Universal Verification Methodology for Power Management Unit

Márcio Soares
Instituto Superior Técnico, Universidade de Lisboa, Portugal
marciosoares@tecnico.ulisboa.pt

***Abstract*—Nowadays, mixed signal applications are widespread in the semiconductor industry. Mixed signal validation adds complexity to the verification process, which difficults functional verification Universal Verification Methodology (UVM) is the current standard methodology for verifying digital and mixed-signal designs. Real Number Modelling allows the description of mixed-signal designs using a High-level verification language. This approach imposes limitations upon the verification process but builds the foundation for coverage-driven verification and functional verification. Universal Verification Methodology applied to models based on Real Number Modelling allows for robust verification environments while significantly reducing simulation time and time-to-market. In this work, a UVM testbench environment is proposed for voltage regulators and a power management unit verification, under the scope of pAvIs project. This new verification solution is integrated in the design and test flow of SiliconGate.**

***Index Terms*—Universal Verification Methodology, Power Management Unit, Coverage-driven Verification, mixed-signal testing, voltage regulators**

I. Introduction

Advances in fabrication technology and increasing time to market pressure alongside physical effects of shrinking the process technology impose greater challenges in design and testing of System-on-a-chip (SoC)s. Modern SoCs encompass both digital and analog blocks and require pre-silicon verification to validate their integration [1]. This approach is mandatory as multiple issues can be detected at early stages, aiming for first-time right silicon, saving time and resources. Analog and digital simulation exist in different domains [2]. Analog verification is an ad-hoc complex procedure that performs computation of large data structures with no obvious signal flow pattern, therefore lacking of a standardized methodology [2]. On the other hand, digital verification is powered with versatile tools that allow for testbench automation, constrained-random stimulus generation, coverage collection and much more. To improve the accuracy of the analog model representation, Real Number Modelling (RNM) is used to quantify analog behavior in the relevant wires. Analog behavior is described as real data, where floating point numbers are used. This approach relies on a digital solver to achieve near-digital verification speeds and improves the pre-silicon verification time when compared to transistor-level simulation. By adopting this methodology, full-chip verification is facilitated for a large scale of mixed-signal designs and the methodologies applied for digital testing can be ported and adapted to fit a mixed-signal design. These behavioral models are also important as they are rapidly produced and independent of the process technology.

Universal Verification Methodology (UVM) proves to be an improvement for traditional mixed-signal and digital verification methods as it provides constrained-random coverage-driven test environments. It allows for test automation granting high configurability, interoperability and Coverage-driven Verification (CDV). Directed tests can also be explored by UVM as it provides a robust methodology for test generation. Testbenches can be designed for reusability reducing the effort when migrating components for different projects.

Combining UVM and RNM enables high-performance mixed signal SoC verification. RNM is limited for analog verification, but excels when integrated in a verification environment based on functional and coverage-driven verification. In a mixed-signal design, one can use it to imitate the analog counterpart of the SoC which enables extensive testing and increased simulation speeds. This verification level would not be possible for a pure embedded analog design.

Patient and Environment Aware Adaptive Intelligent Sensor Systems (pAvIs) European project [3] aims to develop innovative approaches in improving the electronics and intelligent sensor systems for professional healthcare diagnosis.

SiliconGate and Instituto Superior Técnico are designing a Power Management Unit (PMU) of an intelligent sensor system for a Magnetic Resonance Imaging (MRI) machine with mixed-signal processing Intellectual Property (IP)s at its core for the pAvIs project. These sensors will be embedded in an environment with strong electromagnetic fields, which require specialized design and testing techniques to obtain a fully functional and robust integrated circuit.

A UVM-based architecture for a RNM model of PMU and its components is presented in this thesis using the pAvIs project as an example. This architecture is used for the validation of voltage regulators and digital core within the model and their integration to fit the specification of the PMU, described in table I. The implementation for the PMU is presented in Figure 1.

II. PMU architecture

A. Analog SV-RNM

When designing mixed-signal circuits it is essential to perform a translation from the specification to the real transistor-

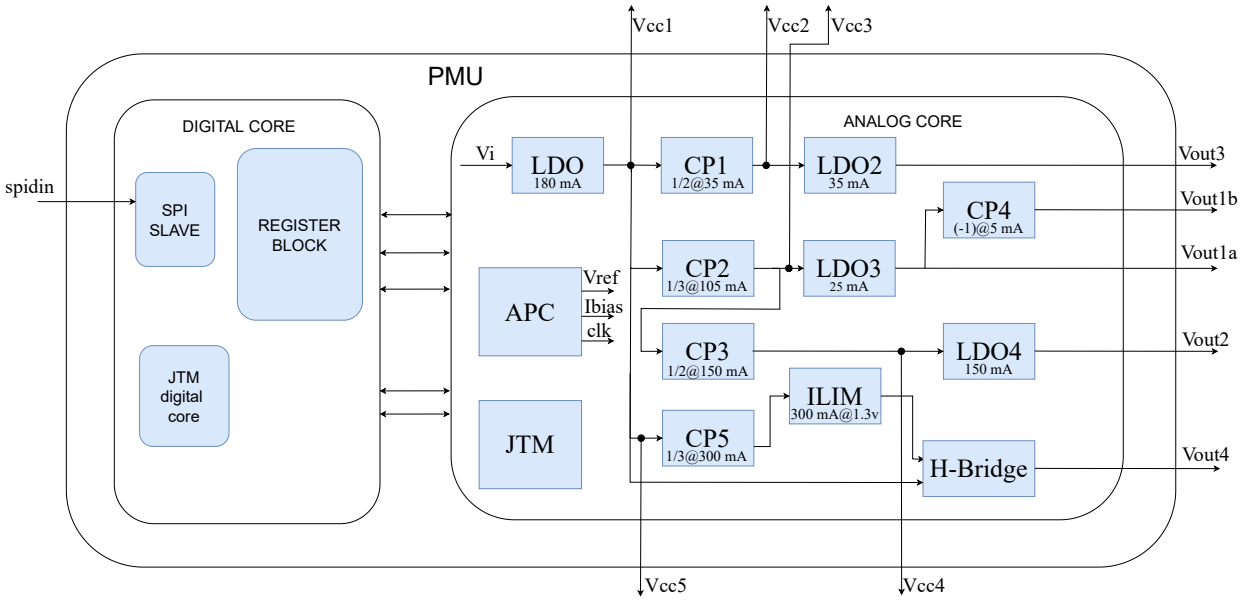

Fig. 1. PMU architecture.

TABLE I
PMU SPECIFICATIONS.

Symbol	Min	Typ	Max
V_{in}	10 V	12 V	14 V
$freq$	-	540/145 MHz	-
$out1a$	2.7 V	2.8 V	2.9 V
	0 mA	13 mA	20 mA
$out1b$	-2.9 V	-2.8 V	-2.7 V
	0 mA	1 mA	5 mA
$out2$	1.3 V	1.4 V	1.5 V
	0 mA	100 mA	150 mA
$out3$	3.2 V	3.3 V	3.4 V
	0 mA	25 mA	35 mA
$out4$[a]	-9.9 V	-11.9 V	13.9 V
	0 mA	0.1 mA	1 mA
$out4$[a]	1.3 V	2 V	2.6 V
	0 mA	160 mA	300 mA

[a]out4 output has two modes of operation.

level implementation. A verilog model of the mixed-signal circuit is also frequently designed using RNM for key parameters. System Verilog (SV) RNM models are of extreme importance for this transition between specification and transistor-level design, as they are responsible to ensure that the design fits the specification and that the integration of digital and analog counterparts is properly implemented.

The verilog model of mixed-signal circuits can be designed in two different contexts. This first one depicts the scenario where the RNM model is built for the first time, only based on the datasheet specification. The analog design team develops the transistor-level circuit to be in accordance with the RNM model and, consequently, in accordance with the datasheet specification.

The second scenario is the one where the RNM models and transistor-level designs are independently ported from a previously existing project. In this scenario, design features and parameters must be updated to fit the new project's datasheet for both RNM model and transistor-level design. At the end both design must present the same behavior.

When the transistor (or gate) level schematic already exists, the extraction of the corresponding schematic is very useful for the verilog model design since it contains the high level structure of the design. Moreover digital cells are already automatically converted to SV logic function and need no further changes. The analog behaviour is the one to model as analog signals are described using a discrete representation, defining the analog counterpart as a signal flow event-driven model. For the scope of behavioral verification information such as supply verification and threshold values, evolution of the output voltage through an output capacitance model (when applicable) and definition of other output signals, RNM provides an accurate representation of each component functional behavior. All these features are described with SV primitives. Analog signals are driven as 64 bit floating point numbers, and converted to real numbers to perform computations ($bitstoreal and $realtobits serve as conversion tools).

B. pAvIs PMU Architecture

A PMU is the circuit responsible for power management of an SoC. Its main roles consist of generating reference voltages, controlling power modes, battery charging, DC to DC conversion and other auxiliary functions to control the power flow. Generally, SoCs powered by a PMU require multiple voltage domains and can also be supplied from multiple power sources which define multiple requirements for operation of the PMU. The architecture of the pAvIs PMU is displayed on Figure 1.

C. pAvIs Digital core

The pAvIs PMU possesses a digital core that encapsulates a register bank to store and configure digital control bits for the IPs inside the analog core. The digital core also receives input signals from the analog core to ensure observability and awareness of the status of the PMU internal voltages. The register bank is composed of thirteen 32-bit registers, that store configuration bits for each IP and other PMU control bits.

D. pAvIs Analog core

The pAvIs analog core is composed of an Advanced Power Control (APC), four Low-dropout Regulator (LDO)s, five Charge Pump (CP)s), a Junction Temperature Measurement (JTM) regulator, a Current Limiter (ILIM) and an H-bridge Regulator (HB).

1) APC RNM model: The APC generates control signals that coordinate the start-up sequence and overall functionality of the PMU, representing the backbone of the PMU. It generates enable signals reference voltages, bias currents and other control signals. Each regulator start-up is complete when it returns the power good (*pg*) indicator to the APC. Only after this bit is held high is the APC allowed to enable the next regulator. Generated reference voltages and currents can be digitally trimmed for increased performance.

The APC model is designed as a SV behavioral Register-transfer Level (RTL) description combined with extracted views of a ring oscillator for clock generation, a bandgap circuit, and a clock division block. This inevitably results in a simplified description of the analog design. To emulate the time related with the rising of reference voltages, debouncer circuits, or propagation of combinational logic, expected time stamps are considered, which suffices in the scope of behavioral verification. Reference voltages and currents are defined as real numbers with a 64-bit representation for each regulator with the respective trimming features. A test block is also designed with the purpose of adding controllability and observability to internal APC nets.

2) LDO RNM model: The LDO is a regulator that provides a constant output voltage, even when the supply voltage is close to the output voltage. LDO's, and voltage regulators in general, include configurable features that are controlled through a digital port. The LDO model is a pure behavioral RTL design. The definition of the programmable output voltage is configured at the digital input di. The output pin *pg* is set to high when the output voltage reaches 95% of the programmed voltage. *pg* is reseted when the output voltage is lower than 90% of the programmed value.

3) CP RNM model: The charge pump is a switched capacitor circuit that regulates the output voltage using predefined ratios of the input and/or output voltage. It possesses a digital control logic that acts on switches to explore charge transfer between capacitors. The charge pump model top representation is extracted from the design schematic. The control block and generation of non-overlapped switching clocks consist of logic gates which are converted in SV logic representation. The power, voltage divider, and comparator blocks are replaced by simplified RNM behavioral models which implements the relevant behavior at a high level of abstraction. The pg and pgdvdd functionality is the same as the LDO.

4) Remaining PMU blocks: The JTM is a high resolution Analog-to-digital Converter (ADC) for temperature, voltage and current monitoring. The HB specifies two modes of operation for one of the outputs of the PMU (see table I). A current mode, where the model outputs a current through the pin and a voltage mode where it defines a positive or a negative voltage through the output pin. The ILIM IP limits the current capping the output current for the HB current mode to 300 mA. These regulators and the APC were not individually tested with UVM. Instead, they were tested with traditional SV flow.

E. Verification techniques

The most basic SV verification environment consists of stimuli being driven to the Device Under Test (DUT) with the intention of performing a self driven verification by the engineer. The verification process relies on viewing waveforms through a wave viewer software and individually verifying every output of the design.

An improved version of this methodology explores an implementation of self-checkable processes which makes use of SV features such as the wait() statement and verification of parameter thresholds with printed messages upon failure. Currently at SiliconGate self-checkable processes are commonly used for functional verification.

UVM aims to consolidate the verification process, providing a robust verification environment with a well defined structure purposely built to explore extensive coverage-driven, constrained-random and self-sufficient testbenches. It possesses specialized tools to automate test generation and systematize self-checkable processes and coverage collection mechanisms.

III. Universal Verification Methodology

A. Overview

RNM [4] allows the description of mixed-signal designs analog behaviour, [5], [6], enabling signal flow event-driven models using High-level Verification Language (HVL). This approach inevitably imposes limitations, but builds the foundation for Coverage-driven Verification (CDV), [7], [8] and functional verification. Functional verification consumes a significant time of the project, [9]. Additionally, when aiming for functional verification, several test scenarios might not be considered.

UVM is the current standard methodology for verifying digital and mixed-signal designs. With its structured library classes, it allows the creation of constrained-random coverage-driven environments benefiting from SV object-oriented features, [10]. It has become the industry standard for hardware verification as it is supported by the main Electronic Design Automation (EDA) vendors and adopted worldwide for digital and mixed-signal IP testing.

An example application of UVM for a RNM design was presented for an ADC [11]. In this work, the use of UVM alongside RNM is illustrated for the validation of an ADC and its components. UVM has been used to test the supply module of a Microcontroller Unit (MCU) inside a Real Time Clock (RTC) to enter or exit the ultra-low power modes [12] . However, in this work, only the RTC functionality was targeted with the UVM validation, and no voltage regulation IP cores were tested. In the previous publication based on this thesis work [13], an application of UVM to test voltage regulators was implemented. The use of UVM to validate verilogAMS models was also addressed in [14] and [15]. The reliability of these approaches was also assessed [15], where advantages and disadvantages were explicitly defined.

Another relevant project explores a UVM CDV environment to test the implementation of a Serial Peripheral Interface (SPI) protocol [7], as a similar approach is considered for validating the same SPI communication protocol inside the pAvIs' PMU core.

Combining RNM with UVM for mixed signal IP verification [11], [16], enables testbench automation, coverage collection and increased simulation speeds.

B. UVM testbench and environment

A typical UVM testbench architecture is displayed in Figure 2.

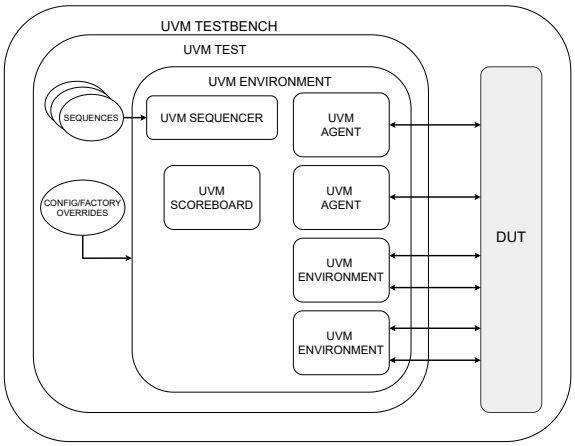

Fig. 2. Typical UVM architecture, [17].

To briefly introduce UVM, some important concepts are explained below.

- Testbench - The testbench represents the top level of the hierarchy. It instantiates the UVM test environments, interfaces, DUT and other key components and coordinates the testing procedure.
- Test - Specifies the test scenario for the testbench. It instantiates the environment (uvm_env) and environment configuration properties, [11]. Several test environments can be instantiated for the same DUT.
- UVM environment - Derived from the base class uvm_env, it is a key building block of UVM test environment and system verification. The uvm_env provides a set of features that allow the re-usability and flexibility of the environment. The environment may have multiple agents for different interfaces, a common scoreboard, a functional coverage collector and even other environments. It is responsible to integrate all the henceforth described components and coordinate sequence generation, monitoring and coverage checking of the DUT.
- Universal Verification Component (UVC) - A UVC represents a self contained, plug'n'play verification environment for a specific interface or a generic IP. It consists of one or more configurable agents with a predefined set of sequences that translates to test stimuli, coverage model and comprehensive failure report protocol based in UVM.
- Phase - UVM controls the creation, configuration and execution of a simulation run using phases. Phasing acts as a synchronization mechanism. Only after each component successfully finishes their execution in a phase, is the control flow allowed to proceed to the next phase. This creates a very structured and intuitive set of events. All phases can be grouped into three main phases: build-time phases, run-time phases and clean-up phases.
- Agent - The agent, derived from the uvm_agent class, encapsulates a driver, a sequencer, a monitor and a collector(when applicable). By doing so, it binds these components together to drive, monitor and collect coverage from the DUT or specific ports of the DUT. This represents an essential abstraction layer as it eases the integration of these components in the environment. An Agent can be active, if it drives signals to the DUT, or passive, if it only monitors the ports of the DUT. A representative block diagram for a generic active agent and its flow is displayed in Figure 3.
  The sequence (represented as P in figure 3) exemplifies a pattern of signals (sequence-items) to be driven to the DUT. The sequencer retrieves, randomizes and sends sequence-items to the driver on demand. The driver implements the protocol to drive signals to the DUT and communicates a item_done flag to the sequencer once the data transfer is done.
- Sequence-item - Derived from the uvm_sequence_item base class, sequence-items represent stimuli and transactions of the UVM environment. A set of attributes, constrains and methods can be defined for the sequence-item for it to fit the needs of the DUT or the transaction type and mold the data.
- Sequence - A sequence, derived from the uvm_sequence class, describes a bundle of transactions (sequence-items) or other sequences. To generate sequences, UVM provides a set of macros that implement the standard flow of transaction generation.
- Sequencer - A sequencer, derived from the uvm_sequencer class, is responsible for sequence generation and controls the items to be sent to the driver. It generates data on-demand and returns them to the driver, as one can see in figure 3). Sequencers can exist inside agents, where their scope is local for the agent,

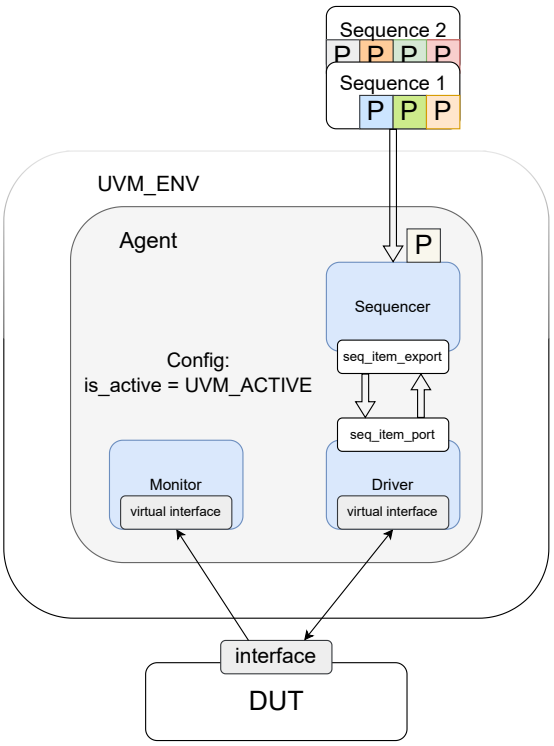

Fig. 3. Agent block diagram.

or inside test environments as virtual sequencers. In this case it provides a centralized sequence generation mechanism, coordinating different elements of the environment.

- Driver - The driver, derived from the uvm_driver class, is connected to the DUT via a virtual interface and drives stimuli to its ports. During the run_phase task, the driver decodes the transaction to obtain the signals and consume simulation time to forward the data.
- Monitor - The monitor is a passive component in the environment. It retrieves signals from the DUT and sends them to a scoreboard or other UVM components to perform behavioural and procedural validation. Monitors are also used to perform coverage control in order to cover several test scenarios. The connection to the DUT is made via a virtual interface.
- Virtual Interface - The virtual interface represents an abstraction layer as it represents a pointer to an actual interface. It allows the defined classes to access the DUT ports while promoting reusability.
- Scoreboard - The scoreboard, derived from the uvm_scoreboard class, verifies if the DUT outputs the expected results. It retrieves information from monitors and, alongside a reference model, checks the correctness of the DUT outputs. The uvm_scoreboard is built with robust failure reporting features. UVM provides a set of report macros that, when correctly implemented, can extensively monitor the execution of the simulation run.
- Reference model - The reference model emulates the behaviour of the DUT generating the correct output for a specific input. These results are afterwards used by the scoreboard to evaluate the DUT's output.
- Factory - The UVM factory provides a standardized way of replacing an existing class by any of its inherited child classes. Upon creation, objects are registered in the factory with UVM defined macros. Once registered, the factory provides flexibility allowing the user to override certain types and instances of class objects by its child types.

IV. Implementation

First and foremost, reusable UVM enviroments are described. These will be building blocks of the IP's UVM environments.

These environments are designed for reusability, aiming for test automation. All voltage regulators have an input and output voltage pin and require an input reference voltage and a dedicated input for sensing the output voltage for feedback purposes. They also possess an enable input and digital inputs to control the output voltage, a digital interface responsible for control and configuration of the IP and an analog interface responsible for providing the current and voltage references and controlling the output voltage. As these interfaces are common to the majority of the regulators (LDOs, DCDC or switch-capacitor based) it is possible to create a parameterized reusable UVM environment, that runs a pre-defined set of tests to validate their correct implementation with respect to each regulator. Power and output interfaces also possess ports that exist in all regulators making it possible to generalize a UVM environment for each.

A. Digital UVC

A digital UVC is generalized and encapsulated with default test sequences for each variable. The agent inside this UVC can be set to active, driving stimuli to the DUT, or passive only acting as a listener of an interface. The passive implementation will be useful when testing the PMU digital core as this UVC can be replicated to sample the output of the registers for each regulator.

The sequence-item component possesses the following signals:

- *dis* - The dis pin is used to enable and disbable the regulator.
- *dislvl* - *dislvl* disables level converters and forces the regulator to ignore all driven digital signals and assume default values. It is used to disable the interface core when digital supply is not present or before the release of power-on-reset by the APC.
- *dissink* - When held high it disables the sink resistors that perform a pull down of the output voltage, if the core is disabled.
- $vprog_n$ - The $vprog_n$ signals can be connected to an array of control bits to program certain features of the regulator. For example, vprog can be used to define the

output voltage, to perform current or voltage trimming or to define operating modes of a regulator.
- *test* - The test interface represent control pins used to force testing modes for characterization and production test. Test modes can provide controllability and observability of internal signals through the *anatestbus* output pin.

To coordinate tests, functional test sequences are developed for each regulator. Constraints are also set inside the sequence-item class, and enables CDV for a more thorough verification. These can be called from all environments that possess a digital UVC in a plug and play fashion.

B. Power UVC

The agent inside this UVC is set to active as it provides supply voltages for the regulators. Different sequence-item classes are generated for each regulator and can be selected on higher levels of the test environment. By taking advantage of UVM object oriented features, the user can perform constraint layering, which allows the definition of new constraints for voltage margins and other parameters.

Generic power interface sequence-item signals:

- *vref* - Reference voltage.
- *avdd* - Supply for analog circuits referenced to agnd.
- *vcasn* - Supply for analog circuits referenced to agnd.
- *dvdd* - Supply for the digital cells and level converters.
- *ibp1u0* - Bias current for internal current mirrors.
- *agnd*, *dgnd* - Analog and digital ground supply.

C. Regulator UVC

This UVC is configured as passive, sampling the output interface in voltage regulators validation, and as active in digital core testing, driving stimuli through its interface. The sequence-item component possesses the following signals:

- *vo* - Regulated output voltage.
- *pg* - Power good indicator. It is set to high when the output voltage reaches 95% of the programmed voltage.
- *pgdvdd* - *pg* converted to digital domain.
- *anatestbus* - Output test bus.
- *anatestreq* - Output bit that is held high in test modes.
- *rl_vprog*$_n$ - Adaptable output real signal.

D. Voltage regulators' UVM environment

The proposed architecture for voltage regulators is displayed in Figure 4. This environment will be used to test the CP and the LDO.

The testbench has four agents, one for each environment. The agents inside the power, configuration and digital environment are set to active and drive transactions to the DUT. The agent inside the regulator environment is set to passive and only samples the output interface of the DUT.

A virtual sequencer coordinates the generation of the constrained random stimuli for the sequencers on the active agents.

During run_phase, input packets are driven to the DUT and the output port is sampled for each bundle of input signals.

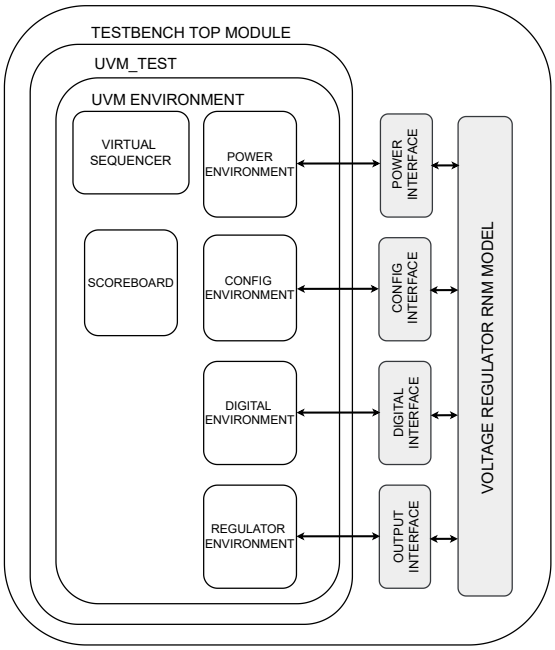

Fig. 4. UVM environment for voltage regulator testing.

Each combination of input and corespondent output are orderly saved locally in queues as these will later be used to generate a reference model for comparison purposes.

1) Sequence description: For each voltage regulator, test sequences are encapsulated inside the sequences class. These sequences aim for functional verification of the voltage regulator model. Therefore directed test are considered and described for active UVCs. Amongst these directed oriented stimuli, pseudo-random sequences are also defined.

2) Scoreboards and reference models: During the check and report phases, validation metrics are generated from the UVM test environment. These contain information regarding simulation status, unmatched comparisons, coverage collection and detailed information about the test execution. To perform validation, the queued input and output samples are sent to a reference model that generates a golden reference based on the input signals. The output is then orderly compared to this golden reference from each input packet during the check phase of the uvm_scoreboard.

E. Digital core UVM environment

The proposed architecture to test the digital core is displayed in Figure 5.

As stated in subsection IV-A, the digital UVC is set to passive in this verification environment. The digital environment registers all digital control pins from all the IPs present in the analog counterpart of the design. To send inputs to the digital core, a SPI UVC is designed. JTM, APC and ILIM UVCs encapsulate specific environments to test specific control bits for the current limiter IP, the JTM and the APC. A general input UVC and a general output UVC are also designed. They possess specific signals to the digital core that do not fit in

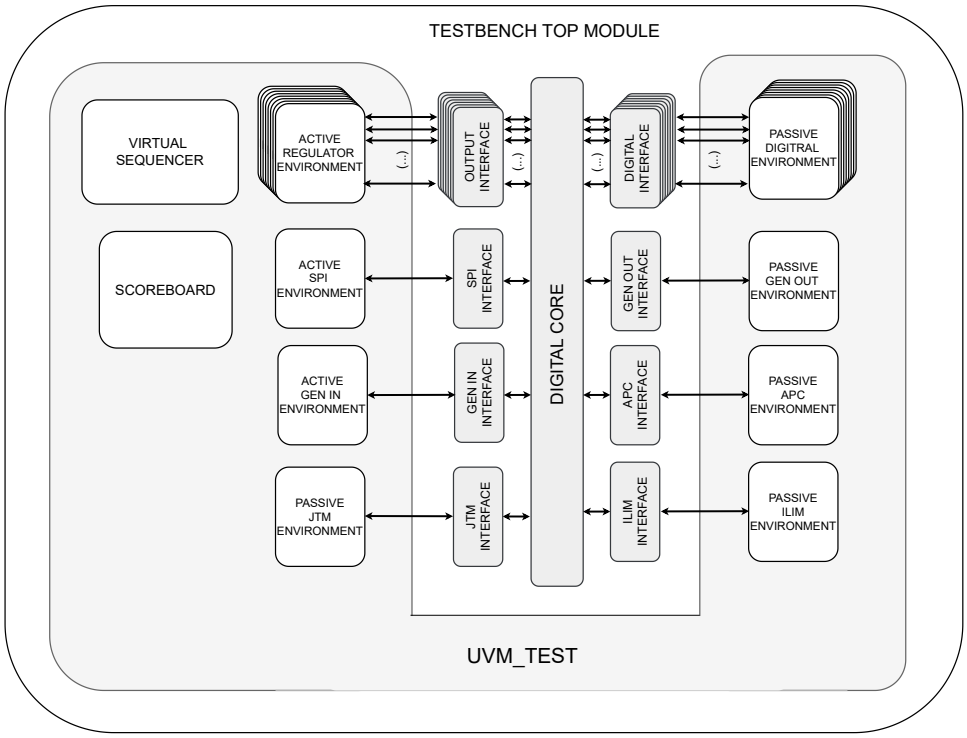

Fig. 5. UVM environment for digital core testing.

the other UVCs encapsulation. The regulator UVC is also instantiated to to simulate analog core digital outputs from its voltage regulators.

1) Sequence description: Test sequences for the considered active UVCs (gen_in, regulator and SPI) are randomly defined. The SPI UVC generates pseudo-random addresses(constrained for the number of addressable registers) and random data buses. Test sequences for writes and reads are implemented.

2) Scoreboard and reference model: The scoreboard implementation has a similar configuration as the one considered for the voltage regulators. During run-phase Every input and output sequence-items are inserted in queues. The SPI databus represents the reference model. To validate writes, output samples are concatenated given an address. A mask is applied to the input SPI data bus and compared to the concatenated output bus. To validate reads the input SPI data bus is compared to the spidout serial peripheral interface.

F. PMU UVM environment

The proposed architecture is displayed in Figure 6.

The purpose of this UVM environment is to test the start-up sequence (Figure 7) of the PMU core and integration of the digital core with the models inside the analog core. This particular start-up order is chosen in order to ensure that important reference voltages are defined as soon as possible.

Test sequences are sent through the SPI interface to program the registers and define control parameters on each regulator.

V. Results

For the scope of this work, results are assessed for coverage metrics paired with scoreboard verification.

Coverage reports are obtained with Unified Report Generator (URG) Synopsys VCS tool while the scoreboard validates the outputs given an input set.

A. LDO's considered coverpoints and crosses

Considered coverpoints, crosses and respective results are shown in table II. Considered crosses are explained below.

- ***cx_test_di*** - This cross is ment to test all programming codes for the output voltage (*di*) with enable, *dissink* and *dislvl* bits of the LDO. The test coverpoint is not considered.
- ***cx_iom_vfb*** - This crosses *iomread*, *vfbread* and *iomsw* coverpoints with a *dislvl* low coverpoint. *iatb* and *vatb* outut signals are assessed for this cross.

Initially directed stimuli is considered to functionally verify the design. Therefore, a standard SV test adopted at SiliconGate is replicated in the UVM test sequences, which resulted in a coverage score of 86.58%. Crosses results are low as achieving certain input combinations require specific sequences which were not considered for functional verification. For example, a di sweep is only conducted for an enabled LDO and therefore several bins are not exercised for the *cx_test_di* cross. Unusual input combination are prone to identifying unexpected behavior.

Once the several pseudo-random stimuli are introduced in the environment, 100% coverage is achieved.

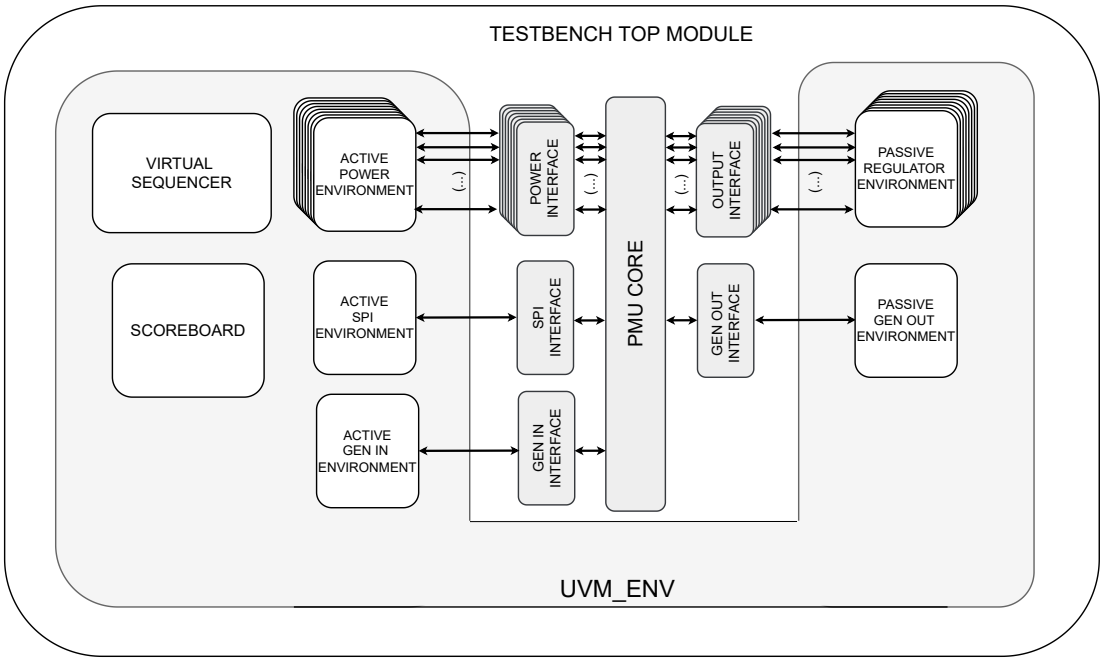

Fig. 6. UVM environment for PMU core testing.

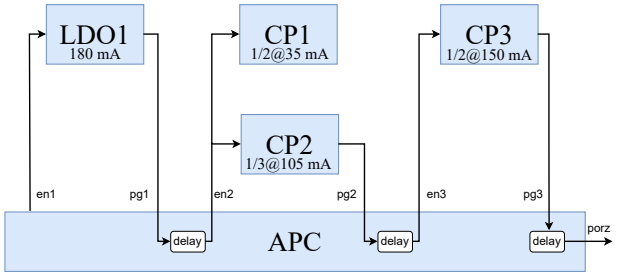

Fig. 7. PMU's start-up sequence.

For all tests, the scoreboard successfully validates the LDO RNM model behaviour.

TABLE II
COVERAGE RESULTS FOR COVERPOINTS AND CROSSES OF THE LDO.

Variable	Expected	Uncovered	Covered	Cov. percentage[%]
enzdvdd (coverpoint)	2	0	2	100
dislvl (coverpoint)	2	0	2	100
dissink (coverpoint)	2	0	2	100
iomread (coverpoint)	2	0	2	100
iomsw (coverpoint)	4	0	4	100
vfbread (coverpoint)	2	0	2	100
di (coverpoint)	11	0	11	100
test (coverpoint)	9	0	9	100
cx_test_di (cross)	88	73	15	17.05
cx_iom_vfb (cross)	32	25	7	21.88

B. CP's considered coverpoints and crosses

Considered coverpoints, crosses and initial results are shown in table III. Considered crosses are explained below.

Variable	Expected	Covered (SF)	Covered (UVM)	Cov. percentage(SF)[%]	Cov. percentage(UVM)[%]
enzdvdd (coverpoint)	2	2	2	100	100
dislvl (coverpoint)	2	1	1	100	100
dissink (coverpoint)	2	2	2	100	100
iomread (coverpoint)	2	2	2	100	100
iomsw (coverpoint)	4	4	4	100	100
vfbread (coverpoint)	2	2	2	100	100
di (coverpoint)	11	11	11	100	100
test (coverpoint)	9	9	9	100	100
cx_test_di (cross)	88	15	88	17.05	100
Coverage Score				86.58	100

- ***cx_test_all*** - This cross is ment to test all test scenarios for all digital interface bits. The test coverpoint is not considered.

TABLE III
CP COVERAGE RESULTS.

Variable	Expected	Uncovered	Covered	Cov. percentage[%]
disdvdd (coverpoint)	2	0	2	100
dislvl (coverpoint)	2	0	2	100
dissink (coverpoint)	2	0	2	100
mode (coverpoint)	2	0	2	100
swilim (coverpoint)	2	0	2	100
dttrim (coverpoint)	4	0	4	100
test (coverpoint)	12	0	12	100
cx_test_all (cross)	32	21	11	34.38

Similarly to the LDO, directed stimuli is considered to functionally verify the design, as a standard example testbench adopted at SiliconGate is replicated in UVM. The coverage score was 93.44%. Once again the cross presents low coverage results. Tests are conducted aiming for functional verification and unusual input combinations are not exercised as these are not considered in the verification plan.

Once the several pseudo-random stimuli are introduced in the environment, 100% coverage is achieved. An issue was found in the design of the CP. *Anatestreq* output bit is set to 1 when in test mode. For a *dislvl* set to 1, level converters inside the model are expected to output default values. This

means that the start-up sequence is not complete (*porz* = 0) and reference voltages in all domains are not well defined. Therefore *anatestreq* should be set to its default value of 0. This bug was corrected in the RNM model.

VI. Digital Core Results

The proposed architecture, described in section IV-E, was implemented. A covergroup for the SPI environment was assembled. It possesses a coverpoint for *spics* variable and spi_address variable. Considered coverpoints, crosses and respective results are shown in table IV. cx_test_all crosses both presented coverpoints.

Once again the results are presented for a SV standard flow with a coverage score of 66.67%. Several random test scenarios are then created and exercised on the DUT in order to fully cover all the possible combinations of input signals for voltage regulation.

TABLE IV
Digital Core Coverage Results

Variable	Expected	Uncovered	Covered	Cov. percentage[%]
spics (coverpoint)	2	1	1	50
spi_address (coverpoint)	14	0	14	100
cx_test_all (cross)	28	14	14	50

In the standard flow, tests such as writing with a high *spics* were not considered. In this scenario, the input bus should not be driven to the registers as the slave is disabled. UVM enables this tests in an extensive fashion with full coverage for the considered coverpoints. Alongside these write operations, reads are also implemented immediately after for the same address. This ensures that reads and writes are tested for all addresses for an enabled and disabled slave. The scoreboard validates the digital core for all considered inputs.

VII. PMU Core Results

The PMU architecture UVM environment (Figure 6) is implemented. As all voltage regulators and digital core were thoroughly tested and verified, the integration and testing of the PMU is facilitated. Directed sequences are considered to test the startup sequence. For this purpose default values are written in all registers, and the APC is enabled in order to generate the enable signals for the regulators. Figure 8 shows the start-up sequence, depicting enable signals and their respective *pg* indicator, followed by the rise of the *porz* indicator. Voltage regulators output voltages are also shown.

A. Edge Cases

The purpose of these tests is to verify if the parameters edge cases are correctly implemented in the voltage regulators of the PMU. Therefore maximum and minimum programming codes for control bits are set to assess their impact. The LDO output voltage is programable with the *di* input digital variable.

Edge cases for *di* are assessed for LDOs that define input voltages for internal regulators (LDO1 and LDO3).

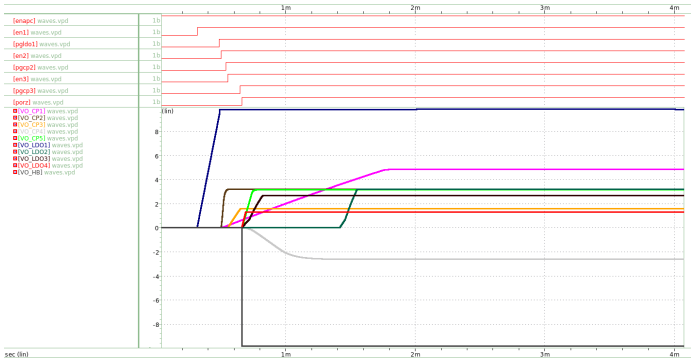

Fig. 8. PMU's start-up sequence.

B. Edge cases results

Figure 9 and 10 displays the *vo* output voltages for both LDOs (LDO1 and LDO3) and the output voltages of the regulators connected to them.

For the edge programming voltages, all subsequential regulators perceive these changes but still operate normally, as expected.

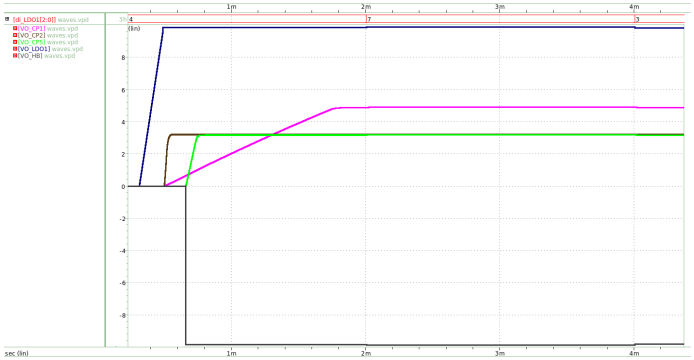

Fig. 9. Edge cases for maximum and minimum LDO1 output voltage.

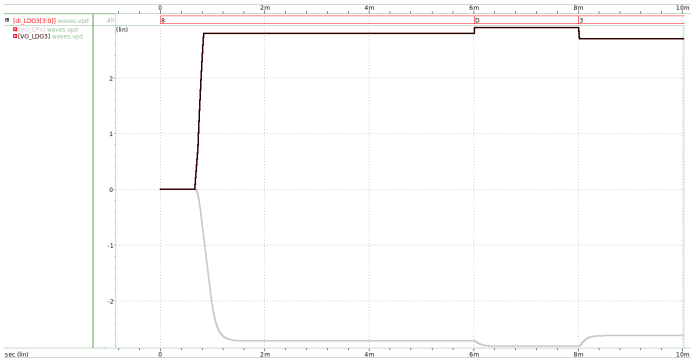

Fig. 10. Edge cases for maximum and minimum LDO3 output voltage.

VIII. Conclusions

UVM-based mixed-signal architectures were presented for a PMU and its constituents and compared to traditional SV testing. Alongside RNM, UVM provides the infrastructure to

test several scenarios in an automatic fashion while simultaneously validating the behavior of the DUT and outputting meticulous reports. Functional verification is simplified and, when aiming for coverage goals with random input stimuli, unexpected misbehavior may be detected by the scoreboard component, as shown in the CP model (section V-B). This feature is much appreciated as finding these faults at earlier design stages is crucial for the success of the project. UVM was implemented in SiliconGate's test flow.

UVM UVCs were successfully reused throughout the verification process as they were intentionally built for such purpose.

Voltage regulators (mixed-signal RNM models) were validated for all considered input combinations of the digital interface ports. For the digital core, an exhaustive amount of tests were conducted. Being this design purely digital, tests simulate much faster when compared to RNM models. Coverage was also assessed to ensure that all addressable registers were exercised for random input data. The scoreboard component in these environments, alongside the reference model, play an important role as validation automation is achieved, and allows the generation of self-checkable processes. Testing of these components separately eased their integration in the scope of the PMU core. A start-up validation was performed and edge cases internal voltages were verified.

The work developed in this thesis resulted in a paper accepted for poster presentation in the Conference on Design of Circuits and Integrated Systems (DCIS) proceedings.

IX. System Limitations and Future Work

The UVM class library, although very flexible and well structured, is very complex. UVM is not recommended for small projects as the overhead of building the environment is not justified by its gains. It excels for big projects where a well-planned verification environment can be implemented to speed-up the verification process. Besides, UVM does not provide a direct link from testbench sequences and code running at the SoC level.

The only approach to hit coverage goals is to increase the number of samples or change the seed. A more robust sequence generation mechanism could be explored in order to reduce the occurrence of repeated sequence-items, allowing for better coverage-closure. An approach to this issue was assessed in [18].

A UVM register model will be considered as an extension of this work. The register model is a hierarchical structure of objects that contains the description of the register on the DUT.

Master-slave topologies will be considered. This allows the user to dynamically generate UVM environments in a centralized fashion, granting an extra layer of controlability. Factory features and further configuration with a the UVM configuration database will be explored as well.

Acknowledgment

This work was supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020 and pAvIs, PENTA Project n. 20016.

References

[1] C. Sapsanis, M. Villemur, and A. G. Andreou, "Real number modeling of a sar adc behavior using systemverilog," in *2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2022, pp. 1–4.

[2] S. Balasubramanian and P. Hardee, "Solutions for mixed-signal soc verification using real number models," in *Cadence, Tech. Rep.*, 2013.

[3] Penta, "pavis - penta," 2022, https://penta-eureka.eu/project-overview/penta-call-5/pavis/, Last accessed on 2022-09-16.

[4] M. M. Ron Vogelsong, Ahmed Hussein Osman, "Practical rnm with systemverilog," in *CDNLive2015*, 2015.

[5] N. Georgoulopoulos and A. Hatzopoulos, "Real number modeling of a flash adc using systemverilog," in *2017 Panhellenic Conference on Electronics and Telecommunications (PACET)*, 2017, pp. 1–4.

[6] N. Georgoulopoulos, A. Mekras, and A. Hatzopoulos, "Design of a systemverilog-based vco real number model," in *2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, 2019, pp. 1–4.

[7] B. Vineeth and B. B. Tripura Sundari, "Uvm based testbench architecture for coverage driven functional verification of spi protocol," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2018, pp. 307–310.

[8] C. Elakkiya, N. Murty, C. Babu, and G. Jalan, "Functional coverage - driven uvm based jtag verification," in *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, 2017, pp. 1–7.

[9] T. M. Pavithran and R. Bhakthavatchalu, "Uvm based testbench architecture for logic sub-system verification," in *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*, 2017, pp. 1–5.

[10] K. Salah, "A uvm-based smart functional verification platform: Concepts, pros, cons, and opportunities," in *2014 9th International Design and Test Symposium (IDT)*, 2014, pp. 94–99.

[11] N. Georgoulopoulos, I. Giannou, and A. Hatzopoulos, "Uvm-based verification of a mixed-signal design using systemverilog," in *2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2018, pp. 97–102.

[12] Y. Liu, N. Tan, X. Xiao, J. Xia, W. Hu, and Y. Ding, "Design and uvm verification of an rtc subsystem with temperature compensation," in *2021 6th International Conference on Integrated Circuits and Microsystems (ICICM)*, 2021, pp. 384–389.

[13] M. Soares, M. Santos, and J. Munhão, "Universal verification methodology for voltage regulators," accepted for poster presentation in XXXVII Conference on Design of Circuits and Integrated Systems, 2022.

[14] C. Liang, G. Zhong, S. Huang, and B. Xia, "Uvm-ams based subsystem verification of wireless power receiver soc," in *2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, 2014, pp. 1–3.

[15] W. Ramirez, H. Gomez, and E. Roa, "On uvm reliability in mixed-signal verification," in *2019 IEEE 10th Latin American Symposium on Circuits & Systems (LASCAS)*, 2019, pp. 233–236.

[16] A. I. Cianga and C. Tepus, "Morphing digital functional verification to meet mixed signal challenges," in *2014 International Semiconductor Conference (CAS)*, 2014, pp. 219–222.

[17] Accelera, "Universal verification methodology (uvm) 1.1 user's guide," 2011.

[18] K. Fathy, K. Salah, and R. Guindi, "A proposed methodology to improve uvm-based test generation and coverage closure," in *2015 10th International Design & Test Symposium (IDT)*, 2015, pp. 147–148.