A Holistic Approach to Low-Power, Mixed-Signal Design Verification Using Power Intent

CPF-Based Interface Elements, Methods, and Guidelines

Vijay Kumar Sankaran
Custom IC, Cadence Design Systems (India) Pvt. Ltd., Bangalore, India
Email: vijay@cadence.com
Phone: +91-9916856489

Lakshmanan Balasubramanian
Embedded Processing, Texas Instruments (India) Pvt. Ltd., Bangalore, India
Email: lakshmanan@ti.com
Phone: +91-9840468408

Bharath Kumar Poluri
Embedded Processing, Texas Instruments (India) Pvt. Ltd., Bangalore, India
Email: bharath.poluri@ti.com
Phone: +91-9740162340

Venkatraman Ramakrishnan
Embedded Processing, Texas Instruments (India) Pvt. Ltd., Bangalore, India
Email: rvenkat@ti.com
Phone: +91-80-25345454

Badrinarayan Zanwar
Custom IC, Cadence Design Systems (India) Pvt Ltd., Bangalore, India
Email: bzanwar@cadence.com
Phone: +91-9945234256

Qingyu Lin
Product Engineering, Cadence Design Systems, Inc. San Jose, USA
Email: qingyu@cadence.com
Phone: +1-4089447309

Abstract: Today's complex, low-power analog and mixed-signal (AMS) systems on chip (SoCs) are comprised of logic (boolean, real) and transistor-level abstractions for design implementation, verification, validation, and test readiness. This situation mandates extensive use of AMS co-simulation[1][2][3]. Such designs are increasingly becoming power managed with multiple power/multi-voltage domains by nature. There are various techniques for verification and validation of these low-power designs, but most of them either have dependencies on the block-level designer to use the right power domain for a multi-power design, or require lot of manual effort for setting up the flow, which is error prone. This paper discusses the verification of AMS SoCs in a very effective way by using the power intent information in the form of Common Power Format (CPF). We begin with discussing the traditional methods[4][5][6] used for insertion of interface elements (IEs), also called connect modules (CMs), to handle inter-discipline signal traversal, followed by how AMS-CPF based IEs proved to be very simple, straightforward, and accurate for top-level co-simulation.

I INTRODUCTION

Traditionally, Analog and Mixed Signal (AMS) verification was carried on by introducing power supply sensitivity for logic-to-electrical signal conversion (and vice versa) by one of two methods, viz., static supply voltages that perform conversion based on fixed thresholds, or to use a special type of interface elements (IEs) or connect modules (CMs) named inherited IEs[4] that inherit a user-specified power supply for signal conversion. In this method of inherited IEs, we first used the CPF[5][6] (from a backend/physical design implementation phase, the Cadence® Encounter® Digital Implementation™ tool suite, for example) to infer the power domain information and then post-processed this information to define the block disciplines needed for AMS or digital mixed-signal (DMS) simulations and create the IE groups that inherited the supplies based on the block disciplines assigned (using Cadence - *setdiscipline* constructs), as illustrated in Figure 1.

However with multi-power domain designs, the complexity scales up for the discipline specification both with static and inherited IEs. It is a manual, error-prone, effort-intensive task to ensure correct discipline specification at the digital boundaries. Though this post-processing can be done through automation scripts and is a one-time effort, for SoCs with many voltage and power domains, creating multiple IE groups require considerable effort and time.

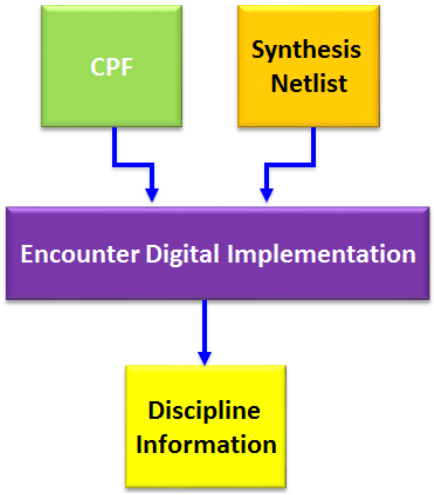

Figure 1: Traditional verification approach towards setting domain disciplines in AMS simulations[6]

In the proposed and implemented method, we discuss making use of CPF directly for implementing supply sensitivity in AMS-RTL simulations[7] and thus eliminating all the manual effort needed to define the block disciplines for IE insertion. We will also see that the flow is more accurate, enables reuse of existing power intent information provided by the designer, and eliminates false failures because of incorrect supply information,which is very costly for AMS simulations in terms of their run times.

II OVERVIEW OF THE TRADITIONAL CPF-DERIVED INHERITED-IE METHOD

Before we delve into mixed-signal design verification directly using power intent in the form of CPF, let us have an overview of the traditional inherited IE method. As explained above, the power intent information is captured in the form of CPF and is considered as the golden reference for the given IP or SoC. Most of the recent flows (for example, Encounter Digital Implementation tool suite and NC-Verilog® simulator with low-power capability) use this information in all stages of the design to implement/verify the power intent, which is sufficient enough to know voltage or power domain of each pin of the IP it belongs to. For example, some typical Encounter Digital Implementation commands used to get the pin power domains are shown in Figure 2.

```
foreach sig_pin [dbGet [dbGet -p1 top.insts.name
$macro].instTerms.name]
MSV::getInstTermPowerDomain $sig_pin
```

Figure 2: Encounter Digital Implementation commands to get the pin domain information[6]

The power domain information thus obtained is then post-processed to get the discipline information that a functional simulation tool (like NC-Verilog) can understand. Figure 3 illustrates the complete flow where the CPF and a synthesized gate-level netlist are passed to the Encounter Digital Implementation tool along with the Tcl file containing the above commands for power domain identification. The first-level output of this flow is the power-domain information that is then passed through a PERL script to create the constructs that the simulator can understand to run AMS co-simulation. The constructs are those of Cadence's Incisive® AMS Use Model[3] flow, which uses the IRUN command for the simulation.

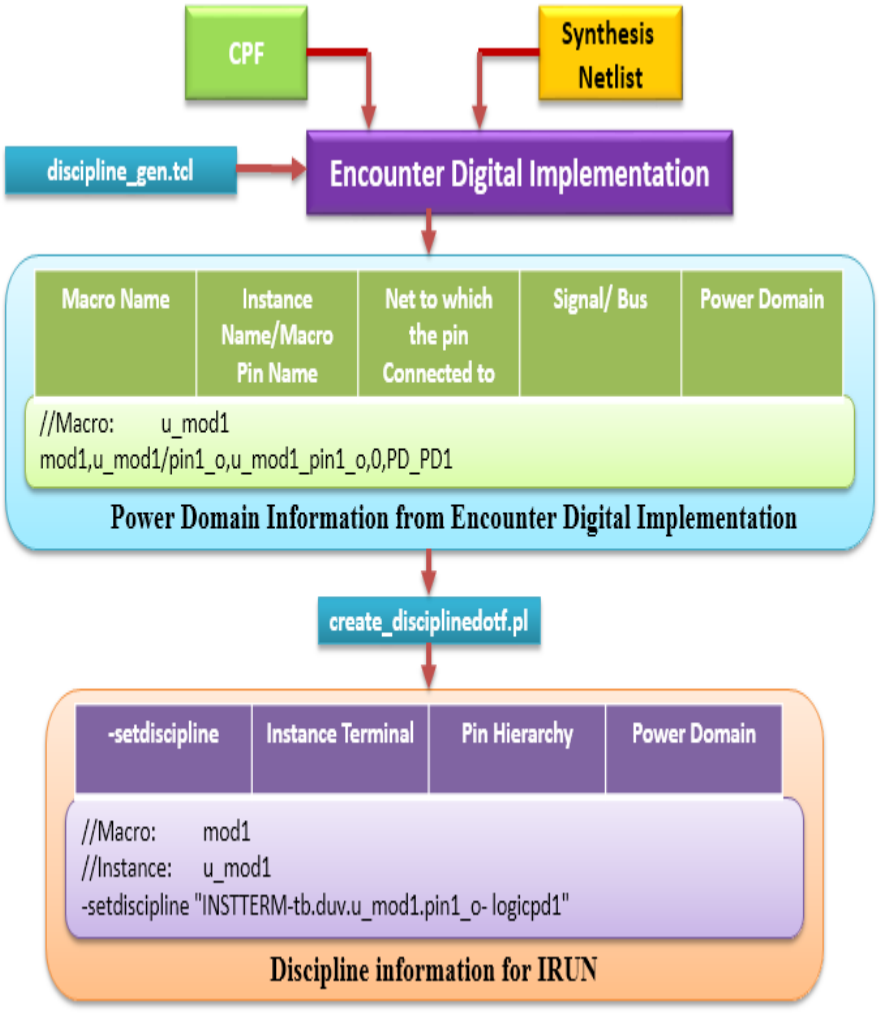

Figure 3: Complete flow of CPF-derived discipline information[6]

Once the full list of *–setdiscipline* (or *–setd*) constructs is available for all the boundary ports of the IPs, it is time to configure the IEs by associating them with the discipline names in the *–setd* constructs. Suppose an SoC contains four power domains, viz, avdd (always on domain), vddc (digital core domain), vddm (memory power domain), and vddsw (switchable domain). For each of these power domains, unique discipline names are assigned in the above

flow. Then the connect rules and CMs for an inherited IE (constituting an IE group) are customized (as a copy from the Cadence software installation path) for each of the power domains. Thus in our case, four IE groups are created.

IE Groups	Connect Rule	IEs
Connect_pd1	CR_full_pd1	E2L_full_pd1
Connect_pd2	CR_full_pd2	E2R_full_pd2
Connect_pd3	CR_full_pd3	R2L_full_pd3
Connect_pd4	CR_full_pd4	L2R_full_pd4

Figure 4: IE groups for different power domains in an SoC[6]

Each IE group will have a default discipline, such as logic, in it. This logic discipline is replaced by the discipline associated with each power domain. Accordingly, the IEs or the CMs inserted into the design during elaboration will be named with the corresponding discipline name as the suffix (illustrated in Figure 4).

The semantics of an IE group consisting of connect rules and CMs for an inherited IE from Cadence installation and how the discipline name is specified in the connect rule is illustrated in Figure 5.

```
discipline logicpd1
                domain discrete;
enddiscipline
```

```
electrical
   (* integer inh_conn_prop_name="pd1";
     integer inh_conn_def_value="tb.ams_pd1"; *) \pd1! ;
 electrical
   (* integer inh_conn_prop_name="gnd";
     integer inh_conn_def_value="tb.ams_gnd"; *) \gnd! ;
```

```
irun  -f discipline.f <other_switches>
 -amsconnrules CR_inhconn_full_pd1     connect_pd1/*.vams
 -amsconnrules CR_inhconn_full_pd2     connect_pd2/*.vams
 -amsconnrules CR_inhconn_full_pd3     connect_pd3/*.vams
 -amsconnrules CR_inhconn_full_pd4     connect_pd4/*.vams
```

Figure 5: Semantics of an IE group and constructs to use IE groups in the simulator[6]

As seen above, the switch *–amsconnrules* is used to compile the IEs from each IE group along with the design source codes. All the *–setd* constructs can be placed in a file, say disciplines.f, and this file can be added to the *irun* command. During the discipline resolution, the tool uses discipline information of each interface and picks up the appropriate IE. The IEs that are inserted into the design based on the *–setd* commands can be seen by referring the file named ams_ieinfo.log created after the elaboration. This file is created by adding the switch *–ieinfo* to the *irun* command.

A snippet of the IEs inserted at various levels in the design as shown in ams_ieinfo.log (Figure 6).

```
11. Interface Elements at the block <instance> testbench.u_duv of <master> duv (file : ./duv.vams)
Automatically inserted : testbench.u_duv.n6__E2L_2_inhconn__ddiscrete_inhconn_full_fast
 Connect Module : E2L_2_inhconn
 Mode :           Merged
 Net :            testbench.u_duv.n6 (discipline: electrical, nettype: electrical)
 Port :           testbench.u_duv@duv<module>.I7@INV_F1_3P3V<module>.A (discipline: ddiscrete_inhconn_full_fast, direction: input, nettype: wire)
 Parameters :
                 vsup_min   :0.5
                 vthi       :0.6666666666666666
                 vtlo       :0.3333333333333333
                 vtol       :0.08333333333333333
                 vtlox      :0.4166666666666666
                 vthix      :0.5833333333333333
                 tr         :2e-10
                 tf         :2e-10
                 ttol_t     :2e-11
                 tdelay     :0
                 txdel      :8e-10
                 ttol_c     :5e-11
                 vsup_off   :0.4166666666666667

 List of Ports connected to net testbench.u_duv.n6 : (Total: 2)
                 testbench.u_duv.I7.A (ddiscrete_inhconn_full_fast input)
                 testbench.u_duv.I6.Y (electrical output)

 Discipline of Port (Ain): electrical, Analog Port

 Discipline of Port (Dout): ddiscrete_inhconn_full_fast, Digital Port

 Drivers of Port Dout: No drivers

 Loads of Port Dout: No loads

 Sensitivity information:
                      No Sensitivity info
```

Figure 6: IEs inserted into the design based on the discipline

Though it is a one-time effort to create IE groups, it requires considerable effort to create such IE groups for SoCs with multiple voltage and power domains. In the next section, we shall see how using CPF directly on an AMS RTL design eliminates these efforts and will see how much time and effort is saved by using very minimal constructs and set up.

However, we need to keep in mind that today's use of CPF for RTL AMS co-simulations is possible only when we make the physical electrical supply connection from the design under test (DUT) to the IPs instantiated by it. This is because CPF constructs as of today do not support supply port connections, as they work only based on the power domains that the ports belong to. By extension, any supply variation (fluctuation) can not be tracked by CPF, so we must have a feature where we can model the supply variations in the CPF file.

In order to overcome this limitation, work is under progress by Cadence to enable supply port connection in CPF as a proprietary feature so that the simulator can track the supplies continuously and perform signal conversion, which gives rise to fully automatic true supply sensitivity through CPF. Currently, this physical electrical supply connection is needed only at the DUT level or at the level where the electrical and logical design partitions interact directly. Once the work to comprehend CPF supply connections is complete, we can use a Verilog-D DUT without physical power connections, and the tool can automatically handle signal conversion based on supply information through CPF.

As we need to make the physical electrical supply connection from DUT to the IPs, we need to have the DUT coded in Verilog-AMS, as Verilog-D does not support electrical supply ports. In the next section, we will see how this has been implemented in an industrial testcase by first going over basic CPF constructs, and then illustrating the case with CPF-based IEs for AMS simulation.

III PROPOSED CPF-BASED IEs FOR MIXED-SIGNAL DESIGN VERIFICATION

We will begin with a quick overview of some basic CPF constructs so that one may understand how power intent is captured for the IPs and how this power intent information is used by the simulator for classifying the blocks under respective power domains.

To identify portions of the design that operate on the same voltage, associate these portions with a power domain using the *create_power_domain* command.

create_power_domain -name power_domain [-instances instance_list] [-boundary_ports pin_list] [-default] [other options…]

To specify additional information that applies to power and ground routing, use the *update_power_domain* command.

update_power_domain -name domain { -primary_power_net net | -primary_ground_net net } [other options…]

When instantiating a macro cell in the design, indicate which CPF macro model applies to the specified instance using the *set_instance* command.

set_instance <inst_name> -model <cell_name> -domain_mapping { {PD_1 PD_1 } {PD_3 PD_4 } }

To identify which cells in the libraries can be used as special cells for power management functions like state retention cells and level shifter cells, use the below commands.

*define_state_retention_cell -cells *DRFF* -restore_function RETN*
create_state_retention_rule -name string { -domain power_domain | -instances instance_list } [other options…]
define_level_shifter_cell -cells LVLLHEHX -input_voltage_range 0.8 -output_voltage_range 1.0 -output_power_pin VDD -ground VSS*
create_level_shifter_rule -name lsr1 -from PD1 -to PD2
create_level_shifter_rule -name lsr2 -from PD2 -to PD3

The definition of a CPF macro model starts with a *set_macro_model* command that specifies the name of the library cell that represents the macro cell.

set_macro_model ABCD_203

In CPF, operating voltages are associated with nominal conditions. To specify the operating voltages used in the design, use the *create_nominal_condition* command.

create_nominal_condition -name string -voltage {voltage | voltage_list} [-ground_voltage {voltage | voltage_list}]
Examples:
create_nominal_condition -name N1 -voltage 1.2 -ground_voltage 0
create_nominal_condition -name OFF -voltage 0 -ground_voltage 0 -state off

To associate the nominal conditions with the power domains, use the *create_power_mode* command.

create_power_mode -name string -domain_conditions domain_condition_list –default
Example: create_power_mode -name PM2 -domain_conditions { PD_VDD1_VSS1@N1 PD_VDD2_VSS2@OFF }

End the CPF macro model with the below command.

end_macro_model

The industrial testcase used here is a complex AMS SoC (shown in Figure 7). This SoC uses four power domains as we saw in the example in the previous section. A high-level block diagram of this SoC is shown below, where we have digital (boolean, real) and analog (behavioral model, transistor level) for various modules based on their level of abstraction. We had used CPF-derived inherited IEs initially, and now we will see how CPF-based IEs are directly used for mixed-signal verification of this SoC.

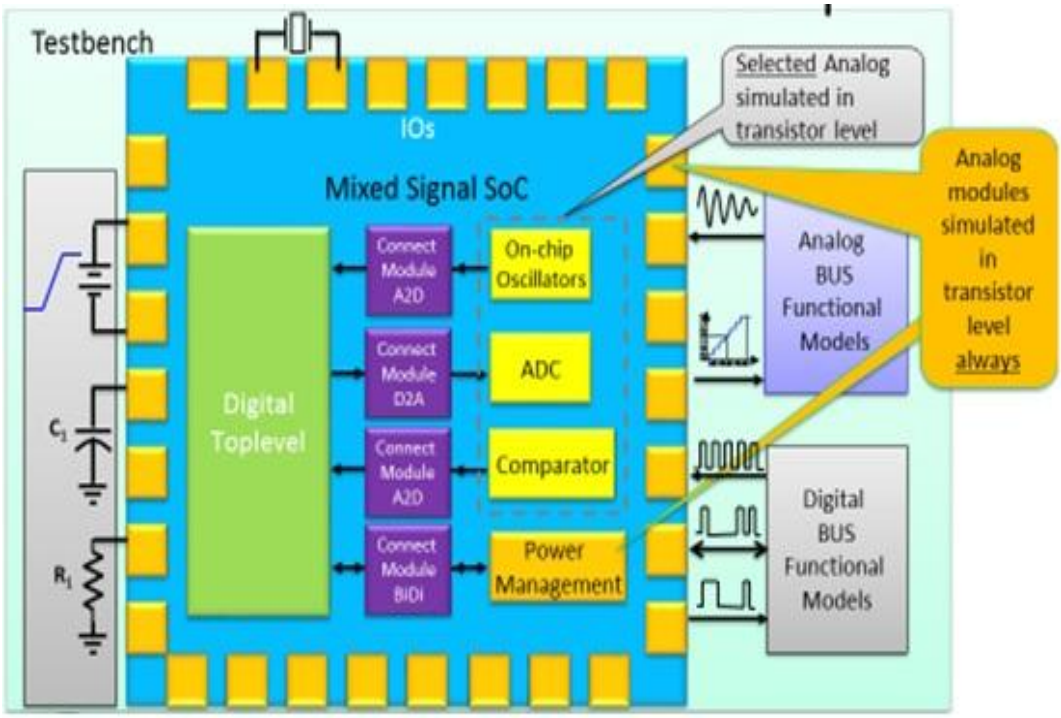

Figure 7: SoC comprising various levels of abstraction[6]

For this SoC, we use a Verilog-AMS testbench and a Verilog-AMS DUT to establish the physical supply connections at the DUT level, to cover the limitation (of CPF) as mentioned in Section II. As seen below, the module named DUT has its input/output/inout ports along the supply/ground ports (VDD*/VSS*) and these supply ports have to be electrical by nature.

```
module DUT (P1, P2, P3, VDD1, VDD2, VSS1, VSS2,  F1, F2, F3);

inout VDD1, VDD2, VSS1, VSS2;
input P1, P2, P3;
output F1, F2, F3;

electrical VDD1, VDD2, VSS1, VSS2;
…
endmodule
```

This one-time effort overcomes the current limitation of the CPF lacking a supply port connection, and this workaround enables the physical supply connectivity between the DUT and the IPs. The rest of the design is governed by CPF, and signal conversion happens by following the power domain information from CPF.

Snippets of some of the power-intent information in the CPF for this SoC are shown below.

```
create_power_domain -name PD_1 –default
update_power_domain -name  PD_2 -primary_power_net VDD1 -primary_ground_net VSS1

set_instance u_inst_1 -model model1 -domain_mapping { {PD_VDD1_VSS1 PD_DIG_1 }  {PD_VDD2_VSS2 PD_VDD3 } }

create_state_retention_rule -name RET1 -instances    {
tb/duv/u_dig/u_ip_wrap/u_0 tb/duv/u_dig/u_ip_wrap/u_1 } \
-save_edge  {tb/shutoff_condition }\
-restore_edge  {!tb/shutoff_condition }

set_macro_model flash_ ip
create_nominal_condition -name N1 -voltage 3.3 -ground_voltage 0
create_nominal_condition -name OFF -voltage 0 -ground_voltage 0 -state off

create_power_domain -name PD_VDD2_flash -boundary_ports { \
  flash_3P3V clk_3P3V en_flash_3P3V ……}

update_power_domain -name PD_VDD2_flash -primary_power_net \
  VDD_flash -primary_ground_net VSS  -equivalent_power_nets \
  { VDD2 VDD3 VDD4 } -equivalent_ground_nets \
  { VSS1 VSS2 VSS2 VSS4 }

create_power_mode -name PM1 -default -domain_conditions { \
  PD_VDD2_flash@N1 }
create_power_mode -name PM2 -domain_conditions { PD_VDD2_flash@OFF }

end_macro_model
```

(All of the above power intent information are samples for illustration purpose from the original SoC, and they don't represent the entire power intent information required for correct and comprehensive low power implementation and verification.)

With this information, the set up is ready for the simulation. We will see some of the *IRUN* options that need to be added to make use of the power intent information properly to insert power-smart IEs. The next section explains the final stage of the verification setup.

IV SIMULATING THE DESIGN USING CPF

The IRUN options to be added before starting the simulation include:

-amsconnrules CR_full_fast to use Cadence-provided built-in power-smart CPF-based IEs
-discipline logic to set a global logic discipline to the entire design
-ams_generate to support IEs inside VHDL generate statements
-lps_pmode to enable power-mode simulation

These options were added and the irun command was executed with all the other required source codes, analog control files, and AMS control files. Simulation was completed and results matched with the traditional inherited IE method. The turnaround time for the set up was reduced from few weeks to less than two days.

Figure 8 shows the snapshot of the ams_ieinfo.log file showing the CPF-based IEs. Note that CPF-based IEs have a __LPS suffix to denote low-power simulation.

```
11. Interface Elements at the block <instance> testbench.u_duv of <master> duv (file : ./duv.vams)
Automatically inserted : testbench.u_duv.n6__E2L_2_LPS__electrical
 Connect Module : E2L_2_LPS
 Mode :           Merged
 Net :            testbench.u_duv.n6 (discipline: electrical, nettype: electrical)
 Port :           testbench.u_duv@duv<module>.I7@INV_F1_3P3V<module>.A (discipline: electrical, direction: input, nettype: electrical)
 Parameters :
                vsup       :1.8
                vlo        :0
                vhi        :1.8
                vthi       :1.2
                vtlo       :0.6
                vx         :0
                tr         :2e-10
                tf         :2e-10
                ttol_t     :2e-11
                tdelay     :0
                rhi        :200
                rlo        :200
                rz         :10000000
                rx         :200
                txdel      :8e-10
                ttol_c     :5e-11
                vtol       :0.15
                vtlox      :0.75
                vthix      :1.05
                r_SUPPLY   :4
                r_STRONG   :200
                r_PULL     :1500
                r_LARGE    :9000
                r_WEAK     :55000
                r_MEDIUM   :320000
                r_SMALL    :1900000
                rth_SUPPLY :28.2842712474619
                rth_STRONG :547.7225575051662
                rth_PULL   :3674.234614174767
                rth_LARGE  :22248.59546128699
                rth_WEAK   :132664.991614216
                enable_highz :0
                duration   :5e-09
                vpso       :0.2


List of Ports connected to net testbench.u_duv.n6 : (Total: 2)
                testbench.u_duv.I7.A (logic input)
                testbench.u_duv.I6.Y (electrical output)

Discipline of Port (Ain): electrical, Analog Port

Discipline of Port (Dout): logic, Digital Port
```

Figure 8: Snippet of CPF-based IEs from ams_ieinfo.log file

Cadence's SimVision™ waveform viewer shows these CPF-based IEs inserted in the design hierarchy as shown in Figure 9, and enables plotting its internal signals just like inherited IEs. One can plot, trace, and look at the drivers of these signals, which show that the signals are driven from the CPF.

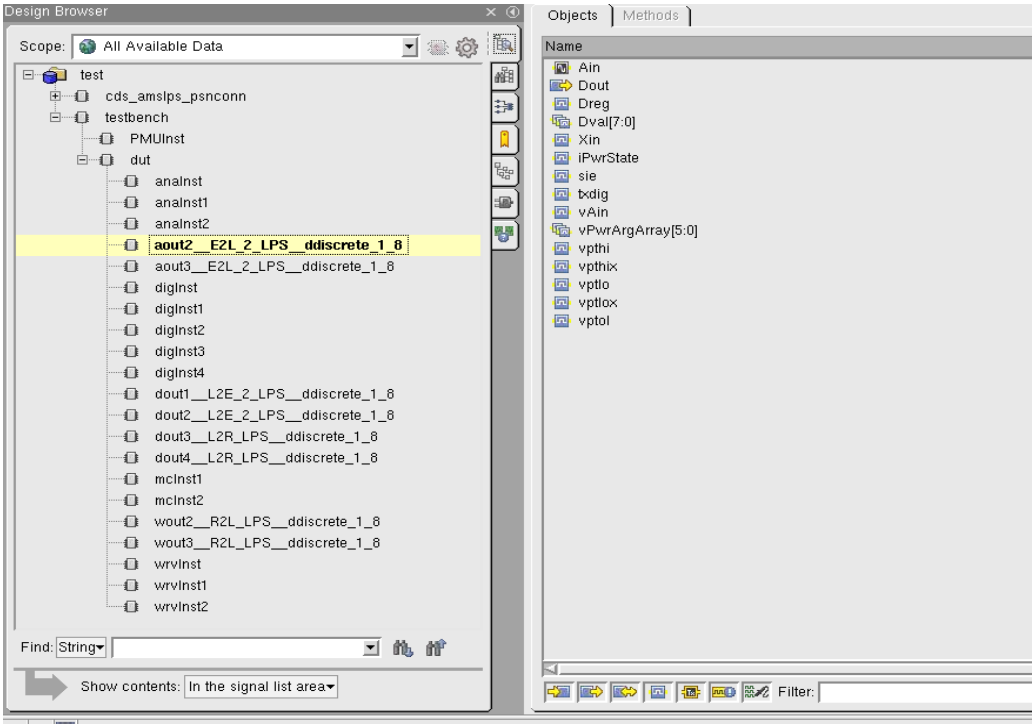

Figure 9: Snippet of various CPF-based IEs inserted into the design

In the next and final section, we will summarize the limitations and future scope of using CPF-based IEs for mixed-signal SoCs.

V LIMITATIONS AND FUTURE SCOPE OF CPF-BASED IEs

The two major limitations in the existing flow of using CPF-based IEs are explained below.

A. CPF-based IEs can be used only for RTL AMS co-simulations, but not for gate-level AMS co-simulations

This limitation is because the synthesized gate-level netlist has the physical power ports connectivity that will be overridden by CPF for domain selection and CM insertion. There is a need for AMS co-simulation using gate-level abstraction with timing annotation for covering timing-critical analog-digital interface paths. To support such requirements, we need true supply sensitivity without having to resort to direct use of CPF, since the use of CPF for supply sensitivity in AMS co-simulation overrides the actual physical/electrical supply connectivity available in the power-connected gate-level netlist.

B. Need for physical supply port connections in CPF to track the supply variations

As briefly mentioned in Section II, current use of CPF for RTL-AMS co-simulation is possible only when we make the physical electrical supply connection from the DUT to the IPs instantiated by it. Currently, CPF constructs do not support supply port connection, as they work only based on the power domains of the ports, so any supply variation (fluctuation) cannot be tracked by CPF. In order to overcome this limitation, work is under progress by Cadence to enable supply port connection in CPF as a proprietary feature, so that the simulator can track the supplies continuously and perform signal conversion accordingly. This feature will provide fully automated true supply sensitivity through CPF.

VI CONCLUSION

In this paper, the authors have presented the need for moving from traditional static or inherited IEs to CPF-based power-smart IEs. The flow of generating discipline information from CPF for inherited IEs was first explained as an interim measure before a fully automated flow, and the time and effort saved by using CPF-based IEs directly was explained. Basic CPF constructs with examples from an industrial testcase were shown. Practical usage of CPF and its limitations were summarized, with future scope and work under progress.

VII ACKNOWLEDGEMENT

We are thankful to Jagdish C Rao, Nikhil Chandrakant Sangani, Ajith Subramonia, and Deepak Choudhury of Texas Instruments, and Vishwajeet V Betai of Cadence Design Systems for their constant support and motivation. We acknowledge the specific technical contribution of Nan Zhang of Cadence Design Systems. We are thankful to Zhong Fan of Cadence Design Systems for the untiring support on AMS co-simulation methodology, technical discussions, brainstorming, and formal tool support.

References**:**

[1] Pooja Sundar, Lakshmanan Balasubramanian, Sandeep Tare, Satish Chandramohan, Charles Jiang, “Enabling Efficient AMS Co-simulation of Mixed-signal SoC with Analog and Power Management Integration”, SyNopsys User Group (SNUG) India, Bangalore, India, 2011.

[2] Chandrashekar B G, Chanakya K V, Lakshmanan Balasubramanian, Prathap Ghorpade, Badrinarayanan Zanwar, “Mixed signal SoC power-up simulations made possible at transistor level”, CDN Live India, Bangalore, India, 2011.

[3] Cadence Design Systems Inc., “Workshop for AMSD Incisive Use Model”, Revision 1.7, AMS Product Engineering, March 2009.

[4] Lakshmanan Balasubramanian, Bharath Kumar Poluri, Shoeb Siddiqui, Vijay Kumar Sankaran, "Efficient methods for analog mixed signal verification – Interface handling methods, trade-offs and guidelines", DVCon, Bangalore, India, 2014.
[5] Bharath Kumar Poluri, Atul Ramakant Lele, Rajesh Kedia, Lakshmanan Balasubramanian, "Unified flow using CPF derived Inherited Connect Modules for Digital and Analog Mixed Signal verification," Designer Track, 51st DAC, San Francisco, USA, 2014.
[6] Bharath Kumar Poluri, Atul Ramakant Lele, Aswani Kumar Golla, Lakshmanan Balasubramanian, "Fully automated interface elements insertion for Digital and Analog Mixed Signal verification in multi power domain designs", 52nd DAC, San Francisco, USA, 2015.
[7] Ian Dennison, "Cadence Mixed-Signal Solutions: Technology Update", CDNLive India, Bengaluru, India, 2014.