13.       file_name : STRING    := "V-observer.rpt ");
14.     PORT (TERMINAL elec_p, elec_n:  ELECTRICAL;
15.       SIGNAL   Enable:  IN BOOLEAN; SIGNAL   Trigger: OUT BOOLEAN);
16.   END id_voltage;
17.   ARCHITECTURE voltage OF id_voltage IS
18.     QUANTITY v_test ACROSS elec_p TO elec_n;
19.     SIGNAL   transgression : BOOLEAN := FALSE;
20.     SIGNAL   msg_type      : STRING(1 TO 10)  := " WARNING: "; BEGIN
21.   PROCESS(v_test'ABOVE(level1),v_test'ABOVE(level2),Enable) IS BEGIN
22.     IF Enable THEN
23.       IF level1<level2 THEN
24.         IF v_test>level2 THEN
25.           CASE detector_type IS
26.             WHEN ABOVE_1 => transgression <= TRUE;
27.             WHEN BELOW_1 => transgression <= FALSE; …71. END PROCESS;
72.   PROCESS(transgression) IS BEGIN
73.     IF transgression THEN
74.       IF detector_type = ABOVE_1 OR detector_type = BELOW_1 THEN
75.           write_detector_messages(file_name => file_name, message
      => time'image(NOW) & "fs" & character'VAL(9) & msg_type& transgres-
      sion_message & character'VAL(9) & " 'level1' = " & real'i-
      mage(level1) & character'VAL(9) & character'VAL(9) & ".   ");
76.           ASSERT FALSE REPORT time'image(NOW) & "fs" & charac-
      ter'VAL(9) & msg_type & transgression_message & character'VAL(9) &
      " 'level1' = " & real'image(level1) & character'VAL(9) & charac-
      ter'VAL(9) & ".   " SEVERITY message_type; …
```

Listing 2: Implementation of the voltage-detector (shortened)

5 Conclusion

It has been shown that with the use of the proposed methodology, the designer is able to create and compose specification rule checkers by using the parameterized basic detectors. Critical design parts can be observed continuously. During simulation, the detectors reliably check whether the design operates in its specifications and raises exceptions otherwise. Consequently, the verification phase can be automated which avoids error prone manual analysis of signal traces. The compliance of the design specification, and therefore the overall functionality of a circuit, can be totally observed with dedicated detectors. This increases the comprehension of the design and of the influence of certain design parameters on the functionality of the circuit, thereby increasing design robustness. Using the Verilog or VHDL assertion statements in the models' source code to observe the behavior of logic designs during simulation is state-of-the-art. The presented innovation is the separation of these assertion statements from the modes' source code to build separate observer models so that it can be used in mixed language simulator environments independently from the models' implementation language especially for analog and mixed-signal designs. Furthermore, this approach provides the option to combine the detectors to a more complex one to enable more versatile observations. An application of combined detectors to observe the continuous and peak current specification of a H-Bridge electronics of a stepper motor was already presented in [1]. All this increases designer's productivity and ensures design security through an accelerated automatic checking and reporting of important events. Since the detectors are implemented in a standardized HDL, they guarantee the compatibility of separate application schematics and different simulators and minimize efforts in creating and embedding specification rule checks independently of the overall testbench. The GUI allows a comfortable way to add detectors and set them up without manipulating testbenches itself. This allows reusing the detector-free testbench for further analysis in order to not waste CPU performance unnecessarily.

6 Outlook

The detector-GUI can be merged with the schematics editor for a visual selection of signals to observe. Detectors offer diagnostic support in several fields. For mixed-signal circuit design, detectors can be placed at internal nets to observe specification violations. A sequencing method can be used to adapt model parameters to converge to measurements or simulations on lower levels or it can be used for optimizing circuit parameters or for designing robustness tests. Circuit transfer efforts from provider to users can be reduced through providing virtual sockets with detectors on IP's ports to observe specifications independently from HDL/simulator. The actual library comprises several detectors for transient analysis. Further models are being added to address small signal simulations

7 Literature

[1] D. Dammers, D. Schollän, L.M. Voßkämper, "Accelerating Mixed Signal System Design Verification using New Diagnostic Method", ASIM-Workshop, March 5-6, 2009 Dresden, Germany, ISBN 978-3-8167-7981-0

[2] S. Max, "Fast Accurate and Complete ADC Testing", Proc. of the IEEE ITC, 1989, pp. 11 1-1 18.

[3] K. Arabi and B. Kaminska, "Parametric and Catastrophic Fault Coverage of Analog Circuits Using Oscillation-Test Methodology," IEEE VLSI Test Symp., 1997, Monterey, pp. 166-171