Plan & Metric Driven Mixed-Signal Verification for Medical Devices

Gregg Sarkinen
Medtronic, Inc.
8200 Coral Sea Street
Mounds View, MN 55112
(1) 763.526.1923
gregg.t.sarkinen@medtronic.com

ABSTRACT
This paper discusses the experience of using plan and metric-driven verification on a recent mixed-signal integrated-circuit (IC) prototype at Medtronic. The design consists of digital and analog circuits which traditionally have been verified with unique tools and methodologies to perform simulation tests. The approach for this IC, while relying on the unique capability of the tools for each design discipline, integrates the test environment for the digital and analog circuitry and adds a unique dimension to the testbench technology. The focus of this effort is to efficiently demonstrate functional confidence in the IC prior to fabrication.

The primary goal, verifying functionality, is achieved by first planning for verification. This consists of analyzing requirements on the analog and digital circuit functionality and understanding the design architecture and interfaces. In order to efficiently achieve full IC functionality within project means, a common test environment must be architected to accommodate the digital circuitry, analog circuitry, block connectivity, and full chip integration as a stand-alone IC. An additional outcome is verification component and behavioral model reuse at higher levels of integration.

Metric driven verification, an advanced testbench simulation capability originally developed for digital circuitry, has been adapted for analog designs as well. This methodology, along with integrated digital-analog simulation capability, is the technology for achieving the project objectives. Metric-driven verification is a methodology for verification by which functional coverage for the device-under-test (DUT) is defined and automatically assessed against the simulation runs. The coverage metrics may include structural measures such as code coverage in addition to functional coverage. Simulation completeness is defined when coverage goals are met while all simulations are passing. Stimulus for the DUT relies upon highly developed constrained-random generation, automating what used to be a tedious process of manual testing. Additionally, the metric-driven methodology is scalable to higher levels of integration because the stimulus and checking constructs have been separated in the testbench environment. Checkers are used that span across the Analog/Digital boundary, which has traditionally been a rich source of error. Historically, these checkers have been confined to the individual domains – analog and digital, making assumptions and approximations about the other domain

The final IC level standalone environment serves to verify digital and analog block functionality, and end-to-end chip functionality. The functional verification of the mixed signal IC prototype is approached as a complete component. Advances in digital simulation have been further enhanced with analog support to allow for feature level chip verification. The benefits are realized by using common testbench components and automated tests.

Categories and Subject Descriptors
A.0 **[General]:** Conference Proceedings.
B.4.4 **[Input/Output and Data Communications]:** Performance Analysis and Design Aids – *Simulation, Verification, Worst Case Analysis*.
B.5.2 **[Register-Transfer-Level Implementation]:** Design Aids – *Simulation, Verification.*
B.7.2 **[Integrated Circuits]:** Design Aids – *Simulation, Verification.*

General Terms
Algorithms, Measurement, Languages, Verification.

Keywords
Metric Driven, Coverage Driven, Constrained Random, Mixed Signal, High Level Verification, Advanced Verification, Specman, AMS.

1.0 INTRODUCTION
Implantable medical devices have strong requirements for correctness and reliability. Thus, the functional verification process and comprehensive planning and metrics are relied on to accurately gauge verification progress and thoroughness. Medical ICs are almost universally mixed-signal in nature since they must interface with the analog world of the human body.

Currently, many ICs are mixed signal in nature regardless of industry. The methodology described in this paper provides a means to systematically perform functional verification on mixed signal ICs through simulation.

2.0 VERIFICATION PROBLEM
This project included designing and verifying a prototype mixed-signal IC that was part of a larger system of components of which this IC was also a component of a system interface. The goal of this project was to verify the complete IC functionality which included: end to end requirements, digital and analog integration, and higher levels of integration with external components. This verification

effort had to satisfy IC verification goals as well as be reusable at the system level of integration. To achieve these goals, a verification process was adhered to which brought initial focus to planning. The planning phase was a critical point in the verification effort because the verification scope and goals were identified and a solution was proposed and communicated to the broader team.

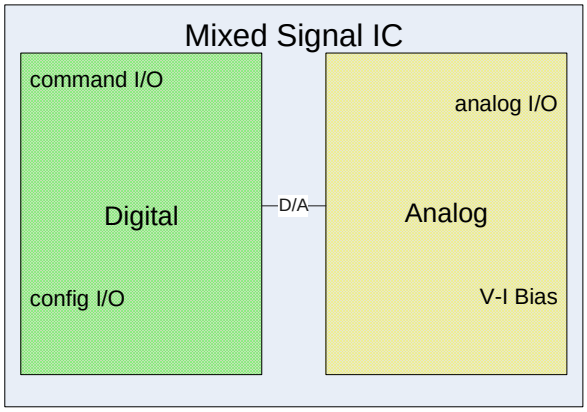

Figure 1 Mixed - Signal IC

3.0 VERIFICATION PLANNING

High level goals of achieving an environment, that was fully automatic and regressable as well as scalable for reuse at the system level, were stated from the outset.

This planning stepped back from individual analog or digital blocks and viewed the completed system. A hierarchical approach was taken where a leveling of verification was applied. For example, the IC verification environment did not need to achieve what was accomplished at block level verification. And likewise, the system verification did not need to repeat IC level verification.

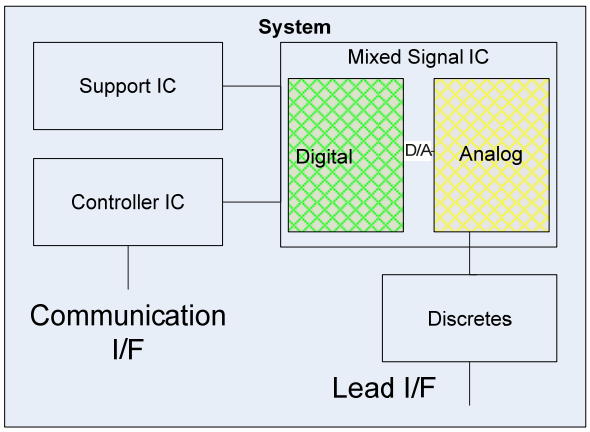

Figure 2 System with Mixed - Signal IC

The requirements to be verified for the IC were focused on the interfaces and end-to-end functionality. Internal interfaces were left to block level verification. The verification planning began with a survey of all the efforts, block-IC-system, being applied. Test coverage goals were defined for each interface, discipline, and functional requirements.

Challenges to overcome during this phase included: developing a complete verification plan prior to implementing and running simulations, attaining feasibility that self checking methods on analog circuitry could be achieved prior to committing to the plan, developing a resource plan to commit to this effort which involved additional development of verification components, and finally reporting progress against milestones which used coverage reports versus the traditional simulation test count executed.

4.0 VERIFICATION ENVIRONMENT

The verification applied to this mixed-signal IC followed a metric driven verification methodology approach. This paper doesn't define the commonly used 'metric driven verification' methodology but rather how it is applied to mixed signal designs.

Numerous complexities were addressed within this project: constrained random analog stimulus, analog self checking, analog coverage metrics, mixed electrical disciplines (SW, digital, analog), various analog and digital design representations, multiple simulation tools, multiple verification languages, behavioral modeling, and a high degree of automation.

One aspect of the IC under test that will be used as a case study of analog self checking is the output pulse. The mixed-signal IC produces an analog electrical waveform with the following attributes shown in Figure 3 below. The input to the mixed-signal IC is a digital command that triggers the output waveform.

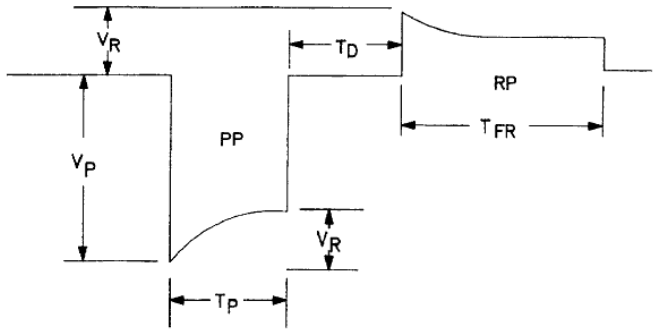

Figure 3: Pace Pulse Waveform [1]

4.1 Overview

The following figure illustrates a high level view of the verification environment. Given the black-box approach, only the external IC interfaces are of particular interest. The black-box approach is one attribute in attaining a scalable and reusable environment. An immediate payback in this effort was the ability to test various configurations of the 'device under test'. For a given level of verification, black-box enforces verification against interface requirements and end-to-end requirements versus the implementation itself.

The verification environment shown in Figure 4 is capable of providing stimulus on both the analog and digital interfaces as well as performing continuous monitoring of all interfaces in both the analog and digital domains.

In continuing the case study, the 'agent' performs the initial device configuration and generates high level test sequences. The sequences in digital form are played through the config and command 'e verification components (eVC) bus functional models (bfm). The eVC monitors collect all transactions, such as a pace command, and forward them to a predictor and scoreboard for checking. The analog I/F eVC monitors the lead interface and collects all pace pulse waveforms for checking in the scoreboard. Thus end-to-end checking is achieved. The agent is also responsible

for generating the analog configuration and stimulus needed for simulation.

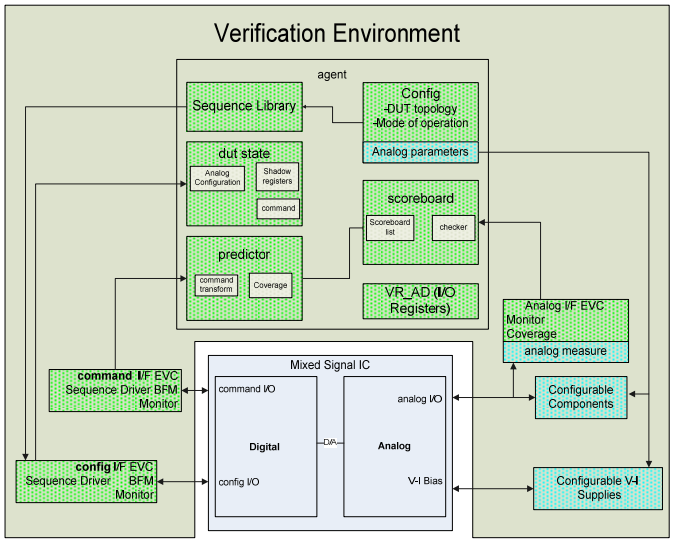

Figure 4 Testbench Architecture

4.2 Stimulus

All stimulus controllability for the verification environment was controlled via the language constructs provided by the hardware verification language. This includes both digital and analog parameters. Individual simulations were controlled via test files with no graphical user interface (GUI) interaction. The verification environment inherently varied all analog parameters with each new simulation run unless constrained by the test file. All stimuli generated represented valid conditions thus the term 'constrained random'. The digital interfaces and parameters were exercised according eVC best practices and not discussed here. The case study will focus here on achieving analog controllability as synchronized with the digital control.

Types of analog parameters presenting a need for controllability were: resistance values, capacitance values, behavior model types (worstcase, nominal, bestcase), voltage supplies, bias currents, initial conditions, DUT topology (AMS-Schematic-LPE,etc). With the capability to vary component values in a constrained random way, elements of worst case circuit analysis were achieved.

The following code samples demonstrate one pathway, a static configuration setting, for deriving analog values and passing them into the environment. Since some analog parameters are needed at compile time, a preprocessing step was executed to create a defparams file (explained further in section 5 'Integration').

```
struct mdt _voltage {
volt_current : real;
keep volt_current==1.116 or volt_current==1.2 or volt_current==1.204;
keep voltage_current_corner == MIN => volt_current == 1.116;
keep voltage_current_corner == NOM => volt_current == 1.2;
keep voltage_current_corner == MAX => volt_current == 1.204;
};
```

Figure 5 Data structure for varying a voltage supply

Figure 6 Shows the e code used to process a structure shown in Figure 5 to create the alter module shown in Figure 7.

```
for each in me.as_a(ACTIVE mdt_analog_parameters).voltages {
        msg_tmp = appendf("defparam analog_stim.%s = %e;",
            it.inst.as_a(mdt_voltage_instant_t),
            it.volt_current
            );
            lmsg_alter_vams.add(msg_tmp);
         msg_tmp = appendf("%s dc %e",it.inst.as_a(mdt_instant_t), it.volt_current);
         lmsg_param.add(msg_tmp);
    };
msg_tmp = str_join(lmsg_param,"\n");
    outf("%s",msg_tmp);
    files.write(alter_vams_file, msg_tmp);
```

Figure 6 Specman code to create alter file

```
module alter;
defparam analog_stim.volt_current = 1.204000e+00;
endmodule
```

Figure 7 Sample alter file

AMS code to drive stimulus was embedded within the hardware verification language (HVL) code for additional flexibility. This method enables the power of both languages to be used together.

```
AMS code within 'e'
        unit verifier {
            keep analog_agent_code() == {
                "real a_phase;"; // aux variable
            };
            keep analog_code() == {
                "a_phase = 2*`M_PI*120M*$abstime;";
                "V(top.vin) <+ 1*sin(a_phase);";
            };};
```

Figure 8 Embedded AMS code [2]

Areas to pay special attention to are introducing artificial effects into the simulation. For example a current source that turns on cannot turn on instantaneously (creating infinite currents and voltages). Mitigating this concern requires building BFM like structures in the AMS domain to provide realistic stimulus. Another example is dynamically changing an electrical component value which may lead to false results or convergence problems. To mitigate this concern, some of the parameters were selected to be static and only changed or set at compile time.

Modifying the device under test topology is covered in section 4.4.2.

4.3 Checking

The case study continues here with identifying pace waveforms automatically and reporting key attributes to the scoreboard for final checking.

The self checking aspect of the analog electrical interfaces was the most challenging task in this verification effort. Similarly to the stimulus section, the self checking of the digital interfaces is not described in this paper. The prospect of identifying a waveform in the analog domain with all the pertinent attributes without prior knowledge (eVC monitor principle, monitor wasn't triggered by stimulus to detect waveform) was indeed daunting. Other challenges included how to partition the monitor across the AMS and e language domains, e.g. which language constructs to use for each monitor task. To summarize, a problem was to be solved without an obvious solution.

One question that is asked and must be answered is: Can self checking of analog interfaces be achieved? This project answered that in the affirmative. Another option is to manually inspect signals with the naked eye using a waveform viewer. This project did make

use of manual visual inspection to build confidence in the monitors. Nevertheless the motivations for verification components with self checking capability are addressed here. Electrical signals with information in the frequency domain are difficult to analyze by visual inspection. Another challenge associated with visual inspection manifests when signal content is only visual under specific zoom (time domain resolution) settings. Once self checking monitors are in place and functioning, many more permutations of the simulation state space are realized. The self checking lends itself directly to having a regression suite whereby design changes can be resimulated quickly.

The first step in articulating a solution was to understand with clarity the requirements on the interface (there is potential to overlook this step if verification discipline is not adhered to). Secondly, analog interfaces inherently present themselves with noise (capacitive coupling). An understanding of what constitutes acceptable noise must be developed as well. Understanding of the noise characteristics is refined through actual running of simulations and dependent in part on the analog simulator configuration parameters.

The second step in deriving a solution was to form abstraction layers in describing the analog interface, Figure 9. This method enabled a modular solution with each layer not having to concern itself with details from the other layers.

Language	Signal Abstraction
e	Waveform Attributes: Pulse width, Amplitude, etc
e	Signal entities: slope, edge, peak, max, min
e, AMS	Raw Electrical Measurements: voltage,current,cross

Figure 9 Interface Layering

One point to note is the higher layers span more time and details. The hardware verification languages are equipped to process information and perform analysis at these levels.

The following diagram illustrates how the processing is modularized and decoupled from each layer.

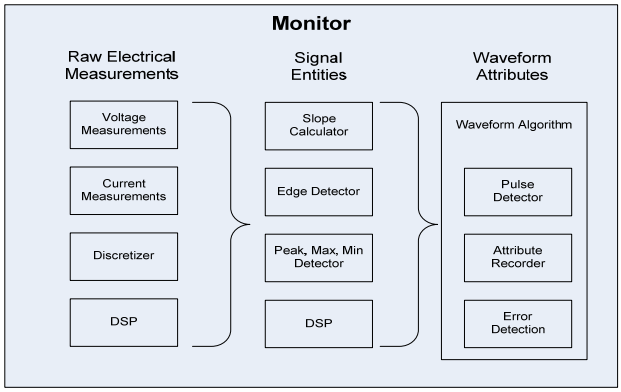

Figure 10 Modularized Monitor Structure

4.3.1 Raw Electrical Measurements

We now investigate the direct forms of observability used, particularly on the analog interface. Signals emanating from the DUT analog interfaces are of interest. There are multiple means to query electrical activity from the analog portion of the DUT. The first method shown below in Figure 11 utilizes the verification language constructs to read voltages directly.

```
Real Number Support
   keep port_x_voltage.hdl_path() == "ams_path.var_x";
   keep port_x_voltage.analog() == TRUE;
   sig_voltage : real;
   …
   sig_voltage = p_smp.port_x_voltage$;
```

Figure 11 Directly sample analog voltage

Another useful technique was the use of the AMS language built-in 'cross' function. This function triggers an event when a signal passes a programmed threshold. The verification environment can use this event to trigger a process or method as shown in Figure 12.

```
Events  (event_start is output from AMS cross function)
   keep port_event_start.hdl_expression() ==
         "ams_path.event_start";
   keep port_event_start.vams() == TRUE;
   …
   event start_e is @p_smp.port_event_start$;

   on start_e {…};
```

Figure 12 Cross Function

A third form with many derivatives is through the use of DSP (digital signal processing). Discretizers and filters placed in the AMS layer of the testbench monitor can reduce signal traffic between the simulator and verification environment. Nyquist rates apply when using the discretizer so correct sampling frequencies must be calculated. The filters serve to reduce noise sensitivity as simulators can be very precise. These methods are used in conjunction with the first method described in Figure 11.

To summarize the 'Raw Electrical Measurements' section, most language constructs used were in the AMS layer of the testbench as shown below.

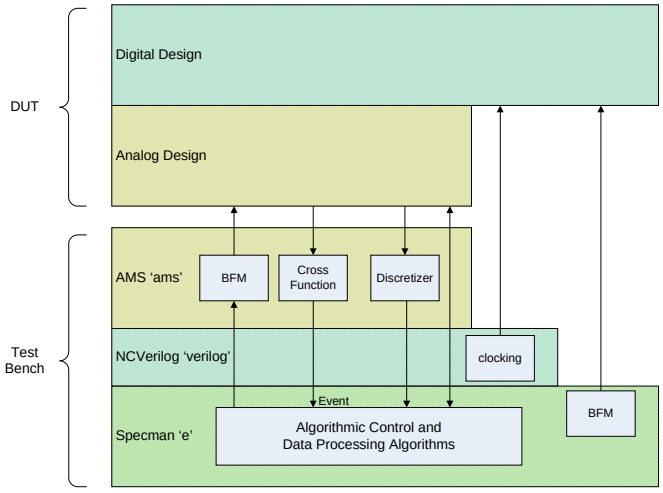

Figure 13 Discipline – Language Testbench Architecture

4.3.2 Signal Entities

By leveraging the list data type and list methods, many of the signal entity values are easily calculated. Many of the 'Raw Electrical

Measurements' were used in tandem to derive the calculated values. The vantage point the 'Signal Entity' level provides is the ability to view electrical signals from different perspectives and resolutions. For example, during critical measurement windows, a high resolution signal measurement (or low resolution signal) may be preferred to determine precisely the context of the signal.

The challenges offered in this domain center around understanding the noise floor and how to interpret signals that have been filtered.

4.3.3 Waveform Attributes

This layer of the monitor is comprised of multiple types of components. Finding a balance between real time processing and post processing can be shifting and dependent on the maturity of the algorithms used. Typical elements used here are state machines, post processing of lists, frequency analysis, power analysis, etc.

Typical continuous time functions are realized by processing lists of data in a discrete context.

This layer is responsible for ultimately creating the scoreboard data item and hence one challenge is defining precisely the scope and content of the scoreboard data item. For example, all the parameters in Figure 3 would be fields in the scoreboard data item.

4.3.4 Checking Summary

By carefully layering the monitor, the reuse potential increases by enabling the monitor via parameter settings to process simulation output from either simulated design capture or simulated behavioral models. In order to efficiently develop a monitor, one should make use of a debug environment where simulation run time is not dependent on the design. In other words, the debug environment is capable of executing one or two orders of magnitude faster.

4.4 Test Coverage

Test coverage is equal in importance as stimulus and checking in that it measures objectively what features have been tested and under what conditions for each specific DUT topology.

4.4.1 Coverage Reports

For this project, the coverage of the analog metrics was achieved by using discretized values. Rather than recording a continuous spectrum of values for a resistor value, three buckets were created for the minimum, nominal, and maximum values. For the amplitude of the pace pulse, a corresponding bucket was created for each digitally programmed value. Time values were rounded to the nearest pertinent unit.

Coverage definition, construct generation and closure had relatively few barriers in execution. Coverage was collected and monitored in the e language. Analog values were covered in the following manner. The same data structure that was used for the analog stimulus in the precompile step was also instanced in the verification environment during run time. During the precompile step an alter file with defparams is created along with a text equivalent file. At run time, the text equivalent file was read in, Figure 19, and the values were populated into the analog structure. Thus a seamless mechanism was used to pass all pertinent analog data into the verification environment. Shown below are a few sample snapshots of coverage reports.

The following figure shows coverage for a specific supply voltage.

Grade	voltage_current_corner	Runs	Hits	At Least
100%	MIN	613	613	1
100%	NOM	628	628	1
100%	MAX	595	595	1

Figure 14 Voltage coverage metrics

The figure below shows which corners were executed for a specific Spectre model.

	Grade
).capacitance_corner	100%
).leakage_corner	100%
).resistance_corner	100%
).inductance_corner	100%

Figure 15 Spectre model corner coverage

The next coverage report shows amplitudes that were observed on the output. For the amplitude report, values were normalized and do not reflect absolute values.

Grade	normalize_pulse_amplitude1	Runs	Hits	At Least
100%	[0x00001]	65	130	1
100%	[0x00002]	46	87	1
100%	[0x00003]	60	117	1
100%	[0x00004]	64	132	1
100%	[0x00005]	54	121	1
100%	[0x00006]	45	97	1
100%	[0x00007]	43	73	1
100%	[0x00008]	53	115	1
100%	[0x00009]	51	98	1
100%	[0x0000a]	71	148	1
100%	[0x0000b]	58	103	1
100%	[0x0000c]	54	112	1
100%	[0x0000d]	56	97	1
100%	[0x0000e]	47	83	1
100%	[0x0000f]	53	110	1

Figure 16 Amplitude coverage report

As can be observed from the three reports above, definition of the naming, buckets, and values is defined by the user and project.

4.4.2 Device Under Test Topology

The device under test topology had multiple realizations comprised of verilog RTL – schematic design capture, behavioral models, Spectre models, Gate level netlists, and LPE annotation. Results of the verification planning selected a specific set of these combinations for realizing test coverage. Which specific model representation to use, was a functional of the requirements under consideration. For example, behavioral models of the analog circuitry provided for more extensive coverage of the digital interface requirements.

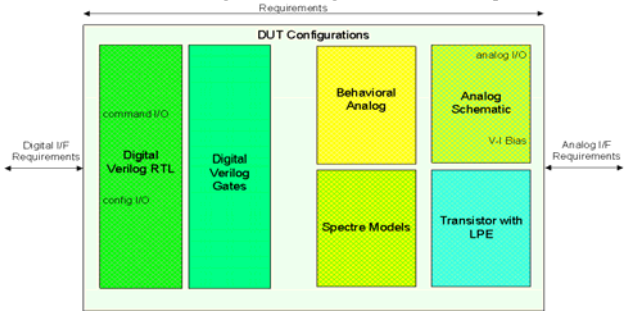

Figure 17 Device Under Test Configurations

A total of four different topologies were chosen for test coverage. As already mentioned, the verification planning strategically selected which topology was to be used to fulfill test coverage for a set of requirements. Behavioral model validation is not addressed in this paper.

In achieving automation, the netlist for each topology was automatically generated and treated as a separate configuration. Hierarchical configuration files for the testbench and the device under test were utilized to achieve maximum reuse and seamless transition of device topology into a common testbench environment. Unique Specman configuration files enabled accurate handling of constraints to all agents and eVCs within the environment as shown in the figure below.

```
// Instantiate the env
extend sys {
    pacing_env: ACTIVE'active_passive mdt_pacing_env is instance;
        keep pacing_env.has_ams_models == TRUE;
}
```

Figure 18 DUT topology constraints

Extremely useful was the generation of coverage reports based on DUT topology. This provided a mechanism to review exactly what test coverage was achieved for each DUT topology. The use of 'Scopes' within the executable verification plan enabled this automation.

5.0 INTEGRATION

This section focuses on how to keep all the languages, disciplines and tools synchronized to achieve a seamless compilation, elaboration, and simulation. The goal was to achieve a single point of controllability and observability, e.g. the 'e' verification environment, and thereby ease the processing of input test constraints as well as the generation of coverage data. Because certain analog parameters are required prior to the Specman test phase, a preprocessing step was performed which preserved the seed for the subsequent simulation.

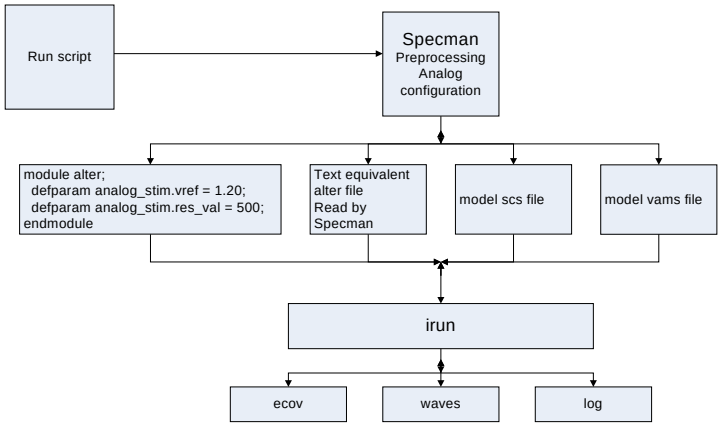

Figure 19 Discipline Integration

6.0 RESULTS

By using Emanager to launch parallel jobs 24x7, over 3000 mixed-signal simulations were run with pass/fail results. The coverage reporting mechanism within Emanager enabled thorough coverage analysis and closure.

Integrating the tools for simulating all languages e-verilog-AMS-Spectre simultaneously was accomplished via the 'irun' command from the Cadence Incisive Enterprise Simulator (IES).

Emanager vsif files enabled batch regression of all DUT topologies against a selected set of test files.

Many challenges have been noted throughout this paper. Additional items of interest are convergence and analog simulator settings. Convergence issues are inherent in the analog simulator solver (new concept for digital simulations) if initial conditions or artificial effects are introduced into the simulation. The analog simulators have many parameters controlling the simulation execution which can directly affect the result of the simulation.

Moving the analog/mixed-signal simulations to a high degree of automation required controlling stimulus from the HVL and not relying on a GUI.

Clocks crossing into analog domain and back into the verilog RTL must remain logical otherwise a transition fro 0 > 1 will result in 0 > X > 1 which may trigger two posedge clock events.

A playback method was created to save DUT analog output waveforms from a previous simulation and replay them to the verification environment in subsequent runs. This method enabled expeditious development of the verification components without having to endure longer run time overhead of simulating the 'device under test'.

Topology reports from within the simulator are especially useful in determining current sources and sinks in the cases where there appears to be 'extra' or 'missing' current on a node.

7.0 VERIFICATION REUSE

By following the eRM methodology and imposing verification discipline, the verification environment created for the DUT in Figure 1, was completely reused in the system shown in Figure 2. The verification environment for Figure 2 utilized the system CPU to run software test sequences exercising the 'Mixed-Signal IC.

8.0 CONCLUSION

Advanced verification methods commonly applied on digital designs are capable of being applied in the mixed signal and analog domain. The mixed signal and analog verification space stand to benefit by applying self checking constrained random techniques. Having a robust verification environment that bridges analog and digital brings the two disciplines together earlier in the design process. Finally, the verification effort must bridge domains and not be limited to design disciplines.

9.0 REFERENCES

1. US Patent 5,782,880 Low Energy Pacing Pulse Waveform
2. Cadence IES Help documentation