Advanced Digital-Centric Mixed-Signal Methodology

Methodology for Analog and Mixed-Signal Model Integration and Progression inside a Mature Digital UVM Testbench

Michael Kontz (michael.kontz@hp.com, 970-898-0880**)** - Hewlett Packard Company
David Lacey (david.lacey@hp.com, 970-898-1786**)** - Hewlett Packard Company
Peter Maroni (peter.maroni@hp.com, 970-898-0508) - Hewlett Packard Company
Fort Collins, CO 80528

***Abstract*- Digital testbench methodologies have been at a mature state for a decade and continue to grow. Universal Verification Methodologies, pseudo-random stimulus, coverage driven verification, dynamic and comprehensive checkers, detailed verification planning, and Assertions Based Verification are the foundation for digital verification environments. However, to date, these environments have been primarily focused on functional verification of RTL. Using our next generation SerDes transceiver IP as an example, the authors will show how they expanded the capabilities of their mature digital verification environment to include analog and mixed-signal models and the ease and value this new approach brings to mixed-signal verification.**

I. INTRODUCTION

Hewlett Packard's Systems Design Lab (SDL) is part of HP's Enterprise Group and develops a portfolio of IP such as multi-gigabit transceivers which require custom digital and analog VLSI circuits for high frequency PLLs and DLLs, advanced transmitter and receiver frontends, and silicon photonic solutions. Additionally, SDL has implemented advanced verification methodologies through the years, utilizing the latest industry solutions as well as driving new methodologies. This investment in verification technologies has enabled the delivery of the advanced ASICs that drive HP's industry leading servers.

As system interconnect bandwidth demands increase and process technologies shrink, high-speed analog design is reaching the limitations of basic physics therefore requiring more digital feedback logic to tune, calibrate, and dynamically control circuits to ensure robust and high quality designs. For digital verification, advanced techniques such as coverage driven verification, pseudo-random stimulus, dynamic checking, formal model checking, 24x7 regressions, and increased automation continue to be refined and improved from project to project. With the dramatic increase in scale and complexity of mixed-signal solutions, these advanced verification methodologies are required to achieve schedule and quality goals.

The mixed-signal methodology presented describes how we are able to pull the mixed-signal models into the structure and power of our mature UVM digital verification environment. On the tools side, we have added a model swapping capability which allows the full suite of mixed-signal and analog models to be easily and seamlessly simulated in the mature digital testbench. Paired with this, we have developed a methodology and framework for our transceiver IP which enables low effort mixing and matching of various model types in the digital testbench simulation as they become available during project execution.

II. PREVIOUS APPROACHES

A. Custom Analog Mixed-Signal Verification Methodology

Historically for custom analog IP, our lab's mixed-signal verification has been exclusively owned by the custom analog designer. Their responsibilities included environment, small schematic testbench, and stimulus creation to exercise the design under test while ensuring physical robustness across process, voltage, temperature, and frequency. As typical with pure analog design, their validation efforts would focus heavily on ensuring margin at the worst case corners under the worst case stimulus (e.g. high frequency, low voltage, high jitter, etc.). And given a comfort level with analog solvers, such as SPICE, a majority of design was often left as low level, FET models with verification completed via manual observation of waveforms.

B. Functional Verification Methodology

Functional verification of the higher level link digital design in our lab has traditionally been owned by the logic and verification engineers. The testbench, components, and stimulus have been aligned with the UVM structure for several projects and more recently have used UVM-*e* and UVM-SV languages. The main drivers for the functional verification of the link design have been a digital feature centric test plan, constrained random regression, and coverage closure (assertion and functional). For previous generations of transceiver IP, a full discrete gate model from the analog design team was used. All modeling of analog components in this model has been strictly 4-state behavioral Verilog.

C. Why Change?

A methodology change or addition always requires a clear return on investment argument. Our argument for change was that for future link designs, the risk of a faulty design due to digital-analog interoperability had reached a tipping point with our current verification setup. The existing methodology was already starting to show weaknesses on the latest released design. Future designs promised to make things worse.

1. Problems in Previous Design

One of the link types in our last design had a much higher number of post-silicon issues than past projects. The post-mortem review of these issues revealed many of them could be blamed on insufficient testing of the digital and analog designs together. The main deficiencies identified were as follows.

Late availability of full functionality model for digital testbench. The first transceiver model used in the digital testbench typically came from a gate level export of the analog design team's final or near final schematic netlist. This dependency meant realistic full link training simulations, even at the pure digital level, could not begin until the analog design team had built up its entire transceiver design stack in schematics. For new technology nodes with a bottoms up composition methodology, this delayed integration and fine tuning of the digital testbench to work with the real transceiver design by several months.

Digital behavioral models not accurate enough. The transceiver model used in the digital UVM testbench never progressed past discrete behavioral Verilog for its analog components. This resulted in missed bugs in the upstream digital control logic because the behavioral representations lacked fully accurate modeling of sensitivities and timings to that of the real analog circuits.

Analog circuits and models only stimulated with worse case conditions. Broad coverage of all important operating conditions was sometimes lacking in the small mixed-signal testbenches. These analog testbenches used manual signal observation instead of automated pass/fail checking and did not have official coverage driven metrics that could have pointed out uncovered stimulus.

2. More Problems on the Horizon

The speed and complexity of the next generation transceiver designs require more digital logic to auto tune the analog circuits. This tuning also involves more complicated algorithms. For the next design cycle, it was clear to the team that better co-simulation of digital and analog designs had to happen.

III. NEW METHODOLOGY

A. Choosing a Direction

Like most methodology changes or additions, we decided to invest in a new mixed-signal verification approach at the start of our project design cycle. The next project involved new link and transceiver designs and a new process technology.

After investigating various improvements, we came up with two main areas of focus given the weaknesses identified last project. One was sharing more of the automation flows from the digital verification side with the analog/schematic based verification environment. This included items such as fully self-checking tests, automatic regression, and coverage. The second item was to improve the accuracy of the modeling in the flagship digital UVM testbench. The topic of this paper is the latter effort of bringing AMS to the digital UVM testbench we already have.

The direction of bringing mixed-signal into our digital UVM testbench had several key motivations behind it. First and foremost, we felt it would cover all the weakness areas identified above. Secondly, the vendor tool's mixed-signal support had reached a point where simulating various analog models and running the analog solver alongside the regular digital simulator was mature technology. Section V Model Mapping will show how this was only a few additional flags to our current simulator command. Thirdly, the incremental effort over our previous DV flow was

perceived to be relatively low. The digital block level verification already required the creation of a full featured digital testbench for the link training logic verification. Adding mixed-signal models into this testbench would be a way to take double advantage of all the verification methodology investment there already. An alternative approach would have been to invest more heavily on the schematic testbench side and add UVM type infrastructure as was done in [1]. Although some of this was done, investing fully in each of the numerous schematic testbenches looked like far more work than if we could fully bring the mixed-signal simulation into the single, larger digital testbench which included all of the mixed-signal blocks.

To make this direction a success, there were two main pieces of the methodology we had to create: 1) how to map mixed-signal models into the digital testbench and 2) how to manage the progression of these models over time.

B. Model Mapping Requirements

We desired a mechanism to interchange any possible model of equivalent functionality. This mechanism needed to be low effort and seamless such that the digital testbench stimulus and checking is not impacted by swapping in different model configurations. The interchange mechanism also needed to be flexible so models that did not have the exact same pinout could still be swapped. During project execution, the next iteration of schematic design might differ from the current behavioral model being used. The model mapping functionality needed to handle this non-ideal world without a lot of overhead. Section V Model Mapping covers our eventual solution.

C. Model Progression Requirements

Although we wanted some flexibility for swapping models on a feature granularity instead of an exact pinout granularity, we knew the closer in size and shape we made the different models the better. The reality working against us, though, was the bottoms up design of the transceiver (also called the PHY in this paper) by the circuit design team. The newness of the technology node and link requirements prevented them from being able to define a complete and rigid hierarchy and set of sub-blocks up front. One challenge with this was how to get an initial realistic model to start the higher level digital logic and link training verification. Another challenge was how to contain inevitable changes made in the transceiver design so they did not break the PHY model or its model mapping. To solve these challenges, the analog and digital teams came up with an evolution plan for the PHY model which allowed an initial model to be available on day one and minimized breaking the model swapping flow as the PHY model morphed into the final structural design of the real PHY. Section VI Model Progression section covers our PHY modelling solution.

IV. APPLICATION

A. Digital UVM Testbench

The starting place for our mixed-signal methodology is the block level digital UVM testbench for the high speed SerDes link. The testbench is built on UVM and contains either a pair of link halves or single link half in loopback. See Figure 1. The primary verification language for this testbench is UVM-*e* (Specman). The RTL side of the testbench is Verilog.

The DUT in the link testbench is comprised of a PLT (Physical Layer Training) and PHY (Physical Layer Transceiver) block. The PHY encompasses all the digital abstraction and control logic for the transceiver as well as the actual analog circuits and functionality. The PLT controls higher level transceiver algorithms including link training and auto equalization. This testbench contains the minimum set of logic needed to accomplish a complete link training sequence.

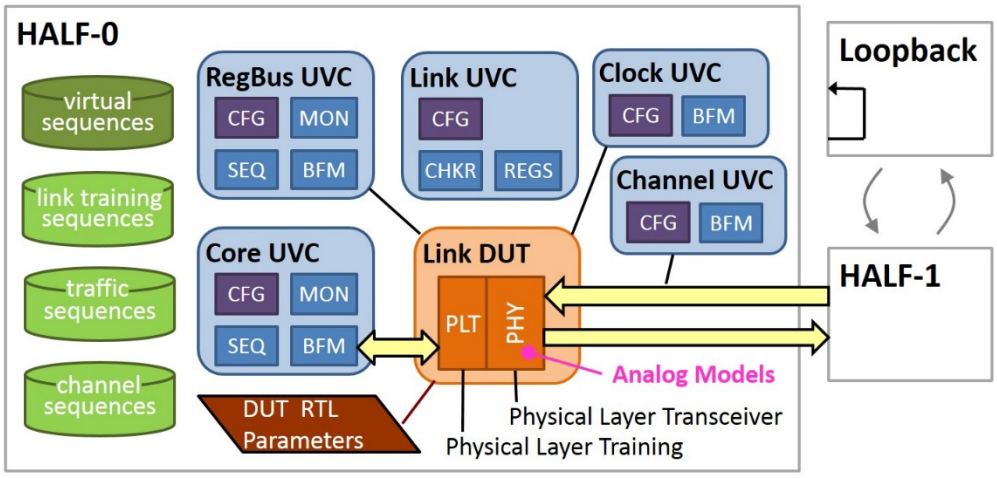

Figure 1: UVM Digital Link Testbench

B. Parameterized IP

For this project, the PLT and PHY design is heavily parameterized. This follows the industry trend of highly parameterized IP for the purpose of re-using in multiple products and over multiple generations. This heavy design parameterization had the advantage of allowing the DUT and testbench to shrink down to a single lane and provide the smallest possible complete design in which to swap models into.

C. PHY Model

The sole target of our mixed signal verification in the digital UVM topology is the schematic designs in the PHY block. Figure 2 shows the high level design of the PHY. The main sub-blocks chosen for AMS were the driver front end, receiver front end, and PLL (orange in figure 2). These blocks contained the most complex portions of the analog design while also requiring the most complicated interaction with digital algorithms. The TX serializer was also targeted for AMS testing because it was the first design available and was used for developing the model mapping support.

It is important to highlight that the PHY design is comprised of many layers of hierarchy in the schematics. This played well into our goal of being able to swap in models for different scopes of functionality. As an example, a model for the entire PLL could be swapped in. Or, we could swap in just a model for a specific sub schematic within the PLL like the VCO (voltage-controlled oscillator). Trying to swap models for one piece of a flat design is much harder than making an entire hierarchy layer (module definition) the target.

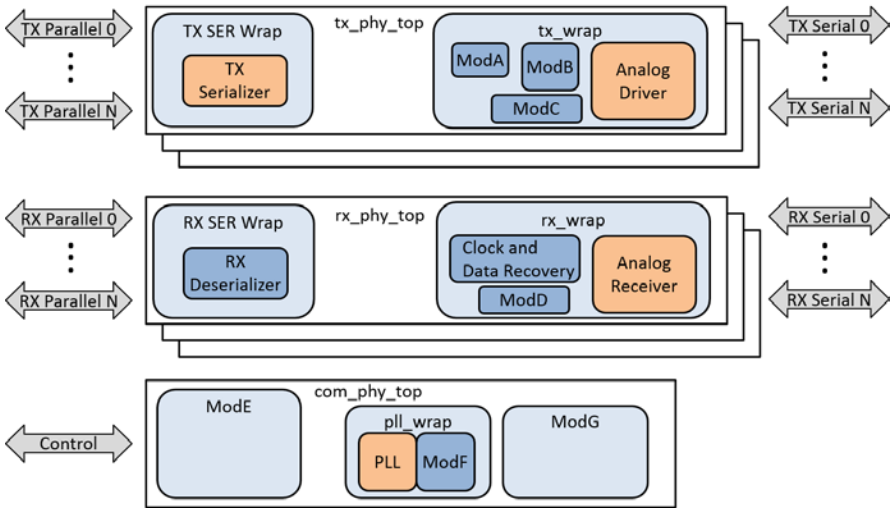

Figure 2: Top Level PHY Model

V. MODEL MAPPING

A. Model Swapping (Build flow details)

One of the key flexibility aspects of this methodology is the ability to easily substitute in alternate models at any point in the design hierarchy. This flexibility needs to include the ability to utilize any type of alternate model – four state discrete, Real Number Model (RNM), VerilogA/Verilog-AMS, schematics, or even RC extracted SPICE. Additionally, there was a requirement to allow the mixing of various model types in the same simulation.

The catalyst for this flexibility starts with a behavioral model skeleton that provides a Verilog design hierarchy of modules which mirrors that of the circuit design hierarchy. This structural design can be exported from the schematics with the analog design tools. With this design hierarchy in place, the substitution of models follows two possible paths. The paths used depends on whether the alternate model is a SPICE circuit. It is possible to substitute in different types of models into the same simulation, with some models being SPICE and others non-SPICE.

1. Verilog Config Block (not SPICE)

We found the easiest method to control the model substitution for models that are not SPICE is the *Configurations* feature of the Verilog language, which was introduced in the Verilog-2001 standard [2] [3]. Configurations allow the user to specify a mapping of source code to modules throughout the design. These mappings are captured in a library mapping file, often named `lib.map`. Many different configurations can be defined. The simulator will provide a mechanism to select which, if any, configuration definition is to be used. For the Cadence tools, this is accomplished with the `-libmap` and `-top` options to `irun`.

Utilizing configurations provides the user the ability to easily substitute alternate models for any module, specifying to replace either all modules of a given type or specific instances of a given module. Multiple configuration can be defined for use in different simulations. Configurations can be easily mapped to items in a verification plan.

```
# libmap file containing configuration definitions
config cfg_rx_amx;       # replace specific analog receiver instance with AMS model
  design worklib.top;
  instance link_top.\rxphy[0].rxwrap.analogrx use worklib.hp_analog_rx_ams:ams;
endconfig

config cfg_pll_rnm;      # replace all instances of pll with RNM
  design worklib.top;
  cell pll use worklib.hp_pll_rnm:rnm;
endconfig
```

Figure 3: Verilog Configurations

2. Analog Simulation Control File (amscf) (SPICE)

Verilog configurations cannot be used when integrating SPICE models. It can be used for all discrete models and all other analog models such as Verilog-A or Verilog-AMS. For the SPICE model cases, an alternate solution is used. While the specific syntax is different, the concepts are very similar. The approach described here details the solution provided by the Cadence tools and will need to be adapted to other vendors' solutions.

Analog simulation control files allow the user to specify a variety of control points related to the analog simulator. For our methodology, two features were utilized to enable the integration of analog models into the functional testbench. First, the schematic source files are loaded with `include` statements. This method was used for vendor model libraries, connect module definitions, and the custom analog circuit models. Second, the `amsd` block is used to specify the remapping instructions. Finally, the transient analysis command and options were specified to provide the analog solver instructions on how to operate.

```
simulator lang=spectre
include "cntl.scs"                      // connect modules
include "vendorLib.scs" section=top_tt  // vendor libraries
include "deserializer.scs"              // circuit definition
amsd{
  // match ports between schematic and module using port names
  portmap subckt=deser autobus=yes reffile="deser.v" porttype=name
  # must quote full path if path has generate loop instances
  config inst="link_top.rxphy[0].rxwrap.analogrx" use=spice
}
tran tran stop=3000n annotate=status max_minstep_nonconv=1000 max_approach_minstep=1000
```

Figure 4: Analog Control File

As stated earlier, one of the areas where the mixed-signal tools have matured is at the interface between the discrete and continuous domains. For the Cadence tools, connect modules are automatically inserted on all signals that cross between the digital and analog domains. The user can choose to use default connect module definitions or develop custom definitions which more closely model their specific translation of the signals between the two domains.

It is necessary to instruct the tools how to match ports between the different models. For Cadence, this is done using the `portmap` instruction inside the `amsd` block. There are a number of different options for this command, including a fully manual specification (used when port names are completely different between different models) and fully automated using port name matching. Refer to the vendor tool's documentation for details on all the options. The main challenge we faced on this front was the lack of search path support typically found in digital tools for finding reference files that may be specified in this command. This restricts where the file must be saved.

For the Cadence tools, it is necessary to specify analysis instructions for the analog simulator anytime the analog solver is used. These instructions are necessary even when using Verilog config blocks to map in analog models. For our methodology, we have utilized the transient analysis option. The work done by the analog solver must synchronize with the digital testbench simulator which means many of the more advanced analysis options such as Monte Carlo cannot be used. For the transient analysis, a duration must be specified. This tells the analog simulator how long to run. This value must be greater than the length of the simulation as driven by the digital testbench. This allows the digital testbench to be the master of the simulation and choose when to end the simulation. If the analog simulator ends the simulation, the digital testbench will normally report a failure that the simulation was ended prior to all the digital stimulus and checking being completed.

B. Debugging Model Swapping

One of the challenges faced when bringing all of these different flows and tools together was to determine if the right set of models was selected as a result of the Verilog configuration and the specified analog simulation control file. Each simulator should provide a mechanism for a verbose output of the model selection process. For the Cadence tools, using the `-libverbose` option provided detailed descriptions of what model was selected for each instance in the design. There is also an indication provided regarding why the resolution was chosen, such as the result of a Verilog configuration. A few examples are listed below from the Cadence help documentation.

```
Resolved design unit 'foo' at 'top.a' to 'source.foo:rtl' (`uselib at ./srce/top.v,3).
Resolved design unit 'foo' at 'top.b' to 'source.foo:behav'.
Resolved design unit 'bar' at 'top.c' to 'source.bar:behav'.
Resolved design unit 'adr' at 'top.a1' to 'aLib.adr:rtl' (using Verilog configuration).
Resolved design unit 'foo' at 'top.f1' to 'aLib.foo:rnm' (using Verilog configuration).
Resolved design unit 'foo' at 'top.f2' to 'aLib.foo:rtl' (using Verilog configuration).
```

Figure 5: Debugging Model Swapping

The user should take time to validate the build flow is operating as desired, even if the simulation builds and runs. The authors experienced an issue where an incorrect hierarchical path in the analog simulation control file resulted in no error or warning message. Since the hierarchical path was not correct, no model substitution was made as expected but no error or warning message was generated. This issue has been addressed in the tools since it was first found but illustrates the need for the user to validate that the desired models are being simulated. Appendix B contains additional output examples and debug commands.

C. Analog build flows

With our environment, one big enabler relates to our vendor tools' specific implementation of the build flow. In the Cadence build flow, there is a common build script (`irun`) which is shared between the command line interface used for both the functional and analog simulators. The use of a common build interface made the integration of analog models into the build flow of our functional simulation environment very seamless. Experience with other vendors may require more custom scripting to merge the build flows from each domains. Regardless of the process, this methodology should be viable with all the major vendors.

D. Analog Stimulus Additions to Digital UVM Environment

1. Supply Connections

One of the main differences between the port lists for a behavioral model versus a schematic model is the inclusion of supply ports (VDD/VSS). When swapping in an analog model which requires supply connections, our methodology currently uses Out of Module References (OOMRs) to connect to a module which defines the supplies.

```
module power();
   parameter real vdd_real = 1.8;
   analog begin
     V(top.invx1_1.VDD) <+ vdd_real;
     V(top.invx1_1.VSS) <+ 0;
   end
endmodule
```

Figure 6: Connecting Power and Ground

2. Controlling Temperature, Voltage, and Process

One of the powerful techniques used in digital simulations is pseudo-random stimulus combined with volume regressions. One area in the mixed-signal realm where this can be applied is to randomize select analog parameters. To date, we have looked at randomizing the values for supply, temperature, and process. We found that the mixed-signal tools are not as flexible in this area. The process must be selected through the AMS Control File, which is a build-time setting. Temperature and voltage are set through parameters, opening an opportunity to use the powerful randomization features of modern verification languages. However, the challenge we found is the mixed-signal tools read the settings for these parameters during file parsing, which occurs before any of the verification code is run. We have found a couple of solutions for this challenge.

If only one parameter is to be randomized in a single simulation, the AMS Control File allows the setting of a parameter in a parameter file. With the Cadence mixed-signal tools, this file is not read until simulation time zero, which allows an opportunity for the simulation to start up, invoke verification code to run prior to simulation time zero, which writes out a file with the correct format to set the randomized parameter value. Using this approach, the parameter value can either be set only for time 0 or varied throughout the simulation.

```
temperature.txt
    tscale 1e-6
    time   value
    0       89
    100    197

amscf.scs
    tran tran stop=125us param_file="temperature.txt" annotate=status
```

Figure 7: Controlling Single Analog Parameter

If multiple parameters need to be randomized in a single simulation, a `paramset` can be defined in the analog control file. Each parameter which will be randomized is specified in a separate column. The limitation with this approach is the parameter set must be defined in the AMS control file and must be read at file parse time.

```
pset1 paramset {       time  reltol  temp
                       0u    1e-3    27
                       10u   1e-3    50      }
tran tran stop=40us paramset=pset1 param_step=0 annotate=status
```

Figure 8: Controlling Multiple Analog Parameters

There is certainly opportunity for more flexible features in this area. The authors' ideas are captured in section *X Future Plans*.

E. Difficulties and Challenges Faced

While the major EDA vendor tools have supported mixed-signal simulations for a while, we found the maturity of these mixed-signal solutions was lacking in certain areas. The area which caused us the most difficulty was crossing Mixed-Signal simulation with advanced Verilog and System Verilog constructs. The areas where additional support was required from our tool vendor is documented in XIV Appendix A. Many of the issues described have since been resolved by our vendor or are slated to be resolved soon. This information is shared to prepare other users on the types of issues they may face when deploying the methodology we are describing.

VI. MODEL PROGRESSION

A. Progression

Planning out the creation and evolution of the PHY model in the digital UVM testbench was an essential step to our mixed-signal plan. To meet schedule demands for our new design, the digital logic team required a simulating PHY model to make progress until the analog team could finalize the design's microarchitecture. Investing in a solution with limited long term utility such as a temporary, inaccurate PHY emulator was highly discouraged. Our goal was then to create a PHY model framework that could provide a simple transition path from early behavioral models to analog and mixed-signal models as they became available. Figure 9: PHY Model Progression illustrates this transition path.

The first step in the model progression plan was for the analog design team to define a resilient, high level hierarchy for the link's PHY. Once this was established, the first PHY model was created using high level SystemVerilog constructs and was referred to as the 'generic PHY' since it did not include any technology specific models. The goal of the generic PHY was to enable the digital logic design and DV team to begin simulations with a PHY block in their testbench.

After a period time testing with the generic PHY, the analog design team, following a bottoms up methodology, would eventually progress to a point where analog or mixed-signal modules could be substituted into sub-blocks of the generic PHY. Though each generic PHY sub-block approximately matched its analog equivalent in functionality, port lists were bound to diverge preventing or complicating model swapping via Verilog configs. To facilitate selection between the generic behavioral model and hierarchical gate model, a technology specific wrapper was created for each major sub-block. Each wrapper incorporated a Verilog generate statement with an associated parameter to provide the conditional selection. This hybrid PHY model enabled simulating with a mixture of generic behavioral and/or real gate level models from the actual analog implementation. As the generic model often needed tie offs and adaptor logic, the wrapper provided a method to match the interfaces of the real gate model. As the analog design converged, incremental sub-blocks matured allowing substitution for their generic variants in the digital testbench. This allowed the digital team to integrate the any sub-block as soon as the analog designer had a new model ready.

With the creation of a hybrid PHY, analog and mixed-signal models could immediately be swapped into any instance in the gate level model. These methods, detailed in section V Model Mapping, worked seamlessly as the targeted sub-blocks for analog or mixed-signal model substitutions were equivalent to those of the gate level model generated from schematics.

The final step in the model progression is to instantiate a gate model of the entire PHY, including all levels of hierarchy that continue to allow model swapping similar to the hybrid PHY. In previous projects, the complete gate model was typically required to enable testing in the digital UVM testbench. By developing a model transition plan from generic to hybrid to real, our digital design team was able to progress in parallel with the analog design team, while incrementally assimilating more accurate sub-block models as they became available. This plan allowed creation of a single UVM testbench where all block interactions could be tested and reusing stimulus, checking, and coverage.

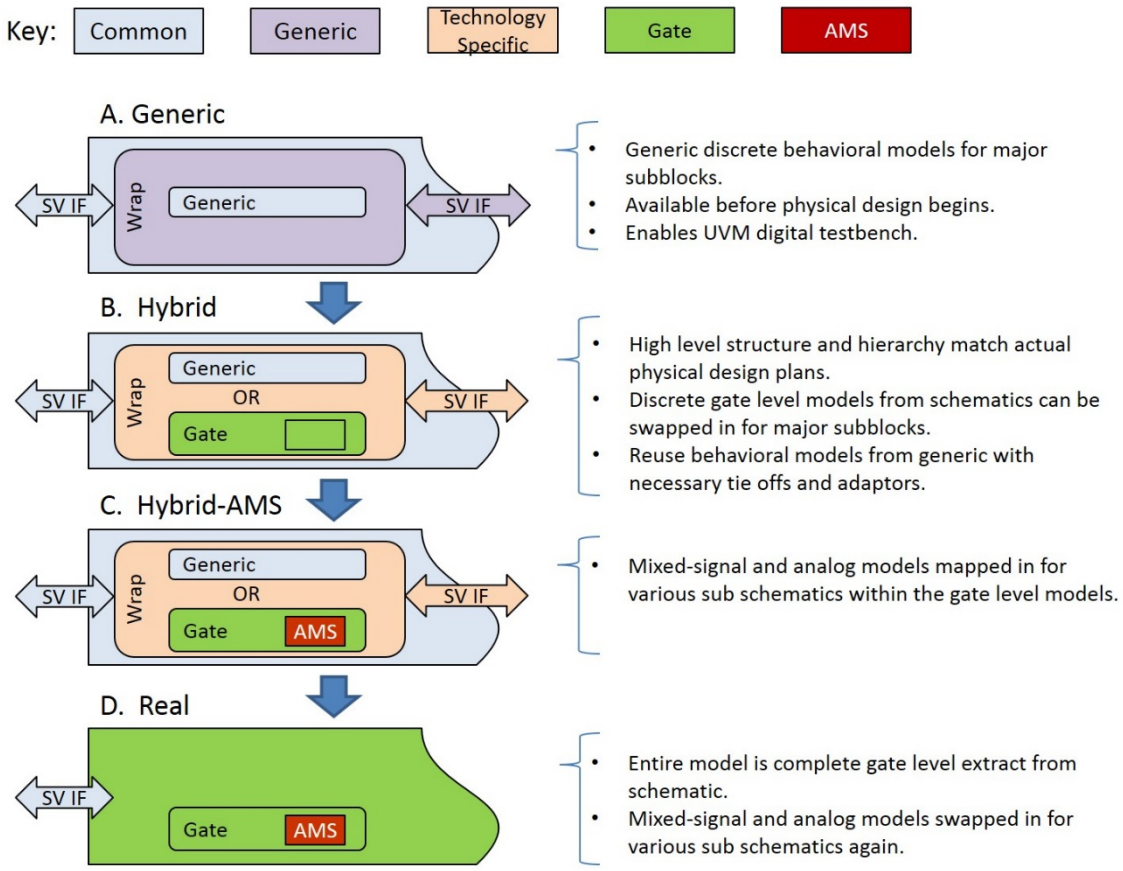

Figure 9: PHY Model Progression

B. Avoiding Thrash

Due to unknown design requirements with transitions to new process technologies, it is impossible to define a full module hierarchy and internal APIs for the PHY block up front. The internal sub-blocks and interfaces are bound to change as the bottoms up analog circuit design progressed. A key aspect of the PHY model was to make use of SystemVerilog interfaces to avoid explicit architectural signal declarations. These interfaces allowed the creation of a static PHY model that could more easily accommodate low level signal changes between major sub-blocks while also providing a framework to use behavioral models where only specific signals might be required (i.e. data vs. a bias current). For our implementation, these interfaces could be found at the top level, between the TX, RX, and COM slices, and between the digital and frontend sub-modules of the TX and RX slices.

VII. ROLES

A. Which Discipline Should Do What Work?

Bringing mixed-signal capability into the digital UVM environment required vendor support and a high level of teamwork between functional verification and analog engineers. Initially senior engineers from both teams worked to understand the capabilities of the different environments and then defined the high level vision for the mixed-signal simulation and PHY model progression in the UVM environment. Environment structure and behavioral model implementation was done by the verification team near the beginning of the project. Once the infrastructure and working behavioral models were in place, delivering and integrating gate and mixed-signal models was largely handed over to the analog team. This enabled the DV engineers to focus on their usual stimulus, checking, and coverage efforts leading up to design tape release while the analog team constantly improved the accuracy of the PHY model for each week's regression. During this process, it is imperative that engineers are able to work across disciplines to bridge the knowledge gap between the verification and analog engineers.

Work Item	Engineer
Define long term PHY model framework	Analog Microarchitect
Implement PHY model framework	Verification Engineer
Create generic behavioral models	Verification Engineer
Add model mapping capability to UVM env	Verification Engineer
Add analog solver capability to UVM env	Verification Engineer
Create hybrid modules	Verification Engineer
Create and validate equivalence of AMS models and analog schematics	Analog Designer
Integrate gate and AMS models	Analog Designer familiar with UVM env
First pass triage and debug	Digital Logic Designer

Figure 10: Division of Labor

B. Team Organization

One aspect of our lab that we believe enabled this methodology to be successful was the working relationship between engineers in different disciplines. In our organization, there is a clear distinction between the verification and analog design teams. However, while there is a structural division between these roles, there is also a significant amount of open communication and teamwork between the teams. Engineers from both disciplines working in the same physical building has helped this. Also, verification engineers were significantly involved on previous test chip efforts, building relationships between team members from both disciplines that are being utilized on our current project. This collaboration must be an intentional effort for it to be a benefit to the organization.

VIII. INITIAL RESULTS

A. Complete Flexibility Possible

The use of structural Verilog models generated from the analog designer's schematics quickly provides us a design hierarchy that matches the analog circuit. With this design hierarchy, the use of Verilog config blocks and Analog Control Files provides us maximum flexibility in swapping in the exact models placed at the exact instance locations to provide maximum simulation performance while achieving maximum verification coverage at the intersection of the functional RTL and custom circuits.

B. Progression Plan Worked

The PHY progression plan was successful in delivering a useful model earlier than previous projects which increased the verification time on the PLT+PHY design. One direct result of this was validation of the PLT algorithms such as basic link training and auto equalization were completed earlier. Also, this model has stood the test of time by enabling successful integration of completed schematics from the analog team without any refactoring of the PHY model framework.

C. Simulation Length

With our traditional functional link testbench, simulation runs were tens of thousands of nanoseconds compared with typical circuit designer simulations which ran for hundreds of nanoseconds. In anticipation that mixed-signal models would significantly slow down simulation performance, we looked for ways to shorten our link training testcase. The two main areas we were able to reduce significantly were the reset phase and the link training itself.

For the reset phase, it was simply a matter of tuning various testbench settings to get the link training started as quickly after simulation time zero as possible. The link training change made is more significant. Once the DV team better understood what stimulus and checking was interesting for most of the PHY circuits and their interactions with digital logic, a new testbench mode was added which emulated a common post-silicon testing procedure on transceivers known as BER testing. The testcase no longer had to simulate the link training all the way up to a L0 state. Instead, the PLT only brought the PHY into a test state and then with a very simple emulated BERT scope, ran test patterns looped through the transceiver (RX to TX). This much abbreviated testcase is not something the digital link testbench did before and so represents one significant change we did make to accommodate AMS models.

D. Simulation Performance

One of the biggest questions surrounding mixed-signal verification efforts is always where do the models become too slow for the simulation to be useful. The classic model accuracy versus performance chart (see [4]) shows the relative expectations but cannot be used to predict the breaking point for a particular design. With the model mapping infrastructure and hybrid PHY model in place, we are currently in the process of executing our mixed signal verification plan and collecting results. We anticipate that measured results for at least one interesting circuit will be available and its breaking point identified and ready to share at our DVCon 2015 presentation.

To maximize our analog solver performance, we used these options with the Cadence Spectre tool:

```
-spectre_args "++aps=liberal +cktpreset=sampled"
```

Figure 11: Analog Solver Performance Options

E. Licensing

One consideration that must be addressed with this methodology is licensing. A functional validation environment relies only on the RTL simulation license. When bringing in different types of AMS models, additional licenses are normally required. For the Cadence tools, we needed to pull analog and/or real number model licenses/tokens when running simulations with mixed-signal models. Check with you simulation vendor to determine the type and amount of licenses you may be need given the mixture of models you anticipate.

F. Impact on debug

Using the functional simulation environment to run AMS models required our team to look closely at how failures would be debugged. The traditional functional engineers are not familiar with the analog tools and the circuit designers are not familiar with the functional tools. Each team will address this challenge uniquely. For our team, the functional engineers took an AMS training course to grow their knowledge of the analog tools. This allows the functional team to continue to perform the initial triage on failures involving mixed-signal models. As previously mentioned, our organization has developed a close working relationship between logic, verification, and analog engineers. This allows us to use the same functional debug methodology with the failed mixed-signal simulations. A functional engineer will do initial triage on a failure to determine the likely source of the issue. If it is determined that the issue is within the mixed-signal model, the analog designers are pulled in to complete the debug to root cause.

G. Cross Discipline Understanding

The PHY model work has served as a great catalyst for dual environment understanding by the digital and analog engineers. Having some of the analog engineers know how to operate in the digital testbench environment allows them to drive model integration on their own later in the program. The DV team's new knowledge about the PHY design process and modeling has allowed them to expand the digital link testbench to include capabilities focused on verifying the PHY and not just the digital PLT link training. Overall, this collaboration has led to better appreciation from both sides of the verification methods each discipline uses. This most certainly will lead to continued collaboration and improved mixed-signal verification in the future.

H. Return on Investment

The premise of our approach was to take a simulation environment which was already invested in and fully developed and use it as the foundation for running mixed-signal models. This approach contrasts other approaches where the functional verification approaches such as UVM were taken to the analog tool environments. In our view, we are achieving similar results without having to duplicate the development of a UVM based simulation environment. Instead, we are reusing an existing environment.

After our initial investigation costs needed to understand this methodology, the cost of integrating the gate structure models and the different mixed-signal models is small using the techniques we describe. We estimate the additional engineering effort it took to bring the model swapping support online and put in the place the PHY model framework was around six engineering months. One analog and three DV engineers worked on this part time. There is an increased financial cost to purchasing additional licenses. However, overall, we have seen a very small investment cost. The return on this small investment is seen to be very large. There will be savings from reduced verification efforts by the circuit designers. The biggest return on our investment will be from finding additional bugs in the designs related to the interactions between the functional RTL and custom circuits. Even finding just a few bugs in this domain will multiple the small investment many times over.

IX. CONCLUSIONS

Bringing mixed-signal verification to an existing digital UVM testbench is possible and largely supported by today's EDA tools. By reversing the primary domain of the mixed-signal simulations from the analog world to the digital world, the power and capabilities of the digital verification methodologies are easily applied to analog circuit validation. This movement is needed to bring an adequate level of realism to the testing of digital logic which has responsibilities or sensitivities to analog circuits.

Moving mixed-signal verification to a large digital UVM testbench is not just a matter of tool support. The design and model architecture must help facilitate this direction and take into account the progression of design changes, especially when it is difficult to define all the initial specifications, as is typical when moving to a new technology node. The technique we found successful was to plan out between teams how the top level model would progress from abstract behavioral to final structural during the course of the project. Having some level of model available on day one of a project in the digital testbench was important. As project execution progressed, the model went through hybrid stages where certain layers of hierarchy underwent change to sync up with the newly completed schematic layer. Each hybrid stage enabled new types of models to be swapped into the digital testbench and reap the benefits of the more complete stimulus crosses, checking, and inclusion of higher level logic.

The analog design team benefits from a fuller set of stimulus and settings thrown at their block. The digital design team benefits from better models to catch bugs around controlling and reacting to the analog blocks. The project as a whole benefits from less bugs around digital and analog circuit inoperability.

X. FUTURE PLANS

We are currently at the stage where we have completed our investigation of the various options available using this methodology. As our current projects approach their tape release dates, we will have an increased understanding of the value we will realize. At the same time, we have additional ideas on future investment areas to broaden the capability of our new AMS methodology.

A. Optimize Simulation Time

We continually ask our vendors for increased performance from our simulations. With this new mixed-signal methodology, performance becomes an even bigger challenge so we will continue to look for ways to optimize performance. One optimization used in the functional domain is the capability of taking a snapshot of the simulation at a specific point in time (such as after reset and initialization is complete) and saving it off. Then this snapshot can be reloaded by other simulations to enable them to skip portions of the simulation that are of lesser interest. Bringing this capability to the mixed-signal world would be a key enabler to increased performance. This capability is available in the digital-only world and in the analog-only world but does not work in the mixed simulator world.

B. Analog Setting Randomization

One of the most valuable capabilities found in functional simulations is randomization. We see key opportunities to bring more randomization to the AMS environment. Specific, we have experimented with methods to randomize voltage, temperature, and process when running with AMS models. Utilizing randomization in these areas combined with the automation already in place for our regressions would allow us to cover a much broader space. The biggest challenge we have seen in this area is for the tools to allow the setting of these values at run-time versus compile-time. The desire would be to use the robust verification language with its full featured randomization capability to make the selections for these values and use an interface to tell the analog tools the value selections at simulation time zero.

C. Assertions and Coverage

Functional verification has long used assertions and coverage. Our desire is to bring these capabilities into the mixed-signal domain. There has been a decent amount of work in this area already and we anticipate utilizing work done by others in this area. For now, the existing digital domain coverage and assertions will be used to verify analog design interoperability and stimulus coverage.

XI. REFERENCES

[1] D. C. Brownell and C. Schmitt, "Digitizing Mixed Signal Verification: Digital Verification Techniques Applied to Mixed Signal and Analog Blocks and System Level Verification," in *Design and Verification Conference (DVCon)*, San Jose, 2014.

[2] "IEEE Standard for SystemVerilog -- Unified Hardware Design, Specification, and Verification Language," [Online]. Available: http://standards.ieee.org/getieee/1800/download/1800-2012.pdf.

[3] S. Sutherland, "What's new in Verilog 2001 HDLCon presentation," Sutherland HDL, Inc., [Online]. Available: http://www.sutherland-hdl.com/papers/2000-HDLCon-presentation_Verilog-2000.pdf.

[4] W. H. a. S. Cranston, "Real Valued Modeling for Mixed Signal," January 2009. [Online]. Available: https://www.cadence.com/rl/Resources/application_notes/real_number_appNote.pdf. [Accessed 30 January 2015].

[5] B. Bhattacharya, J. Decker, G. Hall, N. Heaton, Y. Kashai, N. Khan, Z. Kirshenbaum and E. Shneydor, Advanced Verification Topics, San Jose: Cadence Design Systems, Inc., 2011.

[6] J. Chen, M. Henrie, M. F. Mar and M. Nizic, Mixed-Signal Methodology Guide, San Jose: Cadence Design Systems, Inc., 2012.

XII. ABBREVIATIONS AND ACRONYMS

AMS models	Behavioral model requiring analog solver, such as Verilog-A.
amsd block	Cadence AMS construct used to control the analog solver.
API	Application Program Interface
Analog models	Actual implemented circuits such as SPICE.
BERT	Bit Error Rate Tester. Common instrument used in post-silicon to measure the signal integrity capabilities of a high speed SerDes design.
Digital Testbench	The RTL testbenches that digital engineers use.
DMS models	Behavioral model requiring only discrete solver, such as real number models.
Gate Model	Export of the analog team's schematic hierarchical design.
Link	Generic term for the entire SerDes transceiver design. Contains PLT and PHY.
Link Training	Link initialization into operational state where data can be transmitted.
Mixed-signal models	Any behavioral model, AMS or DMS.
PHY	Custom Physical layer. Encompasses all the analog circuits and digital abstraction and control logic for the custom blocks and sub-blocks of a link.
PLT	Physical Layer Training. Digital control for higher level link algorithms including training and auto equalization.
Schematic Testbench	The non-RTL testbenches that analog engineers use.
SerDes	Serializer/Deserializer. Generically refers to the interface block of a chip.
Transceiver	Lower Analog portion of the SerDes link.
Verilog config block	Verilog construct used to control which module definitions are used. Encapsulated by the keywords config and endconfig.

XIII. Acknowledgment

Many thanks to Cadence, particularly Harsha Hakkal, for their support last year in helping us understand the latest AMS capabilities and how to apply them in our environment.

Thanks to Dan Berkram in HP for pushing for this capability and arguing for its necessity in our designs going forward. Thanks to Laurel Abeyta in HP for her hard work and skill in coding up the majority of the PHY model framework and behavioral models.

XIV. Appendix A

A. Difficulties with Advanced Verilog Syntax

Over the last four to five years, we have seen a steadily increasing level of support for advanced Verilog language constructs in the digital simulators and the backend synthesizers. This has enabled our design teams to work at a higher level of abstraction as they develop the RTL models of our IP. Features such as SystemVerilog interfaces, generate statements, and advanced parameterization are now commonly used throughout our designs. While these features are fully supported in the digital simulators, they were not by the tools needed for building mixed-signal environments.

1. Arrayed SystemVerilog Interfaces

When working with the mixed-signal tools, we found that while they would support standard SystemVerilog interfaces, they would not support arrayed interfaces.

```
module top  rx_phy_if.phy_side rif);                 // this is supported
module top  rx_phy_if.phy_side rif[NUM_RX_LANES]);   // this is not supported
```

Figure 12: Arrayed SystemVerilog Interfaces

2. Unpacked Arrayed Parameters

The mixed-signal tools using the analog solver do not allow the use of unpacked arrayed parameters. They do support packed array parameters. Be careful when using these features as we saw results where parameters were ignored or the resulting design hierarchy was different than what was specified after the model swapping happened. Utilizing the `-libverbose` option was key to ensuring expected behavior.

```
parameter bit[3:0] RX_LANES[0:3] = '{2,0,0,0}'   // unpacked arrayed params not supported
parameter bit[3:0] [3:0]RX_LANES = '{2,0,0,0}'   // packaed arrayed params supported
```

Figure 13: Unpacked Array Parameters

3. dot-star Port Connections

SystemVerilog supports automatic port connections in cases where the port name and size are the same using the .* syntax. For the mixed-signal tools using the analog solver, we found that this syntax is only supported with basic Verilog datatypes. It cannot be used with complex datatypes, including enum, struct, union, interface, class, and interface class. This is a prime example showing the mixed-signal tools have not been stretched in the same ways the discrete tools have been. The workaround for this is to full specify the port mapping.

4. Positional Port Connections

Our logic design team will often use direct port to port connections without the need for explicit wire declarations. This is not supported with the mixed-signal tools in simulations that are using the analog solver.

```
module pll_wrap (input clk, input rst);
  pll_0 (clk, rst);
endmodule
```

Figure 14: Positional Port Connections

B. Inconsistent support across digital and analog tool sets

1. amsd Config Block: portmap reference file only supports Verilog (not SystemVerilog)

Many of our files now use SytemVerilog syntax. While the differences are subtle when it comes to module definitions, there are specific differences. The `portmap` command in the `amsd` config block does not allow to use of a reference file that uses the SystemVerilog syntax.

2. Using both the Verilog Config Block and AMSD Config Blocks in One Simulation

Our initial structures for mapping in alternate models often involved using both a Verilog Config Block and an amsd Config Block in the same simulation. The initial versions of the tool we used would not support this.

3. Poor error reporting

Early versions of the mixed-signals tools would not report any issue if we specified an incorrect instance path in the amsd config block.

4. Verilog Hierarchy Path with Generate Statements

The generate statement has been a part of the Verilog language since 2001. However, it was not consistently supported in digital and synthesis tools for many years after its introduction. When we started using our Verilog code with the mixed-signals tools, we found that different tools required a different syntax when referencing a design hierarchy path which included a generate statement. Cadence uses an array notation when specifying which instance resulting from a generate loop. When using Verilog config blocks, a portion of the path must be escaped. When using the amsd config block, the string must be quoted and an extra option specified (`-ams_generate`).

```
Config block format: instance top.rx_\phy[0].deser use worklib.deser_ams:sv;
amscf format: config inst="top.rx_phy[0].deser" use=spice
```

Figure 15: Verilog Hierarchy Path with Generate Statements

5. Analog Tools Do Not Have Full C-preprocessor Support

There are a number of preprocessor features that often get taken for granted such as including files, defining macros, and using conditional statements. The analog tools do not have the same level of support as the discrete tools in this area. The `-spectre_e` option is required to invoke preprocessing.

The analog tools allows an `include` statement in the AMS control file but the preprocessor only processes the top level files. If an included file uses any preprocessor syntax, it is not processed. In the example below, the preprocessor directives in the included file cause a syntax error.

```
amscf.scs
    `include cntl.scs
cntrl.scs
    #ifdef OPTION1
    #else
    #endif
```

Figure 16: Analog Control File Include Statements

6. portmap Reference File Does Not Support Search Path

The discrete compiler tools have a rich set of features to support a variety of file organization approaches. One of these features is the concept of a search path to find files. There are features in the analog tools which do not support this type of organization, requiring a fully specified path to files.

XV. APPENDIX B

A. Additional Model Mapping Debug Output Examples

The tools also provide additional debugging information with regard to where connect modules were placed. This provides an indication of where there is a transition from a discrete domain signal to a continuous domain signal.

```
net disciplines for: top (top)

outa.......wire (electrical)
outb.......wire (electrical)
outc.......wire (logic)

net disciplines for: top.a2 (ainv)

in.........input (electrical)
out........output (electrical)

net disciplines for: ring.outd__logic2elect__logic (logic2elect)

aVal.......output (electrical)
dVal.......input (logic)
```

The Cadence mixed-signal tools offer two options which provides additional details in the build log regarding the disciplines being resolved: `+dr_info` and `-iereport`. Both of these options provide information that can be used to ensure the desired set of models was correctly inserted.

```
---------IE report -------------
Automatically inserted instance: top.outd__logic2elect__logic (merged):
  connectmodule name: logic2elect,
  inserted across signal: outd
  and ports of discipline: logic
  Sensitivity infomation:
    No Sensitivity info
  Discipline of Port (dVal): logic, Digital port
  Drivers of port dVal:
    (top.d1) assign #10 out = ~in
  Loads of port dVal:
    Load: VST_S_BLOCKING_ASSIGNMENT, Line 60, Index 0,
     in: top.outd__logic2elect__logic
  Discipline of Port (aVal): electrical, Analog port

IE Report Summary:
  elect2logic ( electrical input; logic output; )        total: 1
  logic2elect ( logic input; electrical output; )        total: 1
  ------------------------------------------------------------------
  Total Number of Connect Modules                 total: 2
```

Figure 17: IE Report

```
  Discipline resolution Pass...
  Performing Bottom-up Traversal..
   top.d1.in has resolved discipline logic
   top.d1.out has resolved discipline logic
   top.a2.in has declared discipline electrical
   top.a2.out has declared discipline electrical
   top.d3.in has resolved discipline logic
```

Figure 18: DR Info Report