Mixed-abstraction Modeling Approach with Fault Injection for Hardware-Firmware Co-design and Functional Co-verification of an Automotive Airbag System on Chip Product

Dipl.-Ing. Dr.-techn. Thang Nguyen
Automotive Power Train and Safety
Infineon Technologies Austria AG
Villach, AUSTRIA
Thang.Nguyen@infineon.com
+43 (5) 1777 6361

Dipl.-Ing. Dieter Haerle
Automotive Power Train and Safety
Infineon Technologies Austria AG
Villach, AUSTRIA
Dieter.Haerle@infineon.com
+43 (5) 1777 6756

Abstract—**In modern automotive and consumer electronic industry, the number of functionalities and complexity is increasing. Additionally, there is pressure to produce reduced time-to-market and first-time-right design products, which pose great challenges for all chip designers. An effective utilization of modeling and simulation facilitates the design process so that these challenges can be met. This paper proposes a *mixed-abstraction* modeling strategy for *chip top-level functional verification* which significantly improves the simulation performance but still achieves high accuracy for a complex hardware-firmware co-verification. The digital content of the chip (consists of digital logic core and digital functional modules) is implemented with RTL format while the analog content of the chip is modeled using VHDL behavior modeling with support of real signal. Hence, a chip top-level simulation (hardware-firmware co-simulation) is achievable with event-based simulation technology. Additionally, the mixed-abstraction modeling strategy and method saves a considerable amount of effort in modeling, thus helps the project to gain time. For demonstration, a complete airbag system on chip (SoC) product was successfully modeled using this modeling approach. At an early stage of the design phase, the model is used in the chip top-level functional verification, and in parallel, supports optimization as well as errors debugging of hardware and firmware design activities.**

Keywords: ***mixed-abstraction modeling, chip top-level functional verification, embedded mixed-signal System-on-Chip application, hardware and firmware co-verification, airbag System-on-Chip, event-based simulation.***

I. INTRODUCTION

The tremendous growth in the silicon integration capability in recent years makes it possible to have a whole system integrated into a single chip [1]. Such systems are called System on Chip (SoC). Generally, the SoC architecture is tailored to the applications rather than being a general-purpose chip. Typically, as in all electronic systems, SoC hardware architecture consists of processing elements (processors that run embedded software, so called firmware or functional-specific hardware accelerators), IO devices, storage elements, interconnection structures linking all elements together [2]. Once the memory map and registers are defined and documented, development firmware (such as boot code to startup and initialize the system, hardware diagnosis, device drivers, functional application firmware, etc.) can be started [3]. According to [4], the most fundamental characteristic of a SoC is its complexity.

Typically, as shown in Figure 1 an airbag system is a heterogeneous system comprising of sensors, an airbag ECU (Electronic Control Unit) – which consists of an airbag SoC chip set, an airbag main micro-controller (uC), and actuators. The airbag ECU is hardware within a multiple airbag system as it controls the deployment of multiple airbags within a car.

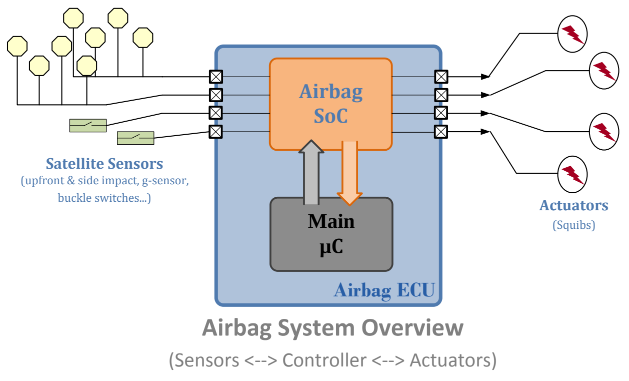

Airbag SoC
Main µC
Airbag ECU
Satellite Sensors
(upfront & side impact, g-sensor, buckle switches...)
Actuators
(Squibs)
Airbag System Overview
(Sensors <--> Controller <--> Actuators)

Figure 1: A typical airbag system from bird's eye view

During the operation, the sensors, such as buckle switches, accelerometers or pressure sensors, mounted in key locations of the vehicle, continuously measure the positions of impact, the severity of the collision and other variables. This information is provided to the airbag SoC chipset in form of analog signals, e.g.: modulated current, via the sensor interfaces. The airbag SoC chipset translates the analog sensor signals into digital words, which could be further processed by the main uC. The translated digital sensor data is reported to the main uC via the SPI (Serial Peripheral Interface) communication. Based on this information the airbag main uC decides if, where (location) and when the airbags (e.g.: actuators) should be deployed. In case of deployment, the main uC sends deployment request commands to the airbag SoC chipset, which drives the deployment interfaces of the airbag ECU, allowing a high analog current up to 2.2A to different squib resistors (depending on the impact locations). As a result, these squib resistors are heated up by the high current, which activates the chemical reaction of the chemical compound around the resistors and consequential in an airbag explosion.

The airbag ECU in this context uses a master-slave SPI communication system, in which the main uC is the master/commander and the airbag SoC chipset is the slave/executer. The airbag SoC acts also as a transceiver of the remote sensor network around the car to the main uC. It is not only responsible for receiving the analog sensor signals but also translating and reporting the sensor data to the main uC. The main uC which acts as a master is responsible for evaluating the sensor data received from the airbag SoC and requests a deployment within the right time window, at the right location, depending on the crash.

II. Problem Definition: Challenges of Airbag SoC Design and Integration

As shown in Figure 2, integrating an automotive airbag electronic system into a single SoC controlled by a main microcontroller makes the airbag SoC a very complex device. The main microcontroller (uC) is the master, and communicates with the airbag SoC chipset via the standard serial peripheral interface (SPI). The main functionalities of the airbag SoC chip include the following:

- Intelligent supply management unit consists of <u>S</u>witched <u>M</u>ode <u>P</u>ower <u>S</u>upply modules - SMPS (buck and boost converter); the <u>i</u>nternal <u>P</u>ower <u>S</u>upply <u>G</u>enerator – iPSG and a CENTRAL power supply control logic, which controls the chip power up and power down stage and monitors the system supply integrity and logic for failsafe state diagnostic in case of integrity violation.

- Remote sensor interfaces supporting different sensor types ranging from acceleration to rotational sensors and compliance to various standard such as PSI5 (Peripheral Sensor Interface) or DSI3 (Distributed System Interface).

- Safing engine assisting the main uC in observing the sensor data transmission and in confirming if a deployment event has occurred.

- Airbag deployment module – Squib drivers and deployment enabling logic – allowing current for the ignition of airbags, covering requirements on deployment timing and current as well as load conditions diagnostic.

These functionalities are distributed over both hardware (digital and analog front end (AFE)) and firmware glued on high density logic digital core, so called DCORE – digital CORE. Two important concepts of hardware and firmware integration are verification and validation since they ascertain that the designed component meets all the requirements. Since integration of hardware and firmware is the most crucial step in embedded system design, the sooner it is done, the better. However, there are many techniques to integrate hardware and firmware. The main goal should be not to waste time on debugging good firmware on broken hardware or debugging good hardware running broken software [5],[3].

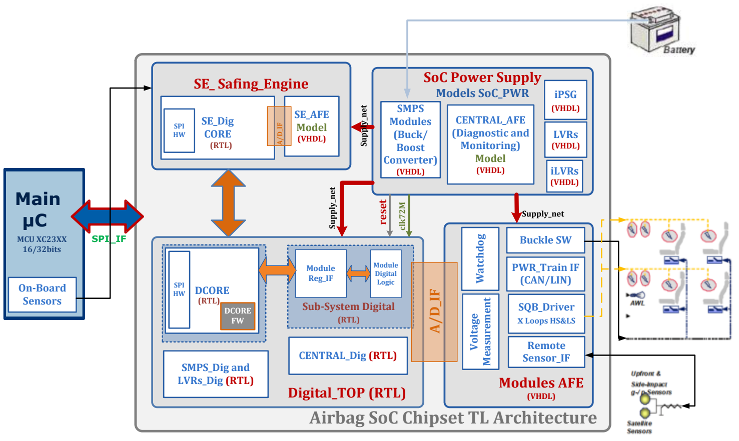


Figure 2: Overview of the airbag SoC chipset top-level implementation architecture

Additionally, the dramatically reduced product time-to-market together with the urge for first-time-right design has recently been strongly demanded in the automotive electronics industry. These trends pose great challenges to the design team. An effective utilization of modeling and simulation is proven to be one of the key factors that can significantly support designers facing technical challenges while still maintaining the industry requirements. Being able to choose a correct modeling strategy and approach has an important meaning for such complex embedded analogue mixed-signal chip top-level simulation, which classically deal with accuracy and simulation performance (simulation speed).

III. Paper Contributions

With the proposed mixed-abstraction modeling approach, the paper contributes the following:

- *A modeling methodology to support hardware-firmware co-design and co-verification* of complex SoC product development. Firmware, analog and digital hardware design teams, can use a common model to test their software/hardware while it is being developed. This also improves the collaboration between multiple disciplinary design teams.

- *A hardware fault injection modeling and simulation method* for the chip functional verification using a *global signaling concept*. This helps both firmware and hardware designers verify the concept of fault detection and of system protection from permanent faults.

- The provided mixed-abstraction approach is also aimed to “bridge” the gap between “speed” and “accuracy” issues of complete *chip top-level functional simulations* of complex ICs. This is achievable due to the fact that the model created from the modeling approach is pure VHDL (direct RTL and behavior VHDL); thus it has a high accuracy with a very fast event-based simulation.

- *The concept of the data flow and communication chain* from the sensor to the airbag SoC chip interface and further to the main microcontroller has been successfully proven from the early phases of the design process. This ensures that the customer gains high confidence in the design team.

- With the mixed-abstraction modeling approach with VHDL, the modeling team could save a significant modeling effort as the modeling task focuses only to model the analog content of the chip in a digital way. This is feasible with the support of real value, thus holds up the design team and the project to the product time-to-market.

IV. The Mixed-abstraction Modeling Approach

A. *The chip top-level functional modeling and verification challenges of the airbag SoC chipset:*

As described above, in order to realize the full functionality of the airbag SoC chipset, it requires correct interaction between, not only digital and analog hardware, but also the firmware part. One of the challenges is that many of such functional verifications can only be done at the chip top-level. Therefore, with such a high complexity and integration of the airbag SoC product, it is important to have an efficient model and verification strategy to cover as much functional verification of hardware and firmware but still does not jeopardize the accuracy and simulation speed.

A number of system modeling methods was evaluated including VHDL-AMS and SystemC(AMS) based on [1],[6],[7],[8] and based on state of practice from other work groups internally (such as sensor and control group or communication group or IP module group). The airbag SoC is a large scale integration of a few different single chips into a single die, e.g.: power supply management chip, the squib driver chip, the embedded safing engine chip. With the experience of having a full startup simulation run only on the power supply management chip using VHDL-AMS, it takes up to a day with some significant effort for the convergence issue to startup the simulation. As such, it is not sufficient to have the complete airbag SoC chipset simulate with VHDL-AMS. Using the SystemC(-AMS) approach could result in a very fast performance model prototype (albeit with some significant effort). On the other hand, the model prototype still needs to be adapted to the final hardware implementation. In comparison to the SystemC(-AMS) and VHDL-AMS approach, the mixed-abstraction modeling approach using pure VHDL has many advantages since it offers:

- Accuracy: the model with the real RTL implementation of the complete digital logic core with its embedded firmware to achieve a more accurate and reliable result.

- Simulation performance: a very fast event-based simulation without convergence issues.

- Effort: complete re-use of RTL implementation of the digital logic core significantly reduces effort for modeling. Additionally, the digital modeling of the analog front end modules in VHDL supported with real value requires far less effort compared to VHDL-AMS approach (which requires the modeling engineer to have a good skill base and experience for complex modules). Therefore, it also helps to improve the product time-to-market.

The decision for using the pure digital modeling approach with VHDL for the chip top-level simulation is driven by the following factors:

- The airbag SoC chip set deploys a high integration of very complex digital logic with high voltage (up to 35V) and high current (up to 2.0A) output driver circuits on a single die.

- The airbag SoC chip set architecture utilizes digital hardware with a high number of analog hardware modules (more than 22 modules); tightly together on embedded firmware (ROM) with real-time interaction requirement.

- The main power supply management unit - PMU (with switched-mode DC/DC supplier – e.g.: buck, boost converter, internal power supply generator, and voltage regulators) is digitally controlled at more than 70 MHz clock rate.

- Last but not least, a fast and reliable setup for the top-level simulation is also required due to time-to-market requirement.

B. *The mixed-abstraction modeling approach:*

As a result of the evaluations, the modeling strategy for the airbag SoC chipset is to have a full digital model of the chip in VHDL with as high accuracy as possible. This is achievable as shown in Figure 2 and Figure 4, by using the following steps:

- The digital core logic and the digital functionalities of the airbag SoC chip are directly reused from the RTL implementation of the digital design team.

- The AFE modules (such as the general purpose lamp driver, the sensor interface, the power management unit, the squib driver for deployment of the airbag, the buckle switches for seat belts, etc.) is modeled using *VHDL behavior modeling with the support of real value* (described in Section IV.C).

Note: Figure 3 shows the general architecture of a functional module in the airbag SoC design. Every functional module is comprised of a module AFE, which interfaces to the outside world or the system load, and a sub-system digital, which is then partitioned into the module logic and the module register interface, controlled by the digital CORE logic of the whole SoC.

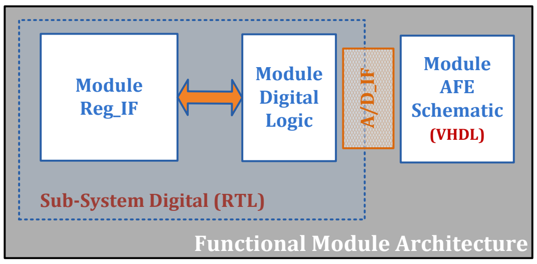


Figure 3: Functional module architecture in Airbag SoC design

- The top level netlist of the chip is *automatically* extracted from the final *tape-out layout schematic* by a specific tool developed internally and applied onto all the modules at the top level. This process is called the "marriage" as shown in Figure 4, and is used to replace the connection process when integrating the chip top-level model. The top-level marriage process described above contributes not only to the saving of effort in making the model, but also to enhance the accuracy of the model with regard to the final product. This is because the model top-level netlist and the toplevel tape out schematic are consistent.
- Finally, the firmware implementation is also injected directly into the model as ROM-mask, allowing firmware and hardware co-simulation. In practice, the firmware team uses the hardware model to verify their firmware while being developed. Bugs found during simulation are analyzed and identified if this is due to firmware or hardware. This results in a close collaboration between the teams.

The approach is called the mixed-abstraction modeling approach. Implementing the airbag SoC chipset model using mixed-abstraction approach enables the simulation of the whole chip with event-based simulation, which significantly improves the overall simulation performance.

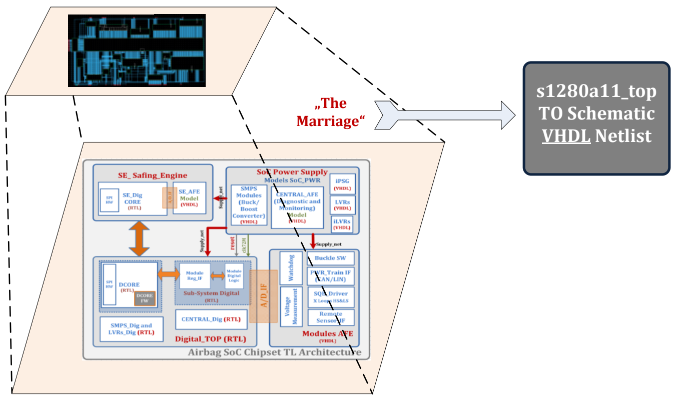


Figure 4: Mixed-abstraction modeling of the airbag SoC chip set

C. *VHDL Behavior Modeling for analog components with real-value support*

When using the mixed-abstraction approach, the effort on making of the complete chip model can be significantly reduced since the effort distributed mainly on modeling of the AFE modules of the chip. Modeling of analog components in a "digital" thinking way was done with the introduction of *analog_t* and *current_t* type, which represents the voltage and current notation of the analog world respectively. The concept of *analog_t* and *current_t* type is based on the real type. An *analog_pack* package is used to declare these two types. The *analog_pack* package was first written to support modeling of sensor and control application and has recently been adapted and widely used by other application groups such as communication and automotive. Within the *analog_pack* package, further signal types and resolution functions (as shown in Figure 5) used for analog modeling and verification purposes could be found such as:

- *analog_u* : Unknown/Undefined value
- *analog_x*: Driver conflict
- *analog_z*: High Impedance

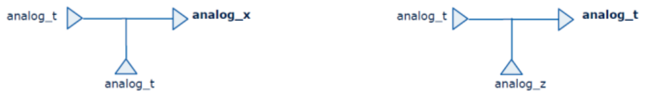
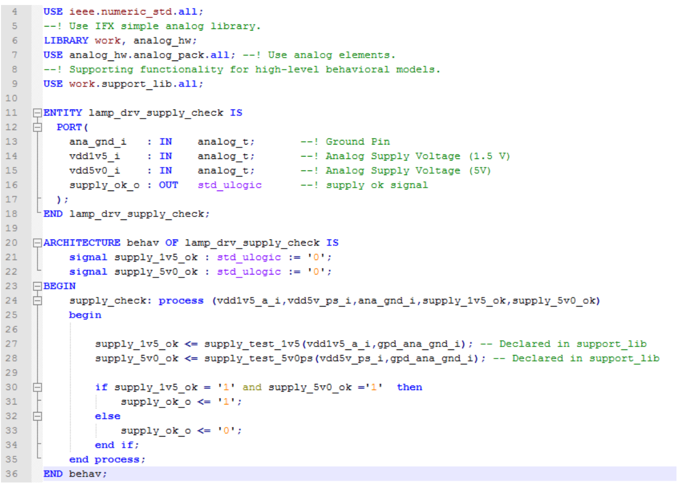


Figure 5: Resolution function examples used in the analog_pack package for modeling and simulation support.

An example of how to use VHDL with support of real value for the analog components modeling is illustrated with the *supply check block*, which is an internal function block of the airbag lamp driver AFE module, shown in Figure 7. This block checks the supply voltages 5v0 and 1v5 (for analog and digital sub-blocks respectively) of the airbag SoC lamp driver against the ground and returns the *supply_ok* output with Boolean type. The output '1' of *supply_ok* indicates that the voltage supply signals 5v0 and 1v5 stay at a stable level while '0' indicates that the supply signals are at an instable level or out of range. The behavior VHDL code implementation of this block is shown in Figure 6. The *supply_check* core process is implemented as a concurrent process. It instantly checks the input voltage and the ground levels. If the difference between the voltages stays at a predefined threshold, it returns a Boolean value to indicate the supply condition. Mostly, if the supply is not ok, the model will remain at a "deactivate" mode. The *supply_1v5_ok* and *supply_5v0_ok* are generic functions used in many different other models of the whole chip, therefore declared in the simulation *support_lib* library.


```
USE ieee.numeric_std.all;
--! Use IFX simple analog library.
LIBRARY work, analog_hw;
USE analog_hw.analog_pack.all; --! Use analog elements.
--! Supporting functionality for high-level behavioral models.
USE work.support_lib.all;

ENTITY lamp_drv_supply_check IS
  PORT(
    ana_gnd_i   : IN    analog_t;       --! Ground Pin
    vdd1v5_i    : IN    analog_t;       --! Analog Supply Voltage (1.5 V)
    vdd5v0_i    : IN    analog_t;       --! Analog Supply Voltage (5V)
    supply_ok_o : OUT   std_ulogic      --! supply ok signal
  );
END lamp_drv_supply_check;

ARCHITECTURE behav OF lamp_drv_supply_check IS
    signal supply_1v5_ok : std_ulogic := '0';
    signal supply_5v0_ok : std_ulogic := '0';
BEGIN
    supply_check: process (vdd1v5_a_i,vdd5v_ps_i,ana_gnd_i,supply_1v5_ok,supply_5v0_ok)
    begin

        supply_1v5_ok <= supply_test_1v5(vdd1v5_a_i,gpd_ana_gnd_i); -- Declared in support_lib
        supply_5v0_ok <= supply_test_5v0ps(vdd5v_ps_i,gpd_ana_gnd_i); -- Declared in support_lib

        if supply_1v5_ok = '1' and supply_5v0_ok ='1'  then
            supply_ok_o <= '1';
        else
            supply_ok_o <= '0';
        end if;
    end process;
END behav;
```

Figure 6: Implementation of the supply check module in VHDL

The complete lamp driver AFE functional model architecture is described in Figure 7. As an overall modeling, strategically the model represents both functional and physical partitioning of the real AFE module implementation. At this level, it is already required that the model pins should be fully compatible with the circuit implementation; so that integration into the top-level of the final airbag SoC model is possible.

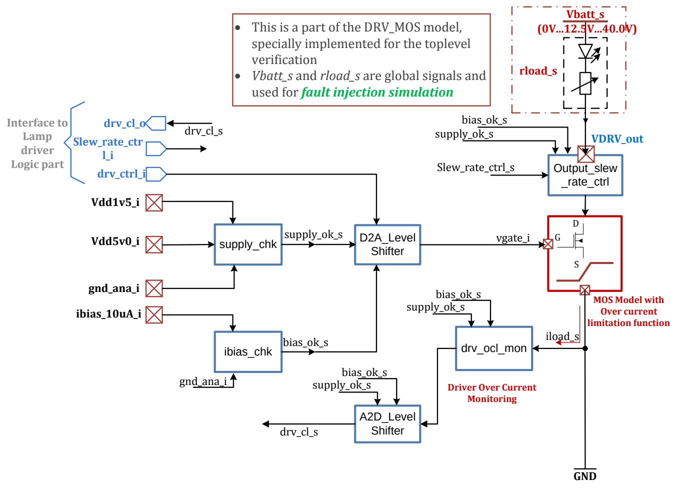


Figure 7: The airbag SoC lamp driver AFE model

The key module of the lamp driver AFE is the MOS model. Its role is a switch including an over-current limitation function. If the driver is enabled, the voltage at the gate will be driven to a higher level than the $V_{threshold}$, in order to close the switch/to open the transistor and to let a current flow through the LED (the lamp in this case is on). This current is called load current (*iload_s*), and is monitored continuously by the *drv_ocl_mon* module. In VHDL modeling, this translates into a process which is always sensitive to the *iload_s* current. The status reported from the driver over-current monitoring block is used by the digital logic part of the lamp driver module for *the over-current detection and driver protection functionality*. The driver turn on/off function; the driver slew rate configuration as well as the driver status report to the main micro-controller are done via *firmware* using SPI communication interface.

In general, the AFE models are verified against the circuit implementation by comparing the model with the real circuit using a subset of the module level verification suite and then reviewed/signed-off by the analog designer, firmware designer and functional module owner (e.g.: usually responsible for both digital, analog and firmware design). There is a specification for every AFE model as well, agreed by the analog designer; firmware designer and the top-level verification engineer. This process helps to avoid over-engineering the model implementation, which often ends up in slowing down the simulation performance unnecessarily. Moreover, the process also helps to enhance collaboration between different disciplinary groups. It has an important meaning during the design phase since common understanding amongst designers could be achieved; thus already lowering the design error due to misunderstanding.

D. *The Serial Peripheral Interface (SPI) model for firmware stimulation*

Instead of having the main uC model, a SPI generator model is used to generate SPI commands as a *firmware* stimulating source. This SPI generator is also used for checking the implementation of the SPI interface hardware module. The SPI generator has 4 digital interfaces complying to the SPI communication standard as shown in Figure 8, e.g.: sclk_o (SPI clock output), ss_o (slave select output), MOSI_o (Master Out Slave In) command line and MISO_i (Master In Slave Out) response line from the airbag SoC chip. The SPI frame generator could be configured to work at different sclk_o, to generate different frame lengths and to have different lead and lag times.

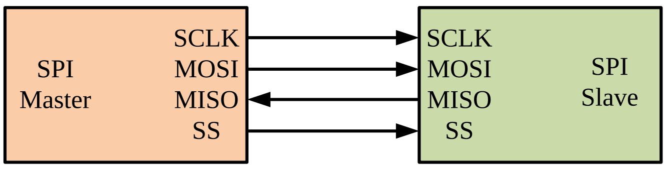


Figure 8: SPI Bus: Single Master – Single Slave Communication Model

V. Fault Injection Simulation Using Global Signaling Concept

The airbag SoC chipset is a safety critical component in the airbag system. Some parts of the airbag SoC chipset is categorized to ASIL-D level of the ISO-26262 safety standard (ASIL = Automotive Safety Integrity Level). Fault injection simulation is mandatory for many safety requirements, such as fault detection and fault protection features. As shown in Figure 7, an over-current event which caused by a high battery voltage level or low load impedance (short circuit) is considered as an application fault. Such fault could damage driver of the application if not be able to detect and protect.

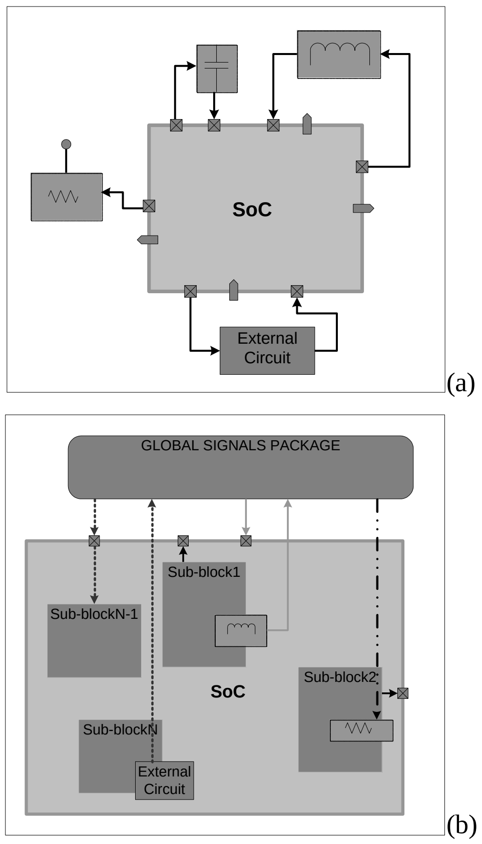
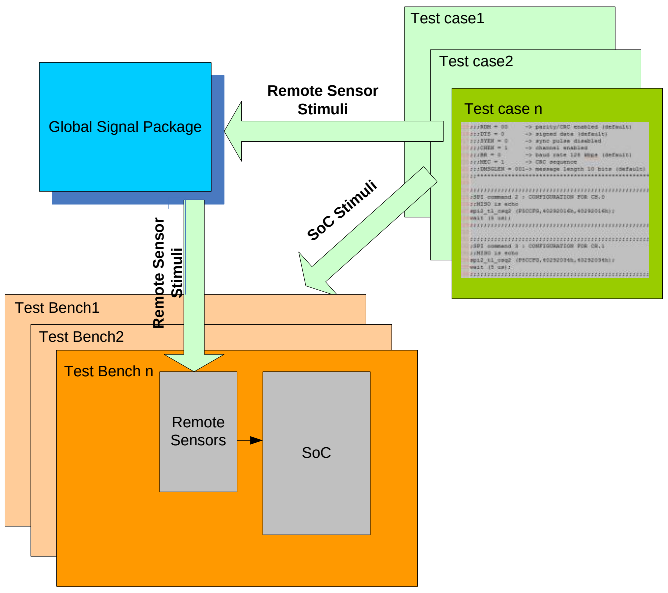


Figure 9: Fault injection simulation with global signal concept

The possibility of injecting faults allows the verification of over-current detection circuit and even driver protection functional verification. This is also very helpful for simulating different customer applications. In reality, the load impedance and the battery is not a part of the chip and illustrated in Figure 9(a) as an external circuit. The fault injection concept for simulation is as follows: in order to inject a fault during simulation, authors modeled the battery voltage *Vbatt_s* and the load impedance *rload_s* as *analog_t* type *signals* in the MOS model and declared them globally in a separate package, called *global signal* package. These signals are used for the calculation of the *iload_s* current. In Figure 9(b), this is illustrated by the external application circuit is modeled inside the chip model. Thus, via the *global signal* package, these load signals and battery voltage can be controlled or stimulated dynamically during the simulation process in order to create different simulation events, including fault events. In other words, the *global signal* package has to be declared at the header of the top-level test bench.

Though the concept is very simple and takes little effort to implement, it really does make fault injection as well as fault simulation possible and very easy. The *global signaling concept* is also widely used in creating different test scenarios for top-level verification. Amongst these, for example, is the generating of a different sensor date pattern using the *global signaling concept* as shown in Figure 10. The sensor data is declared in the *global signal* package. Verification engineers need to define different sensor data patterns in different test case scenarios and use the *global signal* package to pass these data to the test benches. Verifying different sensor data patterns according to different crashed scenarios helps to check the robustness of the airbag system since no false deployment is allowed.



```
;;;RDM = 00      -> parity/CRC enabled (default)
;;;DTS = 0       -> signed data (default)
;;;SYEN = 0      -> sync pulse disabled
;;;CHEN = 1      -> channel enabled
;;;BR = 0        -> baud rate 128 kbps (default)
;;;MEC = 1       -> CRC sequence
;;;DMSGLEN = 001-> message length 10 bits (default)
;;**********************************************************

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;SPI command 2 : CONFIGURATION FOR CH.0
;;MISO is echo
spi2_tl_csq2 (PSCCFG,40292016h,40292016h);
wait (5 us);
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;SPI command 3 : CONFIGURATION FOR CH.1
;;MISO is echo
spi2_tl_csq2 (PSCCFG,40292034h,40292034h);
wait (5 us);
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Figure 10: Using global signaling concept supporting top-level verification of sensor interfaces.

VI. Model Coverage: An Accuracy Evaluation vs. Implementation Effort

This section summarizes the model coverage from the physical implementation and functional point of view.

A. *The airbag SoC chipset model accuracy evaluation*

With the mixed-abstraction modeling approach, the model achieves a high accuracy since it covers the following points of *the chip's physical implementation* as illustrated in Figure 11 accordingly:

- Complete digital hardware architecture implementation (reusing of RTL code)
- Interfaces between digital and analog domains are strictly required when doing the modeling. These interfaces are verified at the top level *tape out schematic netlist* with the sub-module model integration step. If there is a mismatching, users immediately get an error during the compilation phase.
- The top level connectivity between modules is also covered as the real top level netlist is simulated. This top level schematic is used later for the chip layout.
- For simulation, the ROM mask is also used. This ROM mask is a translation of the firmware implementation (in C code) into a ROM structure which fits into the digital processor architecture. The ROM mask is also used later for the physical implementation of the chip.

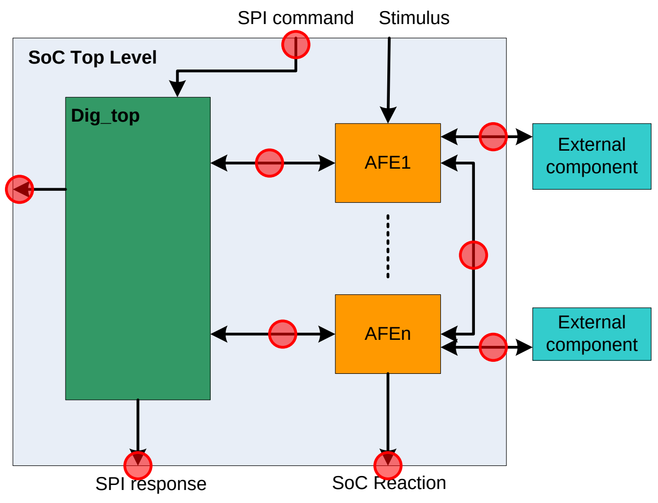


Figure 11: Model accuracy evaluation of the mixed-abstraction modeling approach

B. *The model limitation and the implementation effort*

Despite the fact that VHDL allows implementation of multi-field records to model both voltage and current on the same pin, authors choose the *global signaling concept* to keep either voltage or current information and feedback to the model. This helps to significantly gain the implementation effort and keep the analogue behavior simple. Having stated that, the verification of all the analogue modules are targeted at the circuit/schematic level. It is a part of the overall verification strategy and clearly impacts on the modeling approach.

According to practice, the complete modeling task of the airbag SoC chip including concept, specification, implementation, module validation, and top-level integration took about 3 months by two modeling engineers.

C. *The airbag SoC top-level model validation and verification coverage:*

Figure 12 presents the test bench architecture used for the verification of the airbag SoC chipset. The test bench consists of the following parts:

- Stimulus: digital stimulus (firmware, e.g.: SPI commands sequences and digital hardware) and analog stimulus (e.g.: vbatt_s, rload_s, 1v5 and 5v0 supplies, etc…). The SPI stimuli syntax contains SPI interfaces (SPI1 or SPI2), a chip select name, the MOSI command and an expected MISO response.
- Checkers: different checkers are also implemented for the regression test suite.
- SIM Log: all the simulation events, including stimulation and checkers, are captured and put in a log file (text format). The user can access the simulation log file after the simulation ends.

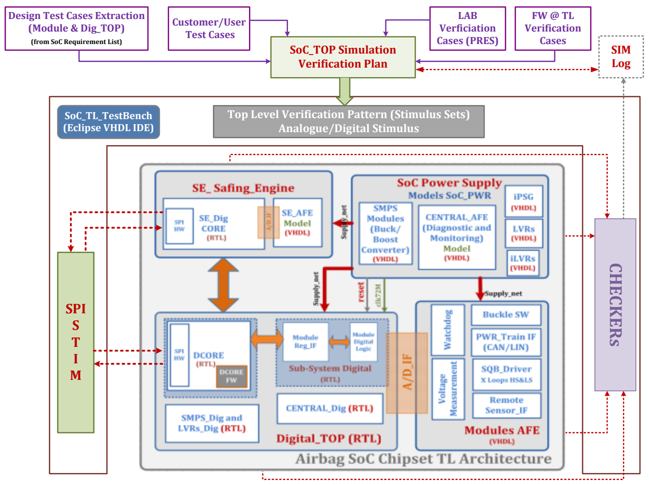


Figure 12: The airbag SoC top level test bench architecture

With the mixed-abstraction model, the airbag SoC *chip top-level simulation* focuses on *verifying* the following aspects:

- Functionalities of the chip, such as, power management function, airbag deployment function etc.; which require an interaction of analog/digital hardware and firmware can be verified. These functionalities can only be verified at the chip top level. For many modules, the analog models are highly compatible with the real schematic implementation.
- The firmware behavior at chip top-level can be verified; any errors found during the verification with the hardware could be directly fed back to the firmware team for bug fix and vice versa.
- The communication chain from the main uC to the airbag SoC chip, including sensor data transmission, command decoding and analog hardware status report can be also verified.
- Customer user cases verification

The checking of electrical parameters is dedicated to functional module verification level with mixed-signal verification technology, in which the module implementation (RTL code and real schematic) is used.

VII. Summary of Results/Conclusion

The mixed-abstraction modeling strategy and method are being successfully applied to a complex airbag SoC chipset product design. The chip has reached its tape out, with successful results using this approach. The whole airbag SoC chipset model is modeled with pure VHDL. In summary, authors would like to highlight the following results:

- The approach and result of the paper has now become a new modeling state of practice of the Power Train and Safety product development group at Infineon Technologies Austria AG. Authors would recommend this modeling strategy for complex embedded mixed-signal SoC product design, especially for hardware-firmware *functional* co-design and co-verification at chip top-level.
- With the mixed-abstraction model, a typical full top-level functional hardware/firmware co-verification is reduced to *less than 1 hour* (compared to a few days to one week) with a high accuracy of the simulation result.
- When modeling the analog front end, the concept of *global signal* is introduced. This allows the simplifying of the system load model and to inject error/fault conditions and finally to verify the system reaction under such conditions.
- Not only could a number of important hardware design features be verified using the system model but also the firmware behavior. Specially, important implementation bugs relating to the interactions and critical timing response between hardware and firmware were discovered in the early phase of the development process.
- The project could significantly *gain the time-to-market* and achieve *the design target* using the developed model.

Acknowledgement

The research leading to these results has received funding from the European Union's Seventh Framework Program (FP7-ICT-2011-7) under grant agreement n°287562 (http://verdi-fp7.eu/).

Authors would like to acknowledge contribution of the modeling team, especially Mr. Andrei Basa; and the hardware

design team as well as firmware design team valuable discussions with.

References

[1] Doucet, F. and R.K. Gupta, *Microelectronic System-on-Chip Modeling using Objects and their Relationships*. 2007, University of California - Center for Embedded Computer Systems: Irvine, California

[2] Lin, Y.-L.S., *Essential Issues in SOC Design - Designing Complex Systems-on-Chip*. 2006: Springer

[3] Andrew, J., ed. *Co-Verification of Hardware and Software for ARM SoC Design*. Embedded Technology Series, ed. J. Andrew. 2005, Elsevier.

[4] Jerraya, A.A. and W. Wolf, eds. *Multiprocessor Systems-on-Chips*. The Morgan Kaufmann Series in Systems on Silicon, ed. P. Ashenden and W. Wolf. 2005, Elsevier.

[5] Mayer-Lindenberg, F., *Dedicated Digital Processors: Methods in Hardware/Software System Design*. 2004, Hamburg, GERMANY: John Wiley & Sons.

[6] Luis, G. and Joao M.Fernandes, *Behavioral Modeling for embedded Systems and Technologies: Application for Design and Implementation*. 2009: Information Science Reference, New York

[7] Cagkan Erbas, *System Level Modeling and Design Space Exploration for Multiprocessor Embedded System-on-Chip Architectures*, PhD Thesis, Amsterdam University Press 2006, Amsterdam, Netherland

[8] Rashinkar P., Paterson P. and Singh L., *System-on-a-chip Verification: Methodology and Techniques*, 2002, Kluwer Academic Publishers, USA