2014 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION INDIA
Sept 25-26, 2014
Hotel Park Plaza
Bangalore, India

Automated correct-by-construct methodology for RTL design and analog mixed-signal test bench generation

Enables early design closure of mixed-signal SoC

Lakshmanan Balasubramanian, Murugesh Prashanth Subramaniam, Atul Ramakant Lele, Embedded Processing, Texas Instruments (India) Pvt. Ltd., Bangalore, India
(*lakshmanan,murugesh,atullele@ti.com*)

Ranjit Kumar Dash, High Performance Analog, Texas Instruments (India) Pvt. Ltd., Bangalore, India
(*ranjitdash@ti.com*)

***Abstract*—Comprehensive, timely verification of complex, analog functionality rich sub-system, system, IPs require manual simulation setup that is error prone and iterative. It also involves an inefficient communication channel through manual review of specifications, intent and implementation between different teams of complementary competencies. Further there are several error prone, duplicated manual efforts involved including test bench for circuit simulation and behavioural model validation. These bottlenecks and limitations also extend to several aspects of digital RTL implementation / coding. To overcome these bottlenecks we have devised a methodology by which we propose to improve the communication efficiency, enable efficient digital design methods, and enable early analog functional simulations through automated generation of correct-by-construct analog simulation test benches, generation of RTL codes and assertions. We have implemented and demonstrated an efficient and intuitive infrastructure for such automation and for analog simulation output waveform analysis.**

Keywords—AMS co-simulation; functional verification; simulation; Integration specification document; testbench generation

I. INTRODUCTION

Typical mixed signal system on chip (SoC) design [1][2][3] involves several complex manual efforts including compiling the system specifications as a document, register transfer level (RTL) design, system integration, test bench and test case creation. This further involves transistor level custom design for analog functionalities and manual waveform analysis of analog simulations. For complex mixed-signal SoC design this poses a big challenge to meet time to market criteria. All these manual processes are error prone and time consuming. Even the high level synthesis involves manual coding the golden source models in C or System-C like high level languages.

Transistor level implementation of mixed-signal modules usually takes longer time to mature. The limitations due to different times of maturity of analog and digital sections of the design are usually overcome with use of analog behavioural models (ABMOD) and mixed-signal simulations with various models of different abstraction levels. Usually the ABMODs are made available to SoC integration before reasonable circuit maturity to enable early system level simulations. But all these involve parallel efforts in validating the ABMODs including the duplicate test bench creation.

The digital interfaces to analog modules are taken care by capturing the stimuli in Value Change Dump (VCD) format from targeted SoC level simulations and using them for analog module level simulations. The SoC level integration usually happens very late in the design cycle. Independent and rigorous verification of complex test cases of the system prior to the availability of VCDs, with an accurate digital control environment is usually not possible or delayed. Further analog mixed signal (AMS) co-simulation at SoC level is only possible at a very mature stage of the system integration. This poses a major challenge to IP maturity and sign-off in the SoC context, especially when there is a huge portfolio of complex analog functionality rich IPs (sub-systems) being reused with little or no modifications.

The above discussed process of defining the integration specification, setting up the test bench based on system specification and reviewing involves a lot of communication bottleneck between different teams of complementary competencies like analog, digital & system level. It was found that in the past project executions, these bottlenecks cause late identification of critical bugs which cause costly iterations and delay in time to market.

To overcome the above stated bottlenecks, in this paper we describe a methodology that we developed for automated generation of a portion of RTL design and an automated test bench generation for analog IP verification using an executable specification as a starting point.

A test case of a power management system in a mixed signal SoC including the analog power up is considered. The automated analog test bench generation resulted in a significant cycle time reduction due to saved iterations and enabled analog IP level SPICE simulations almost 3 months in advance compared to earlier executions of similar complexity with usual concept to tape-out cycle time of about 12 months. The automated digital RTL generation for a specific sub-set of system functionalities is implemented and applied. This paper will discuss the implementation details, advantages and limitations of the current implementation, key results and the directions scope for improvements.

The rest of the paper is organised as follows: Section II describes the integration specification documentation (ISD) and the limitations of existing simulation based verification methods. Section III presents our proposed method of specification driven automated generation of certain designs and analog simulation test benches. Section IV identifies scope for some future extensions to make the proposed methods more efficient and have a wider adaptability. The concluding remarks and summary is provided in Section V.

II. Integration Specification and Simulation Based Verification

Integration specification documentation (ISD) is the usual means by which functional modules' integration requirements or needs are compiled and communicated across electrical, physical implementation and to the system (SoC) integration functions. The requirements are the necessary boundary conditions to be met for a proper integration including a) any input signal timing requirements; b) load boundary conditions; c) physical design limitations like the bounds on the acceptable parasitic elements; d) signal integrity influencers like the bounds on signal and noise coupling; and e) reliability requirements.

Many of these conditions have targeted tools / flows to ensure their compliance especially for reliability, signal integrity, physical design and power management. The compliance of most functional and electrical constraints including power-up scenarios, many mode transition (like different performance and power modes) are usually verified only through functional simulations. A typical functional verification flow is illustrated in Figure 1.

Many boundary conditions are ensured in design usually by ensuring the complete and comprehensive functional simulations are performed with in the stated / specified range. When multiple inputs and load conditions set the specific boundary the comprehensiveness may be difficult to ensure and there can be coverage gaps. Further, it has been an observation of the authors over a long period that the specifications are not complete for lack of complete understanding of the various system level intricacies, and given the manual effort in tracking the verification plan against the specifications the simulations are not comprehensive.

Usually based on the manual process of reading, consulting and understanding the specifications provided, the different verification scenarios are planned, corresponding test conditions are arrived at, and test benches are built manually. The test benches are then reviewed by representative owners of complementary competencies and stake-holders like the applications, systems, system integration, physical design, DFT & test, reliability, and electrical/functional design for their comprehensiveness and correctness. Further the feedback from these reviews may result in iterations of this process until the test benches and scenarios signed-off to be mature. Thus the relevant communication bottleneck and complexity between different teams of complementary competencies (analog, digital, system & test) that matures at a much later stage in product development cycle involved in manual reviews that play a major role in sign-off.

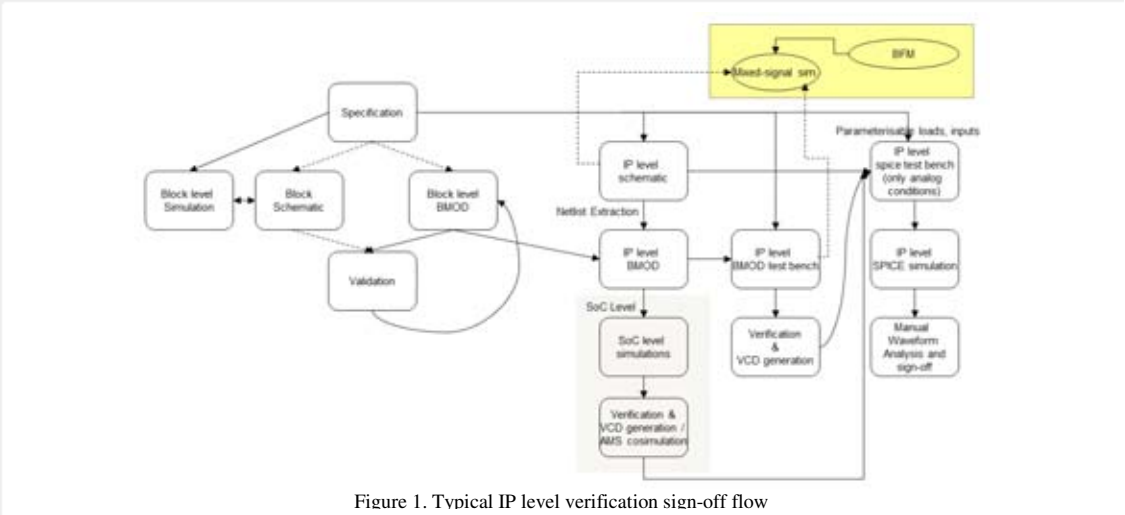

Figure 1. Typical IP level verification sign-off flow

At early stages of design (especially the transistor level design implementation) maturity, the functional verification closure to ensure specification compliance, architecture validity, and system integration correctness are enabled by fast simulation models i.e., ABMOD. While the ABMODs don't have to cover all the performance metrics (bandwidth, noise figure, power supply rejection, common mode rejection, harmonic distortion, linearity etc. to name a few critical ones, though this list is not comprehensive) they should comprehend all functional behavior between all possible input / output terminal combinations including all functional and testability modes of operation. Since only a sub-set of all such combinations are valid scenarios of interest, it is mandatory to identify those invalid modes and at least provide simulation time assertions upon exercise of such modes. This is especially important to highlight and avoid any system integration issue early, as not all such scenarios may show up as recognizable functional failure signatures during simulations either due to their inherent nature or due to lack of comprehensiveness of the verification suite.

One such critical boundary condition is the default state of several input control signals of the analog functional modules upon power-up of the system. Any violation of these may result in irrecoverable system failure, and will be hard to debug. The aforementioned assertions come in handy to save long debug effort and they aid localize the problem quickly and easily.

Further the validation of ABMODs against the specification at early stages of design maturity and implementation at mature stages is another important requirement, since most verification coverage at system or SoC level is obtained with RTL or gate level (GL) simulations using the ABMODs in event driven simulators. Since the AMS co-simulation coverage is limited to critically identified test conditions targeting system level analog behaviours and integration aspects, its coverage to overall verification coverage is limited. The test benches for ABMOD validation are usually manually developed independent of the electrical test benches used for IP sign-off simulations. This further duplicates the effort towards test bench creations and hence may cause errors.

Thus a sub-system level verification and model validation is usually not comprehensive until significant maturity in the SoC level integration, verification, and AMS co-simulation each contributing incrementally.

Even though AMS co-simulation flows have matured and are the right medium for complex system level verification, usually SoC level AMS co-simulation is gated by significant maturity of the IP design and system integration. Further SoC level functional verification is not the right medium for IP maturity sign-off. For complex analog mixed signal (AMS) IPs enabling AMS co-simulation may not be easy in the absence of right bus function models (BFM) that comprehend and emulate the system level scenario. Finally the most analog simulation sign-off happens through a slow, iterative, and error prone process of manual waveform analysis / review. The specific complexity and bottleneck of the verification we focus on and the typical relative design maturity time line are illustrated in Figure 2.

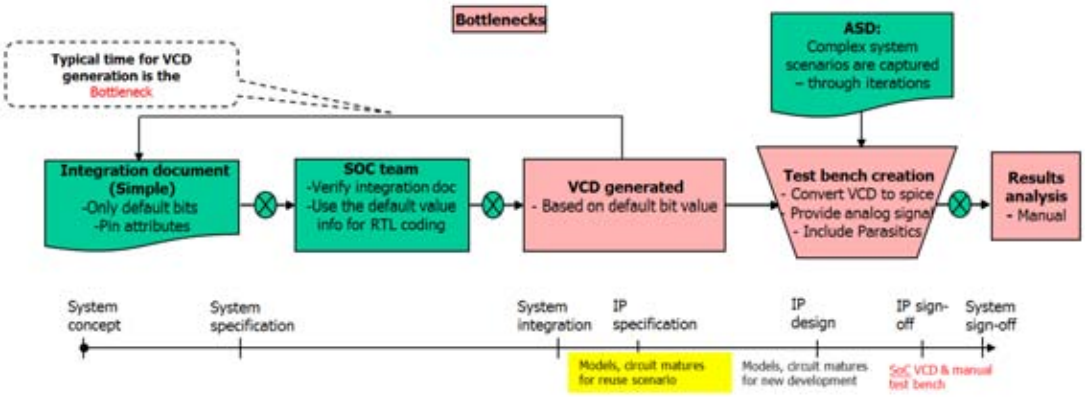

Figure 2. Key bottlenecks of analog IP anbd AMS SoC verification & Typical relative design maturity time line

III. SPECIFICATION DRIVEN AUTOMATION OF DESIGN & TEST BENCH

A. Automated test bench generation flow

To overcome the above stated bottlenecks, we conceptualised a methodology illustrated in Figure 3, with the following key components:

1. A product data sheet in XML format, XML being the widely used format for data sheet compilation & a main source from which various other format including PDF data sheets are generated for publication [4].
2. An extended integration document in a machine readable format (spread sheet) which is derived from the XML data sheet above, with additional fields that are custom to each type of IP to capture information needed for design, integration and verification which are not be available in the master data sheet. In the absence of XML product data sheet this may serve as the sole specification document.
3. An automation engine (perl, TCL and/or Cadence Skill language based) to read, interpret and generate portions of the RTL design, test benches that are correct by construct.
4. A black box simulation of the above infrastructure enables quick and automatic generation of waveforms for documentation. These can be embedded back into the original XML or spread sheet documents for ease of review. The integration document & the supporting documentation generated as stated above form an improved infrastructure for efficient and timely communication between different teams of complementary competencies.
5. This infrastructure also supports other existing input methods like VCD file based stimuli, Spice stimuli. The PVT corner information are provided as separate inputs.
6. An automated engine to post process simulation waveform results and generate HTML reports with goal checking. Enables assertion based verification with a sub-set of assertions generated automatically from the specification.

This methodology lends itself to easy extension for other features including the below listed:

1. Automatic generation of models of various abstraction levels like the functional, timing, power, electrical and physical information in various formats needed for SoC integration.
2. Extension to enable a push button system level analog simulation and verification framework.
3. The existing advanced digital system verification methodologies (like UVM, System Verilog & Specman™) to automate generation of test cases may complement our methodology to achieve greater automation.

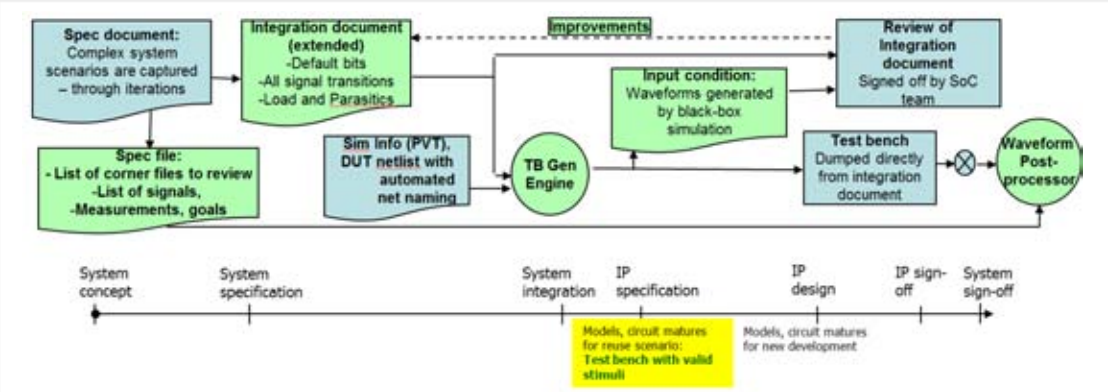

Figure 3. Automated IP level analog test bench generation

B. Automated RTL & BMOD STUB generation flow

The concept is further enhanced as shown in Figure 4, to automate sections of design RTL and ABMOD generation. All required information from a product or IP specification (in a mark-up language like HTML, XML, or spread sheet) is extracted through an XML parser [5], and then an intermediate spreadsheet (CSV) file is generated. This is further augmented either manually or through inputs from other collaterals to include information that is not available in the data sheet but are needed for the targeted automation. Using these information, a custom digital generation engine is developed to automatically generate stubs of RTL code of the design [6], and necessary information for verification including assertions and formal connectivity checks. Some of the information that can be achieved through this generation process includes pin multiplexing information for the interfaces including general purpose IOs, translation of register address mapping, input boundary condition assertions for ABMOD including reset/default states of the input control signals, and valid voltage/current levels.

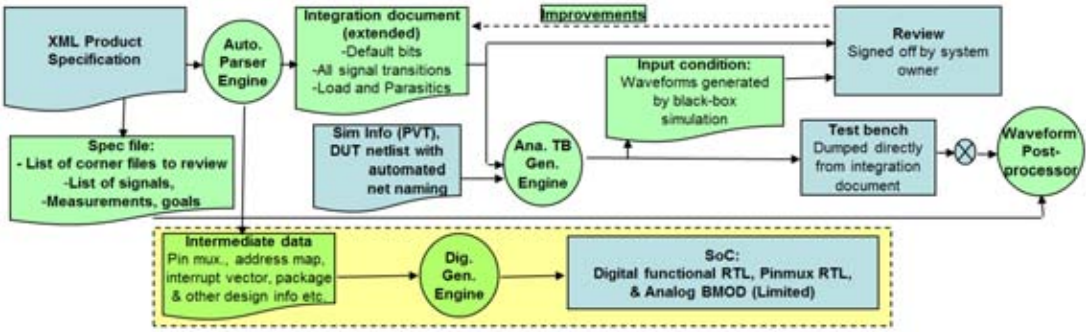

Figure 4. Automated RTL, ABMOD & test bench generation

C. Components of integration specification document

A typical ISD with bare minimum information will have the fields illustrated in List 1. While the below list is representative and shows some of the important entries, it is not a comprehensive list.

List 1. Fields of integration specification

1. *Sub-block: Identifies the unique sub-functional module of the IP.*
2. *Ball / Pin: Identifies the name of the package interface of the IC corresponding to the specific IP level interface port (it could be ball identifier for BGA packages or a pad/pin identification).*
3. *Port: It identifies all interfaces of the IP.*
4. *Direction: The signal direction, which could be input, output or inout / bidirectional.*
5. *Register: Identifies if this is mapped to a register interface.*
6. *Bits: Number of bits if the interface is a bus/vector.*
7. *Description: A verbose textual description of the interface for documentation purposed only and usually not used for any automation.*
8. *Default value: Default or power-up state of the interface.*

9. *Power-up value: Power-up state of the interface if different from default value at system reset. This is only meaningful in such cases where the system power-up transient state has to be identified separately from a reset state. This happens for a system with complete power management integration.*
10. *NVM: Identifies if this interface is mapped to a non-volatile memory interface like Flash or EFuse and the type. This helps in identifying a technology dependent un-initialised state of this interface.*
11. *Voltage domain: Identifies the related voltage/power domain.*
12. *Min., Nom., Max.: Minimum/Nominal/Maximum levels of the valid voltage/current. Important for power, and analog interfaces.*
13. *Associated Ground: If there are multiple grounds in the system*
14. *Current: Maximum allowed current.*
15. *IR drop: Maximum allowed IR drop.*
16. *Location: Physical coordinates of the related PAD interface.*
17. *Source: Identifies the source, if it is an input signal. It could be either board level, system level, another IP in the system or internal to the IP.*
18. *Test Scan: Identifies if it is a test interface used during system scan test mode or default test mode.*
19. *Test Iddq: Identifies if it is a test interface used / exercised in IDDQ test mode.*
20. *Input / Output capacitance: Parasitic input capacitance estimate or maximum output capacitance that can be driven if it is an output.*
21. *Leakage current (if Power): This quantifies the leakage current consumption if it is a power or ground interface in power-down condition.*
22. *Type (A/D/C): Identifies if the interface is analog, digital or a clock.*
23. *P/G/S: Identifies if the interface is a power, ground or signal.*
24. *S/A: Identifies if the interface is synchronous or asynchronous in nature. It is usually relevant only for digital interfaces and for those interfaces with timing criticality.*
25. *Pad/Signal: If the signal a PAD connected external interface. It is usually needed for ESD requirements.*
26. *Control H/L: Active high or low control.*
27. *IO type: Additional custom information like CMOS or WIRE compatible connections and if it is tri-stated.*
28. *Min. Route Spacing, Min. Route Width: Information for physical design constraints.*
29. *Guard ring: If guard ring protection is needed and the type, namely co-axial or micro-strip etc.*
30. *Active power: Power consumption level, usually needed for power/ground interfaces.*
31. *LDO SD condition: If the interface is an output of an integrated power-supply, the logical shutdown/power-down condition. This is used for power intent specification.*
32. *Block level interface: Other interfaces / connections for integration purpose.*
33. *Remarks: Verbose text for documentation.*

The content of ISD is enhanced with the fields shown in List 2 that capture information uniquely needed and necessary for test bench automation. Again this list is not comprehensive but representative for illustration purposes. The Stimuli field shows a useful input format called PBIT that allows to specify a signal pattern.

D. Implementation

An engine is built using Perl script, which parses the ISD to extract necessary information. This information

List 2. Integration document enhancements for test bench automation

1. *Feedback point*
2. *Routing capacitor, Routing resistor, Routing inductor*
3. *Load capacitor, Cap ESR*
4. *Load inductor, Ind. ESR*
5. *Load resistor*
6. *Stimuli: PBIT [dc voltage, value for 0 bit, value for 1 bit, 0-1 time delay(s), 0-1 rise time(s), 1-0 time delay(s), 1-0 fall time(s), bit transition time(s), delay time(s), periodic / non-periodic], bit value pattern(1/0)*

thus extracted is used in combination with the additional information available as the user configuration illustrated in List 3, usually needed as simulation information, to generate required test bench for simulation.

List 3. Contents of simulation information file

```
$voltage_corner="typ";              # typ, min or max
$temperature=27;
$process="nominal";                 #nominal, weak, strong, strong_lkg, skewnp or skewpn
$instance_name="I0";                #NET name prefix used in spice file
$netlist_path="/sim/tb_DUT.spi";
$model_path="/db/pdk/tech_node/models//model.paths.nom";
# input for transient simulation
$TSTEP="10e-9";
$TSTOP="5e-3";
$TPUNCH="100e-6,400e-6,600e-6,3.4e-3,3.7e-3,4.0e-3";
$int_doc_file="/sim/dut_int_doc_1p0.csv";
$tb_file_name="atb_DUT.spi";
# Optional (Default=No)
$auto_file_name="yes";
# Optional (Default=PWD)
$run_path="/sim/DUT/TESTBENCH_AUTO";
# Optional (Default=0)
$debug=1;
$debug_log="debug.log";
```

The design preparation phase includes generating a compatible design netlist with the DUT instantiated in a test bench with all the DUT terminals connected to net names with an aligned consistent naming convention, one such example would be to include the instance name prefix to the DUT symbol pin name as the unique test bench level net name connecting to that pin. An *auto_file_name* option would allow the tool to use a default naming convention for all output files and other artefacts.

E. Automation of results analysis

As already discussed in section III.B, assertions (including analog checks) can be generated using the above infrastructure, there is additional automation infrastructure as shown in Figure 3 & Figure 4 is implemented for automation of post-processing of analog simulation results (waveforms). This requires as a primary input a specification file (a sample is shown in Example 1) consisting of various measurements on identified signals, the goals (or valid limits) and the different table section in which the consolidated results are to be reported.

Example 1. Measurement specification file

```
* Defines a new report table by name "Power-up Sequence" with one column
*{NEW_TABLE "Power-up Sequence<br>" "<br>" "Delay" }
* Measuring delay between VI0_EN_1P8V & VXIO.GEN_1P8V
*VXI0.GEN_1P8V {DELAYXX 1 1.6 VI0_EN_1P8V 1 1.3 >500e-6&<540e-6}
        * Defines a new report table by name "Power-up Measurements" with 7 columns
*{NEW_TABLE "Power-up Measurements<br>" "<br>" "Value before PORZ<br>(V or A)" "Value after
PORZ<br>(V or A)" "N<sub>TR</sub>" "T<sub>TR</sub><br>(Secs)"
"T<sub>R/F</sub><br>(Secs)" "Dip<br>(V or A)" "Peak<br>(V or A)"}
VI0_EN {GETY 0 <10e-3} {GETY 4.5e-3 >=1.4}
VXI0.VDD_INT_1P2V {GETY 0 <10e-3} {GETY 4.5e-3 >0.9&<1.4} {dip 110e-6 5e-3 >0.8&<1.2} {peak
15e-6 5e-3 <1.3}
VI0_VDD_AD_1P8V {GETY 0 <10e-3} {GETY 4.5e-3 >=1.6}
VI0_VDD_CORE_1P2V {GETY 0 <10e-3} {GETY 4.5e-3 >=0.8} {dip 4.1e-3 5e-3 } {peak 4e-3 5e-3}
VI0_VDD_SRAM_1P2V {GETY 0 <10e-3} {GETY 4.5e-3 >=0.8} {dip 4.2e-3 5e-3 } {peak 4e-3 5e-3}
VI0_PORZ_CORE_1P2V {GETY 4.0e-3 <10e-3} {GETY 4.5e-3 >=0.8}
```

It is implemented using a proprietary waveform post-processor (TISpice PUNCH waveform format) interface using Perl, and an extension for FSDB file format is currently supported. This post-processor processes the saved waveforms from analog simulations, and performs all operations in the measurement specification file and generates an output report in HTML format that consists of several tabulated results.

The measurement specification file is defined with a custom syntax briefly explained as follows and whose user interface details are illustrated in List 4.

Any line starting with a*"*"* is comment and ignored except the pattern *"*{"*. This is used to define new results section that would be reported as a new table in the HTML output. All other measurements are specified as per the syntax below, in which each *<measurement_operation_#>* is reported on a separate column of the report table.

<signal_name> {<measurement_operation_1> [<arguments>] [<signal_name] [<goal>]} …
[{<measurement_operation_n> [<arguments>] [<signal_name] [<goal>]}]

A new table is created with the following command:

**{NEW_TABLE <title_as_quoted_string> <comments> <column_heading_1> … <column_heading_2>}*

List 4. Illustration of user interface for waveform post-processor

Punch2report_pc.pl [-help] [<punch_list_file> <signal_list_file> <report_name>]

-help *Provides usage information on command line*

<signal_list_file> *The file that specifies the signals to be processed, type of processing / measurement, intended goals for comparison and highlighting in the report and an inherent report content formatting.*

Comment:
***

User format for table generation:
**{ NEW_TABLE <argument_list>}*
**{NEW_TABLE <title_as_quoted_string> <comments> <column_heading_1> … <column_heading_2>}*

Signal processing:
<signal_name> [time] [{command [args]}]
<signal_name> {<measurement_operation_1> [<arguments>] [<signal_name] [<goal>]} … [{<measurement_operation_n> [<arguments>] [<signal_name] [<goal>]}]

Waveform measurements & Commands supported:
GETX, GETY, AVG, RMS, FREQ, DELAYXX, DIP, PEAK, SET_THRES, SET_GP, SET_TOL, TRANS, TRANSITION_TIME, RISE_FALL_TIME, NEW_TABLE

Macros:
Combination of the above can be defined as a user defined macro

IV. FUTURE SCOPE

The following are some significant extensions of this work identified to make this more formal, useful and easy for interception: a) Identification and extraction of relevant information from specification or integration document and include them IPXACT so that formal tool flows available for system integration supported by IPXACT are utilized and vice-versa; b) Current implementation of analog test bench generation only support a sub set of analog functionality and simulation aspects, and hence identifying, formalizing and intercepting all requirements needed to target simulation of wider analog functional portfolio; c) Verification of SoC memory map, connections such as DMA triggers and interrupts; d) Supporting synthesis/STA timing constraints generation; e) Analog assertions generation.

V. CONCLUSIONS

We proposed and implemented specification driven automation to generate correct-by-construct a) analog test bench for simulation, b) RTL design for IO pin multiplexing and register address mapping, and c) simulation results waveform analysis for analog functions. Application of this method improves efficiency and comprehensiveness of verification of analog functionality rich systems (IP) and enables an early closure of the same, by improving the communication across complementary competencies of product development. While the current flow uses the specifications in spreadsheet or XML formats as input, further extensions of this technique by standardizing their interoperability with IPXACT based flows would make their adaptability broader.

ACKNOWLEDGMENT

We are thankful to Keith Kunz, Neeraj Saxena & Padmini S of Texas Instruments (TI) for their continuous support and motivation; Ravi C V, and Harikrishna P of TI for their support in interception of these methods in several product executions. We thank the specific technical contributions of Devendra S, Sonal R Sarthi, Rajesh Kedia & Cletan Sequeira for their support in implementing the RTL generation aspects.

REFERENCES

[1] Buss, D.D., Chatterjee, A., Efland, T.R., Evans, B., Goodpaster, H.D., Haroun, B.S., Hellums, J.R., Krenik, W.R., Morton, A., Schichijo, H., Tsai, C.-Y., Vrotsos, T.R. 2000. DSP & analog SOC integration in the Internet era. IEEE Emerging Tech. Symp.: Broadband, Wireless Internet Access. Apr. 2000, 1-5. DOI=http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=916520

[2] Moloney, A. 2006. Power Supply Management – Principles, Problems, and Parts. Analog Dialogue, Vol 40. Analog Devices Inc. May 2006. DOI=http://www.analog.com/library/analogdialogue/archives/40-05/power.pdf

[3] Manninger, M. 2007. Power Management for Portable Devices. Proc. of 33rd ESSIRC. 2007, 167-173. DOI=http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4430272

[4] DocZone system for capturing Spec. information in XML. DOI=http://www.doczone.com

[5] Xerces XML parser. DOI=http://xerces.apache.org/xerces-c

[6] Lele, A., Suryaprabha, D., Kedia, R., Sequeira, C. 2014. Automating SOC development using machine readable specifications. Designer Track, 51st DAC. 2014.