Mixed Signal Validation from a Full-Chip Perspective

Gavin O'Donoghue (gavin.o'donoghue@intel.com)

Intel
Shannon, Ireland

www.intel.com

ABSTRACT

This paper will discuss the requirements and learnings in validating Analog IP's (AIP) at Full-Chip (FC) SoC level using Mixed-Signal Validation. Validation at FC level which includes analog blocks, was mainly completed using their behavioural models. Often, in order to simplify validation, the default flow at SoC level, is to bypass some of these models within the AIP. To mitigate against the possibility of issues arising including incorrect models or connectivity, the ability to test at FC level in a mixed-signal environment is becoming ever more important.

There is generally a high level of Mixed-Signal Validation completed at an AIP level before integration into the SoC. Our primary focus is to ensure correct functionality and connectivity between the FC model and the AIP as spice, and not necessarily focused on the validation of deep analog circuitry within the AIP. The findings will be discussed and any important learnings will be presented.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

This paper will introduce and discuss our methodologies and experiences in enabling mixed-signal validation (MSV) at a System on Chip (SoC) level. It was our division's first time in implementing such a solution. Previously we relied solely on digital validation using the Behavioural Model (BMOD) of a particular Analog-IP.

Although it was our first time in implementing this solution, it had occurred elsewhere within the organisation. Some previous issues caught by the use of MSV at SoC level include:

- PLL state machine not validated due to forces in digital validation.
- SERDES micro-partition bypassed in digital validation.
- Glitches on PI clocks that caused data corruption.
- Initialization bug in power gate state machine.

The majority of the blocks we were validating had undergone significant AMS validation at their unit level. The aim in performing MSV at Full-Chip level, was not necessarily to re-validate internal analog components of the IP, but to ensure that nothing from a Full-Chip perspective 'broke' the IP during integration.

However, as part of our test plan there was also a requirement to re-verify critical specifications such as - PLL lock time and frequency measurements of the VCO clock and divided clocks. Other important test plan items included the power up sequences of the IP, and any specification related to the data flow through RX and TX components. Example, Vp-p, and Vcm voltages.

Below details the methodology, results, issues that arouse and details possible future improvements that would be made should we to continue in performing this validation at a SoC level.

2. Tools and Methodology

The following tools were used to complete our test plan.

VCS: The SoC digital simulation tool.
CustomSim: For Spice simulation.
Discovery AMS: VCS and CustomSim co-simulation plus methodology.
DVE: For digital waveform viewing.
nWave: For Analog waveform viewing.

We were able to fully utilise the SoC digital validation environment when performing a co-simulation run. Typically, with one modification to the compile command supplied by the digital validation team, the spice swap could be included.

This was enabled by using the switch: *-elab_opts "-ad=<block>.init"*

The three main setup files required for the co-simulation run were <block>.init, <block_top>.sp and <block>.cfg

After some initial trial and error, which is detailed further on, the final content of all these files almost entirely originates from the unit level MSV team of the IP. With the addition of some SoC specific hierarchy pointers we could utilise all these files.

A detailed review of the RTL defines within the AIP was required. It was necessary at times to include various defines that would enable certain features not seen in digital simulation e.g. Analog BMODS, some power pins etc. If required we would then include these defines with the *–vlog_opts "+define+<required_define>"* switch.

Schematic capture and editing was outside of the SoC environment but was used by the AIP team. We did utilise their environment when netlisting or editing to the schematic was required.

The main IP that was validated and reported in this paper is a High-Speed IO (HSIO). At a high level it comprised mainly of three analog components: RX, TX and a PLL, along with corresponding digital control logic. Our methodology was to utilise a Full-Chip digital test that exercised these components. For this we used a traffic test that passed packets of data on both the RX and TX paths. All 3 components were not swapped at once in any one co-simulation run due to run time slowness, and the potential size of waveforms that would be generated.

Also for our purpose, calibration of the IP was bypassed. Full calibration may have lasted many milliseconds at SoC level. This was not feasible to complete when performing mixed signal at SoC level. For this we worked with the AIP team in correctly by-passing the calibration.

Below Figure 1, indicates an outline of our initial AMS flow.

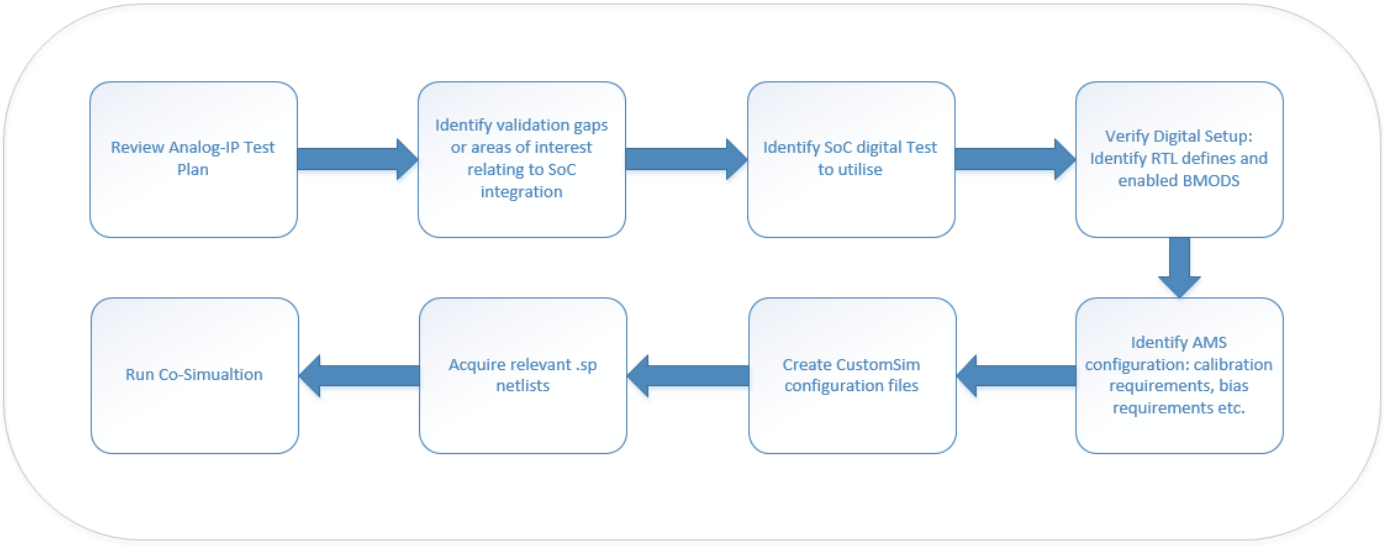

Figure 1: Initial AMS flow

As previously mentioned, our main aim at SoC level, is not to verify specific analog components deep within the hierarchy of the AIP e.g. data samplers, bias generation etc. Our expectation is that the AIP team fully validate this and we also review their test plan.
Our primary focus is ensuring correct start up sequences, correct data flow through the IP, and correct interface connection with the SoC.

3. Issues Encountered

A number of issues arose during the project, which caused delays or debug at a particular time. Some of these may have been project or integration related, while others where tool related. Below is a description of some and in no particular order of severity:

3.1 IP integration into SoC

Even though the AIP team may release their block for integration into the SoC following the completion of a particular milestone at unit level, this does not necessarily mean that the SoC team are ready to integrate it. On a number of occasions we were aware that the spice netlist and BMODS we were using had since been modified. When debug was required and where we asked for the AIP teams help, they preferred that we were always using the most recent release, as perhaps the issue we were seeing was no longer valid in the new netlist.

Also, early in the release process of the AIP, they do not release a spice netlist. It therefore required the use of their environment to netlist the block independently. Typically their team would instruct which tag to use, however on occasion this did not match the version of BMODs released to SoC. As such there was a number of times where issues like pin-mismatches occurred between BMOD and spice netlist in the SoC environment. Although these are minor issues, at times they caused us unnecessarily delays in debug and setup.

3.1 Port Direction

As part of the mixed signal run, there is a report generated that describes the port interface properties of a particular block or sub-block that was swapped from RTL to spice. This is known as interface_element.rpt. Here, the port direction is described, as well as the either digital or analog threshold values used for conversion from one domain to another. A number of instances arose whereby an incorrect conversion was applied. For instance, when an output pin should be going from the analog to the digital domain, the threshold implied in the file was ‘hiv=1v’ and ‘lov=0v’, meaning the tool thought it was a digital to analog conversion. This was despite the fact that both the spice and RTL had the pin as output.

To override this we had to use the *port_dir* command in the <block>.init file where you explicitly specify the direction of the ports [1]. When the affected port directions were set with this command, the correct domain conversion mapping occurred.

3.3 Test Availability

We were entirely dependent on the digital validation team for test availability at SoC level. On occasion the required test that would satisfy our test plan was either not developed or still a work in progress.

Also, typically to ensure full coverage, the digital tests utilise a random feature. For example if it was a traffic test on a multi-I/O IP, the test may randomise which port was enabled. However this did not benefit us due to the inability to match that random selection with our CustomSim setup file. Therefore we had to manually edit the test and statically assign which port was enabled in each test.

3.4 Interaction with IP teams

For the most part, each of the analog blocks we were validating at SoC level had MSV teams at their levels also. They would typically be much further along their validation at unit level than we would have been at SoC level. As a result they had the majority of the various setups available.

Initially when preforming the setup at SoC level, we approached this in an isolated manner. There were occasions where we overlooked certain setup criteria, e.g. missing biases, missing R + C's, and we spent probably unnecessary time in debug to get the setup right. Once contact with their MSV team was established we were then able to fully understand the various setup requirements.

In all cases the setup received from the AIP team could be easily transferred to SoC level by simply updating various hierarchy pointers. The AIP teams were also extremely useful in providing debug support.

3.5 AIP collateral

We had issues initially whereby we were not aware that the inductor models used by the IP team, were their own models and not the standard cell versions available in the SoC environment. This caused problems when performing simulations on the PLL and TX/RX where inductors are used in the path of the receiving/transmitting data. The default netlist performed by the IP team was to include these models. This was put into their flow by their automation engineer, so during initial contact to discuss the issue they were unaware of this. Once the issue was isolated, it required a manual hack to the netlist on our part to include these models as they were not available in the SoC environment.

The AIP team also used Channel Models in their mixed signal simulations on RX and TX paths. In SoC environment these were not enabled as the Digital Validation team did not require them when using BMOD. It was necessary for us to manually instantiate these models in our local SoC environment.

In addition, the IP team had an extra module connected to the RX path that mimicked the TX FIR output of the HSIO TX block. This was built into the AIP unit level environment and during

initial contact with AIP team they were not aware of this module. Again once the issue was isolated, we were required to manually instantiate this model into the SoC.

Note: These models were never checked into the SoC environment and made permanent as they would have caused issues in digital validation. Therefore we manually instantiated them for each checkout of the SoC environment we performed.

3.6 PLL

Initially there was a lot of issues encountered in enabling the DCO to oscillate in any FC simulation when it was swapped as spice. The hierarchy of the PLL_EBB_TOP can be seen below in Figure 2.

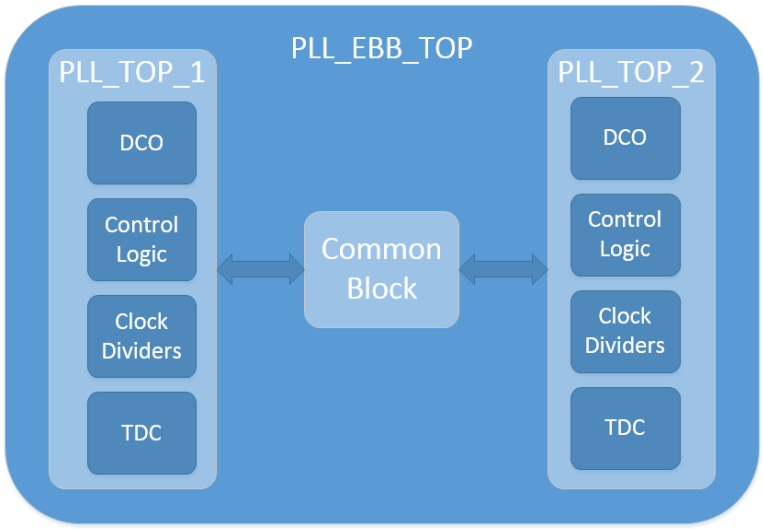

Figure 2: PLL Hierarchy

Simulating two PLL's would have significant impact on performance, therefore the DCO in only one PLL_TOP hierarchy was swapped at a time. During simulation, various issues arose which caused multiple debug and trials to be run. The below Table 1 describes some of the trials and details what component was swapped for spice or remained as RTL.

Initially, little consideration was given to how each sub-block was swapped as spice, or at what hierarchy should the spice swap begin. For instance it was never discussed should the PLL_TOP be swapped in full as spice and the Control Logic then be swapped back using the *use_verilog* command. Or should the full PLL_EBB_TOP be swapped as spice and any blocks required to be as BMOD, then swapped in.

Setup	Swapped as Spice	Blocks Remaining as RTL/BMOD	Result
1	PLL_TOP_1/DCO PLL_TOP_1/TDC PLL_TOP_1/Clock_Dividers	PLL_EBB_TOP PLL_TOP_2 PLL_TOP_1/Control_Logic Common Block	PLL unable to oscillate
2	PLL_TOP_1/DCO PLL_TOP_1/TDC PLL_TOP_1/Clock_Dividers Common Block	PLL_EBB_TOP PLL_TOP_2 PLL_TOP_1/Control_Logic	Voltage- Inductor Loop induced
3	PLL_EBB_TOP	PLL_TOP_2 PLL_TOP_1/Control_Logic	PLL oscillated, but no valid PLL Lock due to calibration issues.
4	PLL_EBB_TOP	PLL_TOP_2 PLL_TOP_1/Control_Logic PLL_TOP_1/DCO	PLL oscillated and locked as BMOD

Table 1: PLL Trial Setups

The first simulation approach was to swap the individual elements within one PLL_TOP hierarchy as spice. The Control Logic stayed as RTL as it had no spice equivalent. The Common Block was kept as BMOD and the biases were applied manually in <block_top>.sp. However we were not able to achieve oscillation in this manner. After direction from the IP team, they recommended to include the common block as spice instead of applying the biases manually.

The second approach added the Common Block as spice. However when trying to simulate, this induced a voltage-inductor loop, and the simulation would crash. Again, based on feedback from the IP team, it appeared to be induced due to the various hierarchy levels that were swapped out and their interaction with the declared VCC power nets that were included in the <block>.init file.

The third approach, did enable the PLL to oscillate. For this the full PLL_EBB_TOP was swapped as spice, with the second instance of PLL_TOP remaining as BMOD, along with the Control Logic in PLL_TOP_1. However we were unable to achieve a valid PLL lock signal in this particular simulation. After further debug with the IP team, it appeared that this occurred due to incorrect calibration bypass setup received from the IP team and the incorrect bias current and gain values generated as a result.

Finally, after reviewing a number of criteria such as, the modification to the setup that would have been required, the stage in the project we were at, and taking into account that the IP team were able to achieve a valid PLL Lock signal in their unit level AMS validation, it was decided that the best course of action was to keep the DCO as BMOD.
This then negated the impact of incorrect bias current generation in the setup and allowed the PLL to oscillate and lock as BMOD. It also enabled us to avoid use of large number of forces that may have been required to achieve correctly bypass the calibration in the Common Block.

As there was lot of trials and debug during initial phase, and because compilation may have taken 24 hours, plus another 48+ hours until oscillation was potentially reached, we also utilised a debug technique, as detailed in Appendix A.

For this, we first ran the VCS traffic test with all PLL BMODS enabled. The resulting .vpd wave file was converted to a .vcd with a hierarchy pointer to the level of the DCO initially, and then ultimately concluding at PLL_EBB_TOP level after initial trials. A resulting Testbench was created and we were able to swap the DCO as spice using the normal CustomSim techniques. This flow is extremely fast in simulation, and we achieved pll_lock in <1 day compared to spice swap of potentially 2+ days.

4. Results

4.1 Simulation time

Performing mixed signal validation at a Full-Chip level had its challenges in terms of run time. The below Table 2, describes the run time differences between running the same test on the HSIO, under different scenarios:

Run Type	Simulation Run Length	Netlist Element Count
(1) VCS – Ceratin BMODS disabled	15hrs	
(2) VCS – ALL BMODS enabled	31hrs	
(3) VCS-AMS – RX as spice	10days	700,000
(4) VCS-AMS – TX as spice	3.5days	700,000
(5) VCS-AMS – PLL_EBB_TOP as spice*	2days**	35,000

Table 2: Run Length Comparison

Run Type 1 above, was the default VCS run by SoC team. For this they bypassed some of the BMODS on the RX path, and some PLL BMODS were also bypassed. The pll_lock signal was based on a timer.
The second mode was a VCS run with the RX BMODS and PLL BMODS fully enabled. We enabled these through the use of certain RTL defines. The remaining three runs were mixed signal runs whereby either the RX, TX or PLL of the IP was swapped as spice. Also detailed is the spice element count which details the usage of diodes, NMOS, PMOS and resistors and capacitors.

We did not run any simulations where-by all 3 components were swapped as spice at the same time. Also, the run length time does not include compile and elaboration. This typically took 24hrs for spice swap run.

* The PLL DCO was kept as BMOD. The PLL Common Block, Dividers, and TDC were kept as spice.
** The PLL simulation was killed approximately 100us after PLL lock was achieved, at 260us simulation time. All other simulations completed at approx. 700us.

4.2 Overall Test Results for HSIO

We were able to get a 'pass' result in all tests ran with a spice replacement. However because of the lack of use of AMS checkers, we also heavily relied on waveform reviews to satisfy the various criteria in our test plan.

In the RX spice simulation, although the overall results from VCS was a pass, on closer inspection, some elements in the test plan failed. For example certain Vp-p and Vcm specifications, as the RX data passed thru the various components of the RX, failed. On further debug with the Analog IP team, it was noticed that some of the calibration was not correctly set and some gain values and timers were not correctly set for our use case. This occurred due to a lack of awareness on the forces that were applied by the SoC digital team. As they ran digital test with majority of BMODs disabled, the values they were forcing had no real meaning or effect in their simulations. Once it was discovered, the correct forces were applied, and our test was re-run.

The TX spice swap had relatively few issues in setup and debug. Once the correct bias was applied we could see correct data travel thru the TX. One issue did arise whereby we had to remove the inductors that existed at the TX pad. Because the TX waveforms when passed thru an inductor showed a lot of ringing, it was hard to correctly set an a2d conversion point. By removing the inductors there was a much smoother waveform and we were able to add correct a2d thresholds. For our purposes at SoC level, we did not require to analysis the signal after inductors, this was completed by the IP team.
Below Figure 3 shows in impact of the inductors for our simulations. The top two signals show the ringing effect, while the bottom two show a smoother signal with the inductors removed.

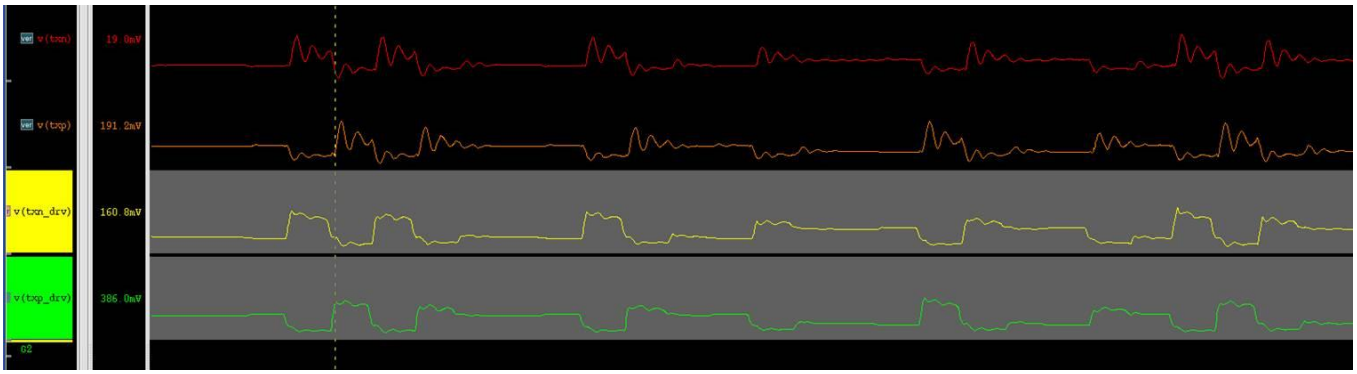

Figure 3: Inductor on TX path

Once we swapped the PLL DCO as BMOD, our PLL simulations also completed. Our test plan criteria such as PLL lock time, frequency measurements etc. were all with the specification limits. Again, because of the high quality of unit level MSV performed we saw limited gain in perusing the debug to get the DCO to oscillate as spice, and we deemed the risk to be low.

4.3 Other Analog-IP Results

Briefly, there were more Analog IP's that were validated at a Full-Chip level by members of the wider SoC MSV team. These IP's had greater dependency and interaction with the SoC then perhaps this HSIO example. They also had limited or no MSV performed at a unit level. Such blocks included a thermal sensor with integrated voltage regulator, and a PLL and clock distribution unit.

Some bugs were found in the thermal sensor relating to the power up sequence. The SoC team did not fully quantify the relationship between the regulator and thermal sensor, and the power-up sequences was not correct.

In relation to the PLL and clock divider, a major bug was found whereby the cml output signal of the PLL was connected directly to the clock divider circuit instead of the converted cml2logic signal. The mix-up occurred because the PLL DCO IP was used in a copy and re-use manner from another IP team. The integrators were not made aware of the significance of these pins exiting the PLL DCO and connected the wrong pins in error. Also, initially the SoC team used forces on these pins to create the DCO frequency, so it was not until late into the project and only once MSV was performed that the issues were seen. The below Figure 4, shows in red where error in connection that was made, and green indicates the correct connection.

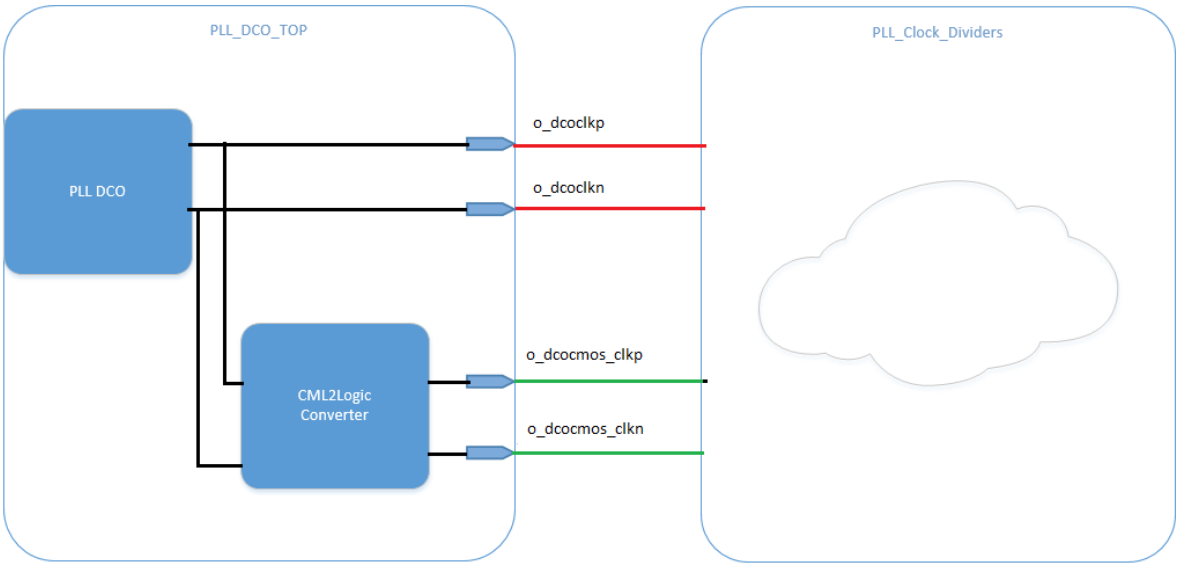

Figure 4: PLL Connection Error

5. Future Improvements

Due to some limitations, there were a number of features that we were unable to implement or build into the flow that we would hopefully include in future.

5.1 AMS checkers

Although some checkers exist in the AMS package, there may be a need to build your own checkers such as for slope/phase/delay measurements. Or build a wrapper script that enables existing checkers. We never fully implemented this. We relied mostly on manual waveform reviews and the overall pass/fail results of the digital test. Waveform reviews were slow and tedious at times due to the large size of waveforms dumped especially during initial debug.

5.2 Regression

We were not able to fully implement a regression flow. This would have been useful to have so the test could be re-run once a new version of the IP was integrated into the SoC environment.
However the ability to use a regression flow should coincide with a solid AMS checkers database so as to avoid the need to manually review waveforms.

5.3 Save/Restore

Due to the long nature of our runs, the save/restore function is an option worth exploring. For many of our AIP, the initial run time during the test was waiting for the SoC to come out of reset. In one example, the transactions of interest to us in the AIP, did not take place until 600us had elapsed. To reach this point may have taken 4 days of simulation time.
Our IP would only then start its initialisation sequence, traffic would pass through the IP and all relevant data for us was complete by 700us. This 100us may have taken another 4 days due to the nature of the traffic going through the device. So the ability to use the save/restore to get to the initial point of 600us could have been a benefit to us.

5.4 Interaction with IP team

As mentioned previously, as early as possible interaction with the IP team is crucial to avoid unnecessary delays in accruing the setup to enable the runs. The IP team typically were ahead of the SoC team in terms of setup and would have had the majority of the CustomSim configuration files in place. Their CustomSim setup files could be used as the initial starting point enabling simulations in SoC, and from there they can be modified as necessary.

5.5 More targeted tests

Our tests were a direct re-use of the digital SoC validation environment tests. In hindsight perhaps we also should have developed AMS specific tests, or perhaps ported the IP level AMS tests into FC environment. There were occasions, for example during PLL simulations, where the only test available to us was a full traffic sequence test. The issue here is that typically the PLL

achieved lock at approx. 250us into the simulation, but the full test would only complete once traffic was seen on the device at 700us. Therefore we typically killed the test shortly after PLL lock and once we saw all clock signals were valid.
It was not possible to let the simulation continue for full length due to the slow nature of PLL simulations, and the size of the waveform files that would be produced.

5.6 Should MSV be performed on an AIP at SoC level?

A detailed risk assessment should be performed on any Analog-IP block where consideration is given to perform MSV at a SoC level. Some of the criteria could include – What level of unit level MSV is performed, what does their test plan include/exclude, is the block primarily a standalone IP, or does it have much interaction with the SoC.

In this HSIO example, the IP team did have extensive unit level MSV simulations. The RX could potentially be regarded as standalone, as much of the control logic and analog components are self-contained within the IP. It could have been argued that the risk was low in not performing this SoC level of mixed signal validation on the HSIO. However, there was an extra level of satisfaction from the SoC team that is was performed and extra confidence that the Analog-IP was integrated correctly and functioning as desired.

Also, MSV at the SoC level was also the only scenario in the entire SoC environment whereby the full RX data path was exposed. In digital validation by default they bypass many of the BMODS in RX in order to simplify the data thru the RX.

5.7 Updates to AMS Flow

Below Figure 5, incorporates some of the main learnings as discussed above and highlights them as important points that should be followed to avoid delay in setup and to hopefully avoid unnecessary delays in running AMS at a SoC level.

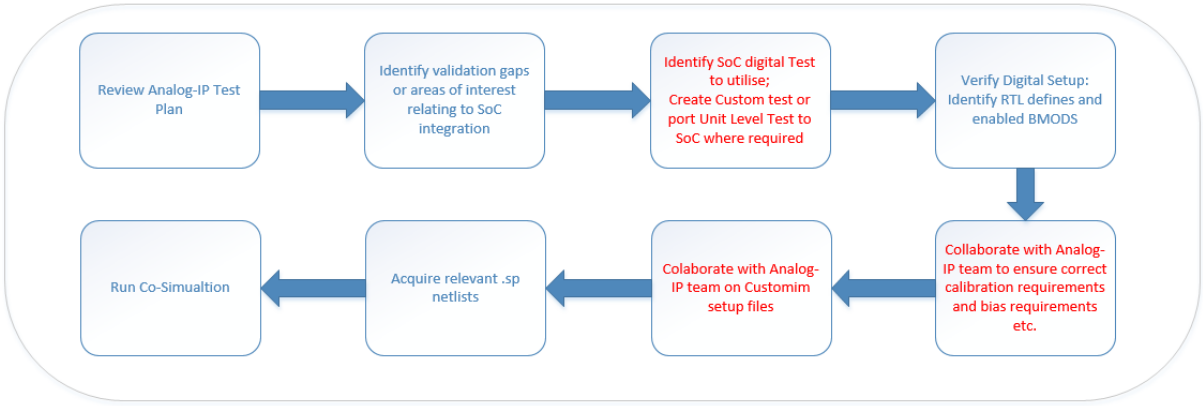

Figure 5: Updated AMS Flow

6. Conclusion

Overall, our experience with using MSV at SoC level was mixed. Although it was relatively easy to integrate into the existing SoC digital environment, there are certainly challenges in ensuring correct setup and test availability.

A good working relationship is crucial with the Analog-IP team in order to utilise as much of their setup as possible if it exists. It is important to ensure that unnecessary time is not spent is determining correct biases, calibration setup, etc., when typically the IP team would have at the very least, a solid starting point from where the SoC MSV team can then build onto if required.
Also, the possibility in utilising their unit level test at a SoC level rather than being solely dependent on using default SoC digital tests, especially in our example of validating the PLL, is worth exploring.

Although MSV is crucial when validating IP's like this HSIO at a unit level, a risk analysis should be performed on the need at performing this at a SoC level. Consultation should include the designers of the IP and the unit level MSV team. Gaps in the digital SoC validation plan should be analysed, and were necessary included into the AMS Validation plan.
In this HSIO example, the only simulation where the full RX path was exposed to the SoC design was in Full-Chip AMS validation.

For analog blocks with limited MSV at a unit level, a strong case can be made for performing MSV at a SoC level. As can been seen in the example of thermal sensor and PLL, errors in connection or incorrect start-up sequences could easily be made.

Finally, because of the fact that the CustomSim configuration files can easily be ported from IP level into the SoC, performing Mixed-Signal Validation at a SoC level, can be a worthwhile exercise and sanity check in ensuring correct integration of the Analog-IP into the SoC.

7. References

[1] Mixed-Signal Simulation User Guide, Synopsys
[2] CustomSim™ Command Reference, Synopsys

Appendix A: Debug Flow

1. Create a command file that defines the hierarchy of interest
    a. Example (vcd.cmd):
       dumpvars 1 top.parA.parA1.parA2.parA3.rxtop_ana

2. Convert golden vpd to vcd
    a. vpd2vcd inter.vpd -f vcd.cmd > tbench.vcd

3. Create testbench configuration file:
    a. The hierarchy name of the module instance
    b. 'testbench' for testbench generation or 'moduleGeneration' for module generation
    c. The module header and port declaration
       Example:
       top.parA.parA1.parA2.parA3.rxtop_ana
       testbench
       module test (a, b);
       input a;
       output b;

4. Generate testbench
    a. vcat tbench.vcd -vgen vgen.cfg

5. Compile and Elaborate
    a. Example:
       vlogan ./<testbench>.v
       vcs -lca -t ps –ad<block>.init" <tbench_module_name> -debug_pp

6. Simulate
    a. ./simv