DVS Interface Element - A Novel Approach in Multi-Power Domain, Mixed-Signal Design Verification

With a closer look at its emergence and holistic features

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

Ji Du
Software Engineering, Cadence Design Systems, Beijing, China
Email: duji@cadence.com
Phone: +861056812036

Qingyu Lin
Product Engineering, Cadence Design Systems, Inc. San Jose, USA
Email: qingyu@cadence.com
Phone: +14089447309

Arumugan A
Product Validation, Cadence Design Systems India Pvt Ltd, Bangalore, India
Email: arumugan@cadence.com
Phone: +91-9945128280

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

Abstract: In today's mixed-signal design development, verification engineers are imposed with the big challenge of designing and verifying systems on chip (SoCs) with ever-increasing complexity in terms of low power consumption, better performance, and above all, cost reduction. It is imperative that verification tools be mature enough to provide features that account for efficient verification, so as to avoid common pitfalls such as inaccurate or faulty execution that misses the intended coverage, which might lead to design re-spin and re-engineering. Over time, EDA tools have phenomenally improved to handle verification of multi-power domain designs using various methodologies like supply sensitivity, supply inheritance, and low-power schemes like CPF and IEEE 1801. However, a much simpler approach of specifying a dynamically varying supply for analog/mixed-signal and digital/mixed-signal (AMS/DMS) co-simulation is always welcome and is a truly fair approach for a novice verification engineer. In this paper, we discuss a dynamic voltage supply (DVS) for the interface elements (IEs, commonly known as connect modules or CMs) implemented using two new system functions that have been introduced in Accellera Verilog-AMS LRM version 2.4.0[1]: $analog_node_alias and $analog_port_alias (section 9.20 Analog node alias system functions)[1] and a Cadence® proprietary system function named $real_net_alias. These system functions are used for constructing DVS[2] CMs that can be used in mixed-signal simulations.

I. INTRODUCTION

Traditionally, digital- and system-level focus has dominated system-on-chip (SoC) verification, but recent trends mandate varying levels of device/transistor-level comprehension during verification to get the required focus on analog/mixed-signal (AMS) contents. A high-level view of a typical SoC-level AMS co-simulation environment is illustrated in Figure 1, where we see the big picture of the verification intrinsics needed to handle the testbench (could be a mixed-signal testbench coded in Verilog-AMS), the analog and digital top level comprising the design under test (DUT), corresponding bus functional models (BFMs), and selected analog modules in the transistor level to realize their power and accuracy.

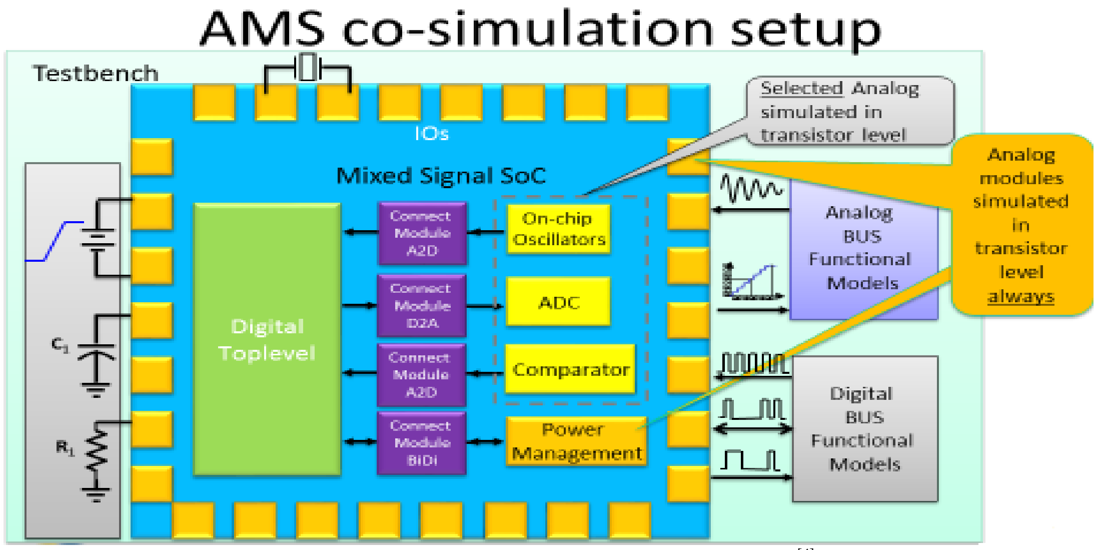


Figure 1: Typical SoC-level AMS co-simulation environment[4]

Such a complex power-managed SoC demands an accurate verification methodology to take care of various factors at the broader level, such as domain crossings, inter-discipline signal traversal, circuit conditions or states (on/off), etc. In order to verify these intrinsics, the verification engineer has a lot of dependence on the IP owners of the blocks and must specify the voltage/power domains accordingly before starting the verification. Though there are different methods provided by EDA vendors to set the disciplines/domains, it is not a trivial task as the power domain specification must be accurate, else it shows functional failure in the simulation, leading to waste of time in the verification cycle.

Dynamic voltage supply (DVS) connect modules (CMs) simplify this task by specifying the power domain for a particular net, instance port, an entire instance or all instances of a cell, or even an entire library wherein the user uses simple existing constructs from the Cadence Incisive® AMS Use Model[2] to specify the supply and ground signals to be used as reference for signal conversion from analog-to-digital and vice versa. We will have a brief look at the existing different flavors of CMs and then have a deeper look at the DVS CMs to describe the simplicity of its use model, accuracy, and lesser turnaround time. We will also see the limitations of DVS CMs and understand where they may not fit in as of today, and where they might need enhancement to provide comprehensive coverage for AMS/DMS verification.

II. EXISTING FLAVORS OF CMs

Depending on the details that are being focused on during AMS co-simulation, the CMs may have multiple requirements, including the following:

- CMs must be aware of the related supply in either a static or dynamic way
- CMs must handle either potential (voltages) or flow (currents)

- CMs must reflect or inherit the port impedance

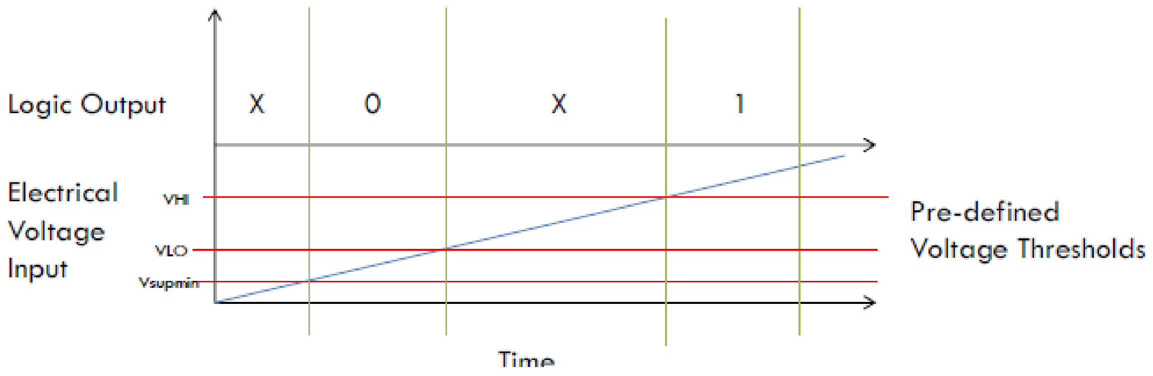


Figure 2: Expected behavior of an E2L CM[4]

Based on the supply sensitivity, the CMs may be further classified as following:

- Static or non-supply aware or non-power aware, and
- Supply sensitive
  - Through global node, also called supply inheritance
  - Through actual physical electrical supply connectivity

a) Static or non-supply-aware or non-power-aware CM

Static CMs work based on fixed voltage values or thresholds that will be used for analog-to-digital (A2D) and digital-to-analog (D2A) conversion. These can be used for designs that are non-power-managed, and whose supplies do not exhibit any variation, such as power ramp up/down sequence, and also cannot track the fluctuations in the supply.

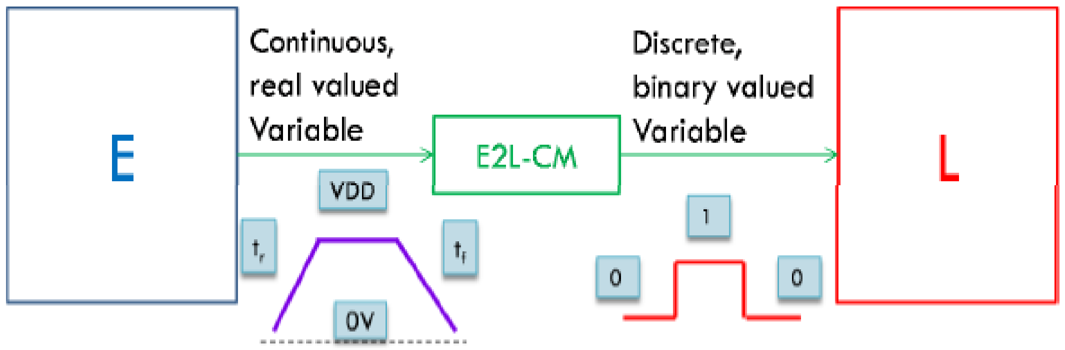

E
Continuous, real valued Variable
VDD
t_r
t_f
0V
E2L-CM
Discrete, binary valued Variable
1
0
0
L

Figure 3: Illustration of a static E2L CM[4]

The use model of a static CM is quite simple and it complies with the Incisive AMS Use Model[2]. The semantics are as follows.

```
amsd {
ie vsup = 1.2 cell = “DIGTOP” inst = “top.u_anatop.u_dig.u_bgap”
}
```

We see the digital cell named DIGTOP and the digital instance named top.u_anatop.u_dig.u_bgap will be using a static supply threshold of 1.2V (practically has vthi and vtlo threshold parameters to take care of X states). Any analog block(s) that is associated with these digital blocks would be using a fixed supply threshold of 1.2V, irrespective of the power domain of the analog or digital blocks. Thus, we see this method provides static supply sensitivity and is not a fool-proof method to address the verification challenges of today’s complex low-power mixed-signal SoCs.

b) Supply-sensitive CM

We will see the two methods of using supply-sensitive CMs, viz, supply inheritance and physical electrical supply connectivity.

b1. Through global node, also called supply inheritance

Supply inheritance is a method wherein the user uses a global electrical node (usually from the testbench) as the reference for signal conversion. As such, the CMs used in this method are also termed inherited CMs, because these CMs inherit a global supply from the top level. In this method, the user does not have to modify any source code for a block to inherit the top-level supply, rather he needs to modify the Cadence-provided built-in inherited CMs. The user may need to have a local copy of these CMs from the installation, and configure them for referencing the supply and ground signals from the design.

A diagrammatic representation of an inherited CM is shown in Figure 4, where the electrical output of an analog block drives the logic input of a digital block by referencing a top-level global supply voltage VDD. This top-level supply is usually from the testbench that defines the global power supply and ground connections for the SoC.

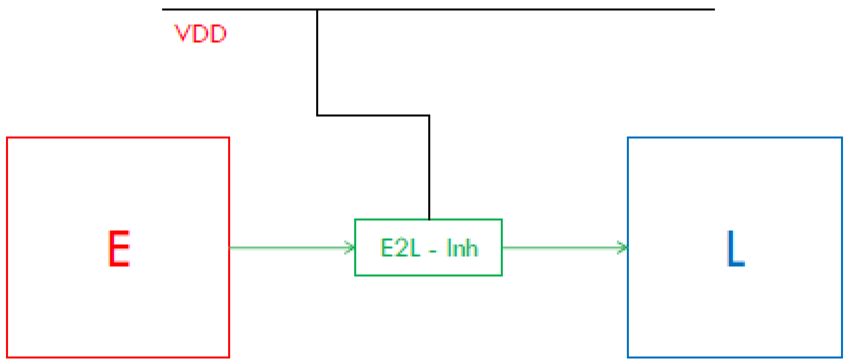


Figure 4: Illustration of a supply inherited E2L CM[4]

A snippet of the inherited CM code is shown below where \vdd! and \vss! values from cds_globals module will be used for A2D and D2A conversion. Here, the Verilog-AMS contructs inh_conn_prop_name and inh_conn_def_value are used to specify the global signals that are inherited by the local electrical signals \vdd! and \vss!. Thus the electrical port of the block that is connected to the local signals in the CM inherit the cds_globals module's supplies and drive the logic side where they are connected to.

```
connectmodule E2L_inhconn (Ain, Dout);
  input Ain; electrical Ain;                    // electrical input
  output Dout; \logic Dout;                     // logic output

// Inherited vdd! and vss!
  electrical     (* integer inh_conn_prop_name="vdd";
        integer inh_conn_def_value="cds_globals.\\vdd! "; *) \vdd! ;
  electrical     (* integer inh_conn_prop_name="vss";
        integer inh_conn_def_value="cds_globals.\\vss! "; *) \vss! ;
```

Yet identifying and associating each interface with a right voltage or power domain is not a trivial and mean task, and requires a manually intensive, iterative, and error-prone process. The flow requires this information to be generated at the SoC-level integration stage and handled at the verification stage, which is a sub-optimal method.

b1. Through actual physical electrical supply connectivity
The supply sensitivity shown in Figure 5 is a specification to each IP, module, or funtional unit, and all necessary information is already available at the IP design stage. Hence handling them and providing necessary supply-sensitivity information as a construct at each IP deliverable (a functional model) is the most optimal way of handling this problem. But it requires a lot of manual effort, at the initial stages of the design, once per IP design including all the standard cells, I/O cells, and analog functional modules. Though this is not a recurring effort, it is still not a trivial and easy task.

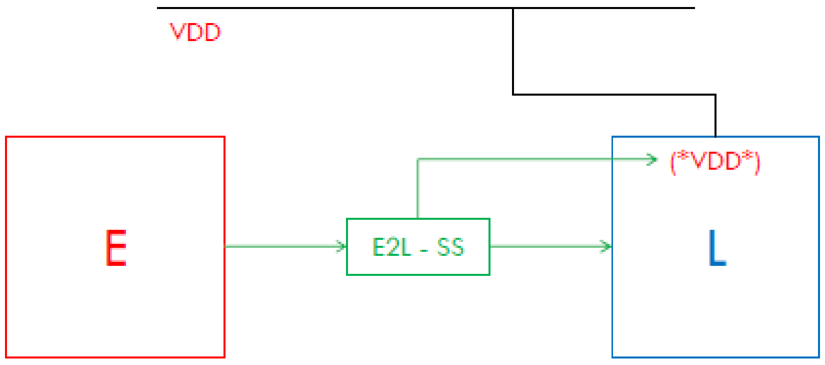


Figure 5: Illustration of a physically supply-sensitive E2L CM[4]

A fully supply-sensitive CM is illustrated below. As can be seen, the only difference to the inherited CM is the supply-sensitivity information as highlighted. It also requires each logical design element in the design to have the same supply-sensitivity information but assigned to its local supply terminal or net, an example is provided below.

```
input VCC; electrical VCC;
input VDD; electrical VDD;
ground VSS; electrical VSS;

input (* integer supplySensitivity = "VCC" ; integer groundSensitivity = "VSS" ; *)  a_3p0v;
input (* integer supplySensitivity = "VDD" ; integer groundSensitivity = "VSS" ; *)  a_2p5v;
```

The application of a supply-sensitive CM is as below. Here, we use a module named ‘cds_globals’ whose \vdd! and \vss! values are used for referencing for electrical to logical conversion.

```
connectmodule E2L_2_ss (Ain, Dout);
  input Ain; electrical Ain;              // electrical input
  output Dout; \logic Dout;                // logic output

// Supply Sensitivity attributes
  electrical (* integer supplySensitivity = "cds_globals.\\vdd! " ; *) \vdd! ;
  electrical (* integer groundSensitivity = "cds_globals.\\vss! " ; *) \vss! ;
```

A block digram of all the CMs discussed and illustrated so far is shown in Figure 6. The diagram shows all the possible combinations of these CMs as used in a mixed-signal SoC. But the challenges of setting up and using these various flavors of CMs always exist and we need a simplistic and comprehensive solution to overcome all those challenges.

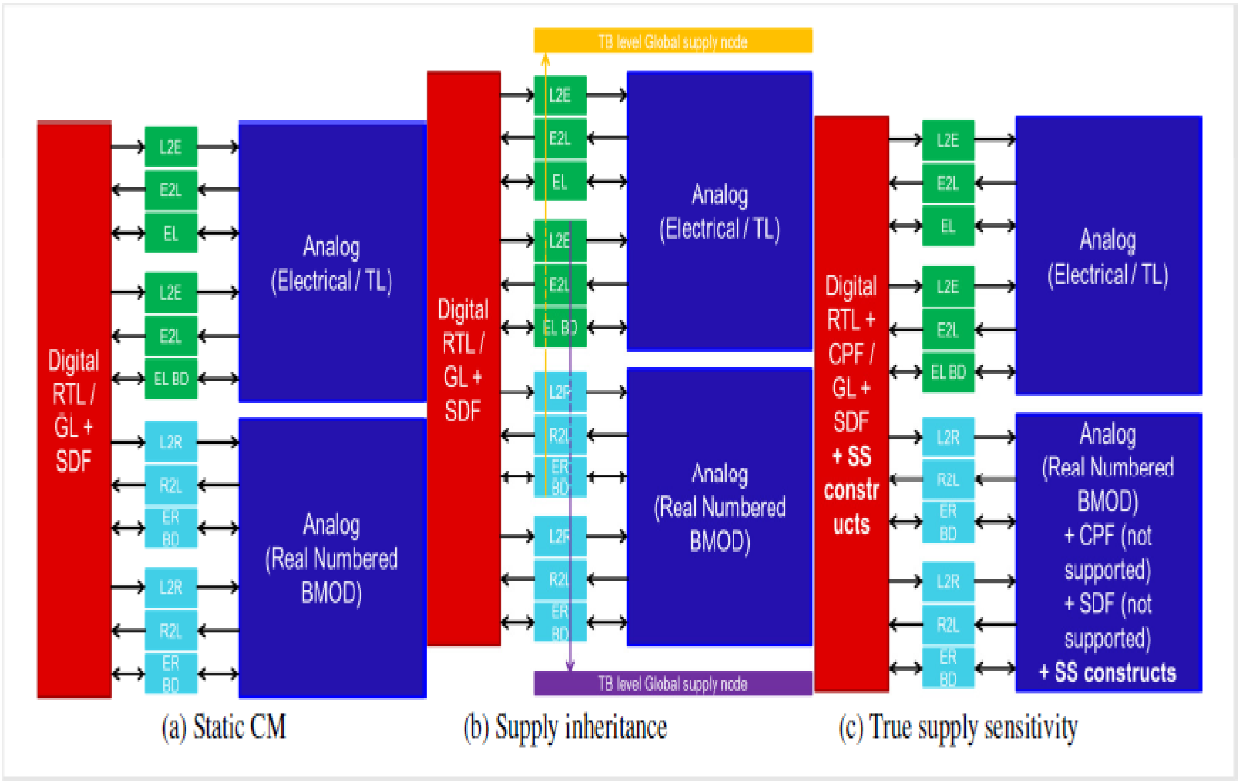

Digital RTL / GL + SDF
L2E
E2L
EL
EL BD
L2R
R2L
ER BD
Analog (Electrical / TL)
Analog (Real Numbered BMOD)
TB level Global supply node
Digital RTL / GL + SDF
Digital RTL + CPF / GL + SDF + SS constructs
Analog (Real Numbered BMOD) + CPF (not supported) + SDF (not supported) + SS constructs
(a) Static CM
(b) Supply inheritance
(c) True supply sensitivity

Figure 6: Block diagram of various flavors of CMs in an SoC environment[4]

III. INTRODUCTION TO THE VERILOG-AMS SYSTEM FUNCTIONS

Before we delve into how DVS CMs take care of all the requirements discussed thus far for a multi-power/multi-voltage design, let us first take a look at the two new system functions introduced by Accellera in Verilog-AMS LRM version 2.4.0[1]. The system functions are **$analog_node_alias and $analog_port_alias** (section 9.20 Analog node alias system functions)[1] and a Cadence proprietary system function named **$real_net_alias** as shown in Figure 7.

VERILOG-AMS LRM v2.4.0, section 9.20 - Analog node alias system functions

```
analog_node_alias_system_function ::=
        $analog_node_alias ( analog_net_reference , hierarchical_reference_string )
      | $analog_port_alias ( analog_net_reference , hierarchical_reference_string )
```

Figure 7: Two new system functions in VAMS LRM for hierarchical electrical node aliasing[1]

Both system functions take two arguments. The *analog_net_reference* shall be either a scalar or vector continuous node declared in the module containing the system function call. If the *analog_net_reference* is a vector node, it shall reference the full vector node, it shall be an error for it to be a bit select or part select of a vector node. It shall be an error for the *analog_net_reference* to be a port or to be involved in port connections. The *hierarchical_reference_string* shall be a constant string value (string literal or string parameter) containing a hierarchical reference to a continuous node. The *hierarchical_reference_string* shall follow the resolution rules for hierarchical references as described in Section 6.7 in the LRM[1].

It shall be an error for the *hierarchical_reference_string* to reference a node that is used as an *analog_net_reference* in another $analog_node_alias() or $analog_port_alias() system function call. It shall be an error for the $analog_node_alias() and $analog_port_alias() system functions to be used outside the analog initial block.

An example scenario using the two system functions is below.

```
module top;
electrical a, b, c, gnd;
ground gnd;
resistor #(.r(1k)) res1 (.p(a), .n(gnd));
checker #(.n1_str("top.a")) ch1 ();
analog V(a,gnd) <+ 5;
endmodule

module checker;
electrical n1, n2;
parameter string n1_str =” "; //empty string
integer status;
real iprobe, vprobe;
        analog initial begin
        // Node n1 will be aliased to top.gnd
        status = $analog_node_alias(n1, "top.gnd");

        // Even though n1 was assigned a valid alias to top.gnd, this one to top.a will take precedence.
        status = $analog_node_alias(n1, n1_str);

        // Here n2 is aliased to the port p in instance res1.
        status = $analog_port_alias(n2, "top.res1.p");
        end

        analog begin
        // Since n2 is aliased to the port p of instance top.res1, we are allowed to probe the port current. In this case,
        // the probe will return a value of 5mA.
        iprobe = I(<n2>);

        // Since n1 is aliased to the node top.a, we will be probing the potential of that node. In this case, the probe
        // will return a value of 5V.
        vprobe = V(n1);
        end

endmodule
```

Thus the system functions $analog_node_alias and $analog_port_alias create a local alias for a string parameter from the top level. This string parameter support in Verilog-AMS has given way to the DVS CMs brought in by Cadence as a new EDA methodology for power-managed mixed-signal design verification.

IV. SYSTEM FUNCTIONS FORMING THE DVS CMs

Here comes the actual usage of the system functions in the context of AMS verification where they are used in the CMs, thereby giving rise to the all new DVS CMs. A ring inverter chain test case is used to illustrate the use and application of DVS CMs.

```
// Digital Inverter
module dig_inv(in, out);
input in; output out;
reg out;
ddiscrete in, out;
always out = #10 ~in;
endmodule

// Analog Inverter
module analog_inv(in, out, vdd);
input in; output out;
electrical in, out, vdd;
real outval;
analog begin
        if (V(in) > V(vdd)/2.0)
        outval = 0;
        else
        outval = V(vdd);
        V(out) <+ transition(outval);
end
endmodule

// Global supply voltage
module global_supply;
electrical vdd;
analog V(vdd) <+ 5.0;
endmodule

// Ring Inverter made of digital and analog inverters
module ring;
dig_inv d1 (n1, n2);
dig_inv d2 (n2, n3);
analog_inv a3 (n3, n1, global_supply.vdd);
endmodule

// Electrical to logic CM
connectmodule elect_to_logic(el, cm);
input el; output cm;
reg cm;
electrical el, vdd; ddiscrete cm;
parameter string vddname = " "; // Empty string, Set via the CR
analog initial begin
        if($analog_node_alias(vdd, vddname) == 0)
        $error("Unable to resolve power supply: %s", vddname);
end
always @(cross(V(el) - V(vdd)/2.0, 1))
        cm = 1;
always @(cross(V(el) - V(vdd)/2.0, -1))
        cm = 0;
endmodule

// Logic to electrical CM
connectmodule logic_to_elect(cm, el);
input cm; output el;
ddiscrete cm;
```

```
electrical el, vdd;
parameter string vddname = " "; // Empty string, set via the CR
        analog initial begin
        if($analog_node_alias(vdd, vddname) == 0)
        $error("Unable to resolve power supply: %s", vddname);
        end
analog V(el) <+ V(vdd) * transition((cm == 1) ? 1 : 0);
endmodule

// Connect rules file that instantiates the CMs
connectrules mixedsignal;
connect elect_to_logic #(.vddname("global_supply.vdd"));
connect logic_to_elect #(.vddname("global_supply.vdd"));
endconnectrules
```

When there are multiple supplies in the design, distinct disciplines must be specified for each digital net that is associated with a given supply, which may be done by explicitly specifying the discipline of the digital nets or by using remote disciplines. Supply sensitivity in multi-supply designs is managed in the same way as above but with the specific supply hierarchical reference provided on each connect rule, as the following example shows.

```
// Global supply voltages for a multi-supply design
module global_supply;
electrical vdd_1v2;
electrical vdd_1v8;
analog begin
        V(vdd_1v2) <+ 1.2;
        V(vdd_1v8) <+ 1.8;
end
endmodule

// Connect rules file that instantiates the CMs for a multi-supply design
connectrules mixedsignal;
        connect elect_to_logic #(.vddname("global_supply.vdd_1v2"))
        input electrical, output ddiscrete_1v2;

        connect logic_to_elect #(.vddname("global_supply.vdd_1v2"))
        input ddiscrete_1v2, output electrical;

        connect elect_to_logic #(.vddname("global_supply.vdd_1v8"))
        input electrical, output ddiscrete_1v8;

        connect logic_to_elect #(.vddname("global_supply.vdd_1v8"))
        input ddiscrete_1v8, output electrical;
endconnectrules

// Real to Logic CM using $real_net_alias
connectmodule R2L_dynsup(Rin, Lout);
  output Lout;
  input  Rin; wreal Rin, vdd;
     initial begin
            if($real_net_alias(vdd, vddname) == 0)
            $error("Unable to resolve power supply: %s", vddname);
     end
   always begin
```

```
                if(Rin >= vth)  R_conv = 1'b1;
            else if (Rin <= vtl) R_conv = 1'b0;
            else if(Rin === `wrealZState) R_conv = 1'bz;
            else Xin = 1;
        end
        @(Rin, supOK, vth, vtl);
    assign Lout = R_conv;
endmodule
```

V. USE MODEL OF DVS CMs

In DVS using the Incisive AMS Use Model[2] flow, the user uses the same amsd {} block as seen before for the static CMs. But instead of using a static supply voltage, one must use string parameters to specify the global supply and ground signals to be referenced. An example of the DVS use model is shown below.

```
amsd{
ie connrules=CR_dynsup_full_fast vddnet="testbench.ams_avdd" vssnet="testbench.ams_VSS" cell="INV_F1_3P3V "
ie connrules=CR_dynsup_full_fast vddnet="testbench.ams_vddc" vssnet="testbench.ams_VSS" cell="INV_F1_HVT "
}
```

The parameter connrules specifies the connect rule name to be used from the tool installation, so when we set it to CR_dynsup_full_fast, the tool picks up DVS connect rules and CMs. The parameters vddnet and vssnet specify the global supply and ground signals to be used for the specified cells according to the power domains they fall under. The tool then uses the values of the global signals specified using the vddnet/vssnet parameters for the given cells. As seen, this method is the simplest to use and very efficient in terms of turnaround time for setting up the verification flow.

VI. HETEROGENEITY IN DVS CMs

DVS CMs extend support for the heterogeneity feature available in inherited and supply-sensitive CMs, where a real number to logic conversion can be based on an electrical supply as reference.

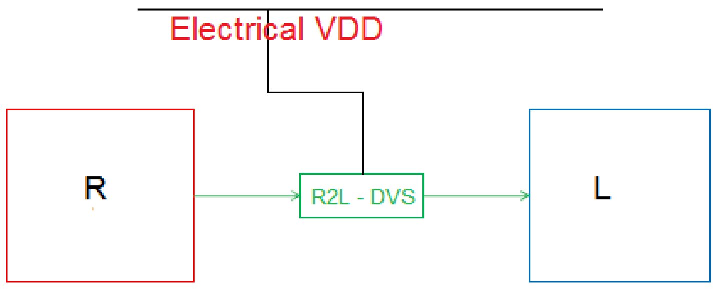


Figure 8: Heterogenous DVS R2L CM using electrical supply[4]

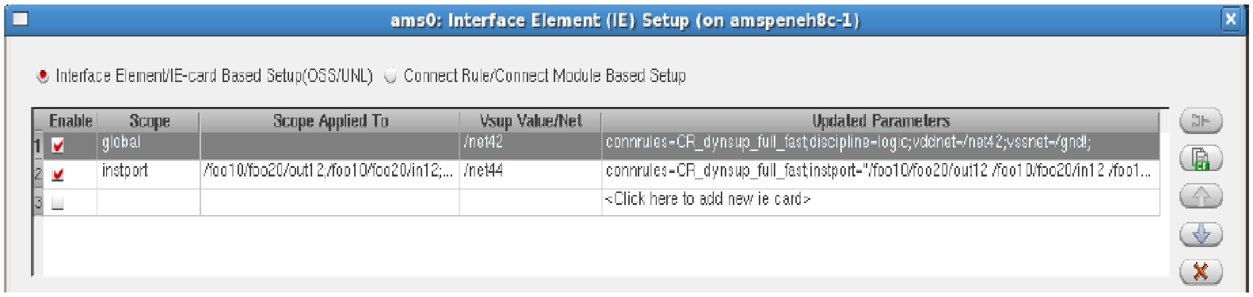
The tool inserts an R2L CM for converting a real number to logic signal (or vice-versa), which ideally will work based on a wreal supply. But with the support available for heterogeneity, an R2L CM will reference an electrical supply and does the conversion accordingly. This feature is applicable in all the three flavors of CMs, viz, inherited CMs, supply-sensitive CMs, and DVS CMs.

VII. GUI USE MODEL FOR DVS CMs

DVS CMs can also be used by AMS users who use the Cadence Virtuoso® AMS Use Model[3] where the AMS setup, simulation, and verification is done using the Virtuoso Analog Design Environment (ADE)[3] GUI. Using the

Virtuoso ADE, one can set up DVS CMs and configure the supplies and ground connections for different blocks/instances as done using the amsd {} block in the Incisive AMS Use Model flow.

A screenshot of the Virtuoso AMS Use Model for setting up DVS CMs is shown in Figure 9. Here we see the user has to select the IE-card-based setup radio button in the IE setup form and set the Scopes, Scope Applied To, Vsup Value/Net, etc. As seen in the figure, /net42 and /net44 are selected for the field Vsup Value/Net, thereby setting up DVS CMs in the GUI.



[● Interface Element/IE-card Based Setup(OSS/UNL)] ○ Connect Rule/Connect Module Based Setup

	Enable	Scope	Scope Applied To	Vsup Value/Net	Updated Parameters
1	✓	global		/net42	connrules=CR_dynsup_full_fast;discipline=logic;vddnet=/net42;vssnet=/gnd!;
2	✓	instport	/foo10/foo20/out12;/foo10/foo20/in12;...	/net44	connrules=CR_dynsup_full_fast;instport="/foo10/foo20/out12 /foo10/foo20/in12 /foo1...
3	☐				<Click here to add new ie card>

Figure 9: DVS CM using Virtuoso ADE IE GUI

VIII. LIMITATIONS OF DVS CMs

Practically, use of DVS CMs might be limited based on the design architecture and number of power domains, and one has to carefully consider the following points before implementing DVS CMs in his verification environment.

- If the set up is from scratch, there is some effort needed to set up various IE cards for a power-managed design, if there are tens of power domains in the design. However this is a rare scenario, as today's SoCs usually consist of three or four power domains.
- As we use the IE card flow, insertion of CMs can occur anywhere in the design that is decided by the tool's discipline resolution process. While there is a chance that a DVS CM will be placed in an undesired location due to this process, it can be taken care of.
- While supply-sensitive CMs make use of the supply information provided by the IP owner, DVS CMs make use of supply information provided by the verification engineer. So, the verification engineer must make sure he uses the right supply voltage using string parameters corresponding to the respective power domains.

IX. CONCLUSION

DVS CMs mainly targets verification engineers who are predominantly used to the Incisive AMS Use Model[2] for digital-on-top designs, though this use model also covers analog-on-top scenarios. Using the Cadence amsd{} block and IE cards[2], one can easily specify the constructs for string parameters that specify the hierarchical supply and ground nets to be used as reference for logic-to-electrical conversion, logic-to-real conversion, and vice-versa. In this way, one can easily migrate from supply inherited CMs to DVS CMs, using automation scripts.

For multi-power domain designs where the setup is needed from scratch, it is a one-time effort of classifying the cells and instances under their respective power domains, and the configuration can then be re-used across test cases for coverage.

Another application is when a design with static CMs setup using amsd{} block becomes a power-managed design. The static voltage thresholds can simply be converted to string parameters that represent the reference supply-ground nets for conversion, thereby implementing DVS CMs.

X. ACKNOWLEDGEMENTS

We are thankful to the following persons at Cadence Design Systems, Inc.:

- Vishwajeet V Betai and Rishabh Gupta for their constant motivation and support
- Qiaofeng Zhu and Zhong Fan in managing and tracking implementation of the activity covered by this paper
- Andre Baguenier for the continuous technical guidance and support
- Vijay Prakash M and Harish Varadarajan for their editorial review of the paper

References:

[1] Accellera Systems Initiative Inc, “Verilog-AMS Language Reference Manual”, version 2.4.0, May 30, 2014

[2] Cadence Design Systems, Inc.,”Workhop for AMSD Incisive Use Model”, Revision 1.8, AMS Product Engineering, September 2010

[3] Cadence Design Systems, Inc.,”Workhop for AMSD Virtuoso Use Model”, AMS Product Engineering, April 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] Cadence Verilog-AMS Language Reference, Product Version 12.1, June 2012