A Framework for AMS Verification IP development with SystemVerilog, UVM and Verilog-AMS

Mike Bartley, TVS, Bristol, UK

Jeganath Gandhi R, TVS, Bristol, UK

Abstract

Analog Mixed Signal (AMS) Design Verification methodologies of different abstraction levels are in practical use by different chip manufacturers and service providers. Yet, there is no standard methodology or re-usable component for AMS verification. This paper outlines an efficient, re-usable, AMS Verification IP (VIP) development strategy including verification plan, architecture of an AMS VIP, coverage collection and signoff for an AMS design, which includes a digital RTL and Analog Model. The VIP leverages concepts from UVM VIP and extends it to the AMS verification with Verilog-AMS to interact with analog components of the design interface. This extension addresses driving stimulus, assertions, coverage collection for these components.

The VIP development strategy outlined in this paper features below phases and components:

- **Feature extraction from AMS Design Specification.**
- **Analyze DUT components: Digital RTL and Analog AMS model.**
- **Creating a VIP development plan from the extracted feature list and digital and analog design partition.**
- **UVM based VIP components including Verilog-AMS BFM to interact with Analog interfaces.**
- **Real value monitor based checks**
- **Coverage Collection and Signoff**

Keywords—AMS Verification; System Verilog UVM, Verilog-AMS, metrics driven verification.

INTRODUCTION

The suggested methodology does not intend to replace Analog verification. This methodology should co-exist with a strong module level verification of the Analog components, focusing on its parameters and analog behaviors. Digital verification and its interaction with the analog world is the scope of this methodology.

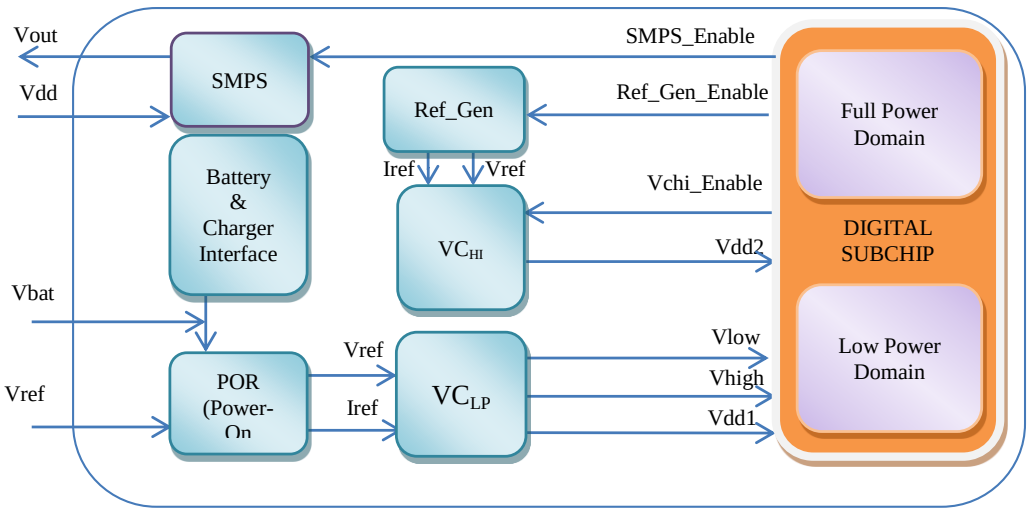

Figure 1: AMS PMIC DUT

In this paper, our example DUT is a Power Management Integrated Chip (PMIC), as depicted in figure 1. The PMIC includes a Power management state machine, that drives the control signals to enable the different voltage

regulators, configuring them in low power mode, based on the current battery voltage level, generating system resets, clocks, responding to interrupts from the host processor. Thus it includes a digital component to interact with the system on Chip, while the analog component interacts with different regulators, bias generators and comparators.

A. Feature Extraction

Feature extraction is typically a Verification engineer's activity, driven by the functional specification,wherein the DUT features are listed down in a way that, they can be used as a metric for test development, checker development, coverage modeling and verification signoff. For instance, PMIC includes a Power on Reset module whose sole purpose is to assert or de-assert the system reset, when a battery level crosses a specific threshold. The feature extraction for this function would be list out the stimulus, Voltage input level, threshold for reset assert, and the hysteresis threshold for the reset de-assert. Example:

- When battery voltage is below 1.5v the state machine is power cut state and none of the LDOs are enabled
- When battery voltage is above 1.5v and below 3.0v the state machine is Low power and VCLP alone is enabled to power the low power domain.
- Full Power state is when batter voltage is above 3.0v
- VChi is enabled only when in Full power state.

This would then guide the test creation to drive stimulus ranging between these thresholds. The checker can use a look up table of thresholds and relative system behavior. The coverage model uses all related inputs as cover points, and any crosses as necessary.

While a typical functional specification lists out the system functions, it does not give a point by point mapping between set of inputs and outputs, which primarily drives your verification activity. Thus a well written feature extraction list is pivotal in driving the verification plan, test execution and closure.

B. The AMS DUT

The significant difference in an AMS Verification flow from the traditional digital only verification flow is the continuous discipline seen in the Analog components used as mode. While a technical specification details the system behavior, there is no standard in place to communicate Analog component model's specification. . Apart from the technical evolution obstacles, the model is more to aid the verification than the help design activity and thus seen as a low priority activity in a schedule that focuses on time to market. On the other hand, knowledge of the AMS model in digital perspective is mandatory to the planning of an efficient verification IP development.

In our example of a PMIC, the LDO output voltage is a function of the control signals from the Digital Sub chip, the bias currents from the bias generator, the reference voltage, see figure 1. The modeling techniques employed to develop the bias generator, reference generator, the Voltage regulator involving the ramp timings, delays modeled are important in designing an accurate Self-Checking VIP. Once the digital part of the design drives the control signals, depending on the accuracy of the analog models, the LDO voltage change can happen instantaneously or after incurring delays to the different modeling components. This continuous behavior has to be taken into account while sampling the analog output in the VIP monitor. The example code of POR VAMS models is shown in Appendix.

Similarly, while measuring a frequency of an analog signal, we might have to drop few samples immediately after power up, depending on the models settling time.

C. The UVM environment

The VIP uses all of the concepts at the higher level from the standard UVM architecture [1]. UVM provides the framework for the automatic test generation, self-checking test benches and coverage-driven verification. Automatic test generation and coverage-driven verification provide us an efficient way to measure verification target achieved/required effort.

A UVM test bench is composed of reusable verification components in SystemVerilog. The verification components include a transaction item, driver, sequencer, monitor, agent and the encapsulating environment.

The difference in the digital VIP and an AMS VIP is the interaction of the monitor and driver with the analog components as detailed below.

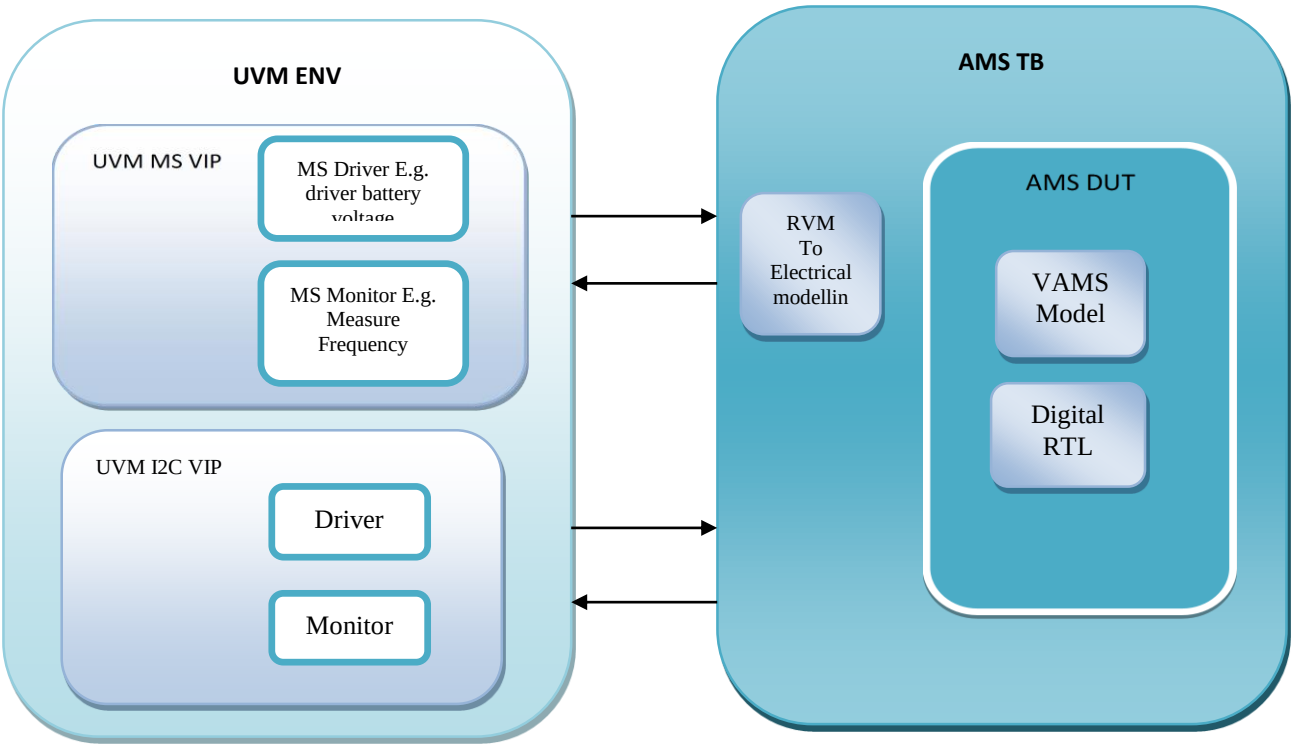

Figure 2: Architectural Block Diagram

D. Mixed-signal (MS) Driver

The scope of the MS driver, part of the AMS VIP, is to interact between the Digital components of the VIP and the analog components of the DUT. Note that this MS driver is not containing analog signal processing, but treats the signals as discrete-time real value numbers. The actual D/A conversion to the electrical signal is done in the Verilog-AMS domain by means of real-value-modeling (RVM) techniques. These are the connect modules that form the interface between the real valued UVM environment and wreal/electrical analog disciplines. Connect modules are used to connect the continuous and discrete disciplines (mixed nets) of the design hierarchy together. A connect module defines the conversion of logic for a specific types of disciplines that it bridges (E.g : real to electrical). Note that this conversion is required only when the UVM environment interacts with the modules having continuous discipline ports and the connect module insertion is automatic. Connect modules are not instantiated if the UVM environment interacts only with the discrete domain. The DUT modeling is beyond the scope of this VIP.

```
Class por_item extends uvm_sequence_item;

rand real vbat;
rand real vref;
rand real iref;
constraint c1 { vbat inside { [0.0:5.0] }; }
constraint c1 { vref inside { [0.0:0.1] }; }
constraint c1 { iref inside { [0.0:0.1] }; }
…<uvm factory registration>…

endclass :por_item

interface por_intf();
real vbat;
real vref;
real iref;
endinterface: por_intf
```

```
Class por_driver extends
uvm_driver#(por_item);

Virtual por_intf por_if;

…<uvm factory registration, phases>…

Task run_phase(uvm_phase phase);
fork
forever begin
seq_item_port.get_next_item(req);
por_if.vbat  =req.vbat;
por_if.vref   = req.vref
por_if.iref   = req.iref;
$cast(rsp, req.clone());
seq_item_port.item_done(rsp);
end
join
endtask :run_phase
Endclass :por_driver
```

Listing 1: System Verilog UVM MS Driver

In our example DUT, the battery terminal needs to be driven with real values at the chip boundary. At the same time, driving digital values from the test case, simplifies the test creation, self-checking mechanisms and coverage collection. The MS driver does the job of converting the digital values from the sequence driver to analog values and driving them on the DUT.

E. Sampling, Checking, Assertions

The self-checking mechanisms, assertions work on the samples received from the MS monitor. For example, we might want to have a self-checking mechanism on a VCO output, which is a continuous analog waveform. Existing language constructs doesn't allow us to have a continuous frequency check assertion on an analog signal. Solution is to convert the value to digital, perform some computations on the values as the sample code below.

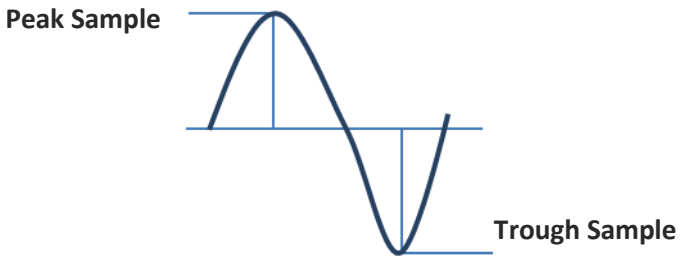

Figure 3: Peak and Trough Sample Points

In the above example, sampling of the analog signals in the environment has to take into account the maximum frequency at which the design can operate and set the sampling frequency, in order to be able to avoid sampling errors. (Sampling theory and the effect of aliasing)

```
virtual task clock_monitor();
var time prev_time;
var real ldo_output;
var real peak = 1.0;
var real trough = 0.0;
var real cur_sample;
var real prev_sample;
var real prev_2_sample;
var bit first_peak = 0;

@(posedge vco_if.clock)
forever begin
@(posedge vco_if.clock)
cur_sample = vco_if.ldo_out;
if(cur_sample == peak) begin
first_peak = 1;
end else if(first_peak == 1) begin
// Frequency Monitor
if(cur_sample < prev_sample && prev_sample
> prev_2_sample) begin
peak_q.push_back(prev_time);
end
if(cur_sample > prev_sample && prev_sample
< prev_2_sample)
trough_q.push_back(prev_time);
```

```
end
prev_2_sample = prev_sample;
prev_sample = cur_sample;
prev_time = $time;

if(peak_q.size() != 0 && trough_q.size !=0)
begin
clk_period = (trough_q[0] > peak_q[0]) ?
(trough_q[0]  - peak_q[0]) : (peak_q[0] -
trough_q[0]);
if(!(clk_period > exp_period - err_margin)
&& (clk_period < exp_period + err_margin))
uvm_report_error("ERR_PERIOD",
$psprintf("Unexpected Time Period :: %0t -
Expected :: %0t", clk_period, exp_period));
end
end

endtask : clock_monitor
```

Listing 2: System Verilog UVM MS Monitor

The above is an example of continuous value based checker. We could have an assertion based on the values from an analog model, which changes based on control signals from digital world.

For example, the comparator output should change within a couple of cycles of the digital enabling the comparator. The analog output is fed through an Analog to Digital convertor in the environment and the monitor has an assertion on the input digital and the ADC digital output.

Figure 4: Waveform depicting AMSVCO Output

F. Coverage Collection

As with any verification effort, the verification goal, the achieved target has to be set or measured respectively. In case of an AMS environment, this is done at the higher layer of the VIP, making it completely transparent to the analog world. The coverage model operates on the inputs and outputs where applicable, on the data from monitor, while the analog-digital conversion is taken care outside the coverage model or collection.

The example Battery voltage coverage along with its trim value to test a VAMS comparator with use of system Verilog RVM is shown below. The conversion between electrical and real-value is taken care at the test bench level.

```
logic [1:0] vbat_trim;
real vbat;

covergroup comparator_cg
@(posedgevco_if.clock);

vbatvalues :coverpoint vbat {
bins power_cut = {[0.1:0.5]};
bins low               = {[0.6:1.5]};
bins full_power        = {[1.6:5.0]};
}

Comp_cross : cross vbat_trim, vbatvalues;
endgroup
```

Listing 3: System Verilog MS Coverage Definition

G. Conclusion

This paper has presented a framework for AMS VIP. There has been a constant evolution in terms of digital methodology, digital tools and the architecture proposed here leverages them to efficiently verify an AMS design. Apart from verifying the Full chip with Analog and Digital domains early in the design cycle, can be used as proof concept for the Analog architecture while the design evolves. UVM and Metrics based verification

helps to bring the methodology based verification to AMS designs and thus reduce cost and increase efficiency. The advantages include, random test generation, self-checking environment, assertion based verification and coverage-driven verification, faster simulation time. The results show that this a scalable architecture for complex AMS designs.

APPENDIX
POR VAMS Model

```
`include "disciplines.vams"
Module        por(vbat,vrefin,gnd,vlow,vhigh,
vclp_vref, vclp_iref);

// Parameters
// Threshold below which vlow goes high and
above which vlow goes low.
parameter real vlow_threshold = 1.5;
// Threshold below which vhigh goes low and
abovew which vhigh goes high.
parameter real vhigh_threshold = 3.5;
// Threshold below which vhigh goes low and
abovew which vhigh goes high.
parameter real vref_input_p = 1.5;

// Define input/output
Input vbat, vrefin, gnd;
//vbat: voltage at the battery terminal.
//vrefin : Input reference voltage for the
comparator
//gnd : ground terminal

Output vlow, vhigh, vclp_vref, vclp_iref;
//vlow : 1 - battery voltage lower than low
voltage threshold
//vhigh : 1 - battery voltage higher than
minimum value for full chip operation

//Define port types
Wreal vbat, vrefin, gnd;
Wreal vclp_vref, vclp_iref;

Reg vlow, vhigh;

Real vclp_vref_r, vclp_iref_r;
```

```
initial begin
vclp_vref_r = 0.0;
vclp_iref_r = 0.0;
vlow = 1;
vhigh = 0;
end

// perform logical assignment of the control
outputs based on input battery voltage and
reference and ground inputs.
always @(vrefin, gnd, vbat) begin
//if(check_levels(vrefin,                gnd,
"REFERENCE_VOLTAGE",     "GROUND"))     begin
//check_levels is a library function used to
check reference voltage levels and ground
connection)
//The library of analog checks helps us to
simplify generic checks.
if(vrefin == vref_input_p&gnd == 0.0) begin
if(vbat>vlow_threshold&vbat<vhigh_threshold)
begin
// When the battery level is higher than the
vlow threshold, but lesser than vhigh
threshold, the device is just out of low
power.
vlow = 0;
vhigh = 0;
vclp_vref_r = 1.0;
vclp_iref_r = 0.1;
end
else if (vbat>vhigh_threshold) begin
// When the battery level is higher vhigh
threshold, the device is full power and
stable state.
vlow = 0;
vhigh = 1;
vclp_vref_r = 1.0;
vclp_iref_r = 0.1;
end
else if (vbat<vlow_threshold) begin
// When the battery level is lower than the
vlow threshold, this indicates a loss of
power.
vlow = 1;
vhigh = 0;
vclp_vref_r = 0.0;
vclp_iref_r = 0.0;
end
end
else begin
//Check interconnection or reference voltage
check failure
vlow = 'bx;
vhigh = 'bx;
vclp_vref_r = 0.0;
vclp_iref_r = 0.0;

end
end

assign vclp_iref = vclp_iref_r;
assign vclp_vref = vclp_vref_r;
endmodule
```

References

[1] Accellera, Universal Verification Methodology (UVM) 1.1 User's Guide May 18, 2011.