Mixed Signal Verification of UPF based designs A Practical Example

Andrew Milne, Damian Roberts
Synopsys NE 100 Brook Drive
Reading RG2 6UJ
UK

Abstract- Due, in a large part, to the huge demand for mobile devices, reducing the power consumption of microchips is one of the most important goals in electronic design today. In order to reduce the amount of power demanded by these chips, engineers are architecting increasingly complex power supply schemes.
These systems now have many power modes that enable sections of the design to be either turned off completely or driven to different supply voltages. These separate power islands (known as power domains) each need their own power switches, as well as the supporting structures e.g. level shifters, retention and isolation circuitry, to ensure that corrupt data does not cause a functional failure of the system. Implementing and then verifying this power supply logic is now a significant consideration in the design of modern chips. Unified Power Format (UPF) is a standard format (IEEE-1801) that allows the power supply strategy of a system on chip to be defined. This UPF file is then used within both the implementation and verifications flows, to ensure that this power intent is followed. The majority of electronic systems contain some analog circuitry e.g. on chip voltage regulators that provide the supplies for the rest of the chip. It is therefore important to verify the functionality of these analog macros in the context of the overall system. This paper illustrates via a practical example, how the power intent for a mixed signal UPF design containing SPICE descriptions of the voltage regulators and analog macros can be verified using a UPF driven simulation approach. The example walks though the UPF creation, simulation and debug for a sigma-delta ADC. Recommendations are made as to how the UPF for the analog blocks can be structured to ensure that the same UPF can be used for implementation and verification of the System. The example also illustrates some of the potential design issues both in the analog design and in the overall system, that can be identified using this methodology.

INTRODUCTION

A Low Power design using UPF can be verified as a whole, including mixed signal and SPICE blocks. The primary means of achieving this is via a co-simulation of digital and analog simulators, allowing the verification of design blocks described using HDLs at the same time as blocks described as SPICE, VerilogA, or Verilog Real Number models.

Use of co-simulation to verify the mixed signal portions of a SOC has been a common technique for a number of years. However, the use of UPF to describe the power intent of the design, does add some additional challenges.

Typically, digital designs contain no explicit power information within the netlist. Indeed, it is this information that UPF adds. SPICE netlists typically do have, and require, this information. Therefore, it is necessary that the SPICE supply nets be driven from digital/UPF side of the simulation. In addition, any changes to the current 'state' of the UPF supply nets, such as voltage changes, power down etc., should also be passed onto the SPICE domain.

Going further, it is very common for SOCs to have on chip Power Supply Units (PSUs) that produce the various multi voltage supplies that are required. If the PSU is modelled using SPICE, then the output voltage of the PSU also needs to drive the digital domain UPF supply nets.

For this paper, VCS-AMS™ was used, i.e. VCS™ as the digital simulator and CustomSim™ was used as the analog engine. The NLP feature of VCS was used for the UPF functionality. Throughout, a conventional VCS UPF simulation flow has been used, but with the addition of the VCS –ad switch to enable CustomSim co-simulation.

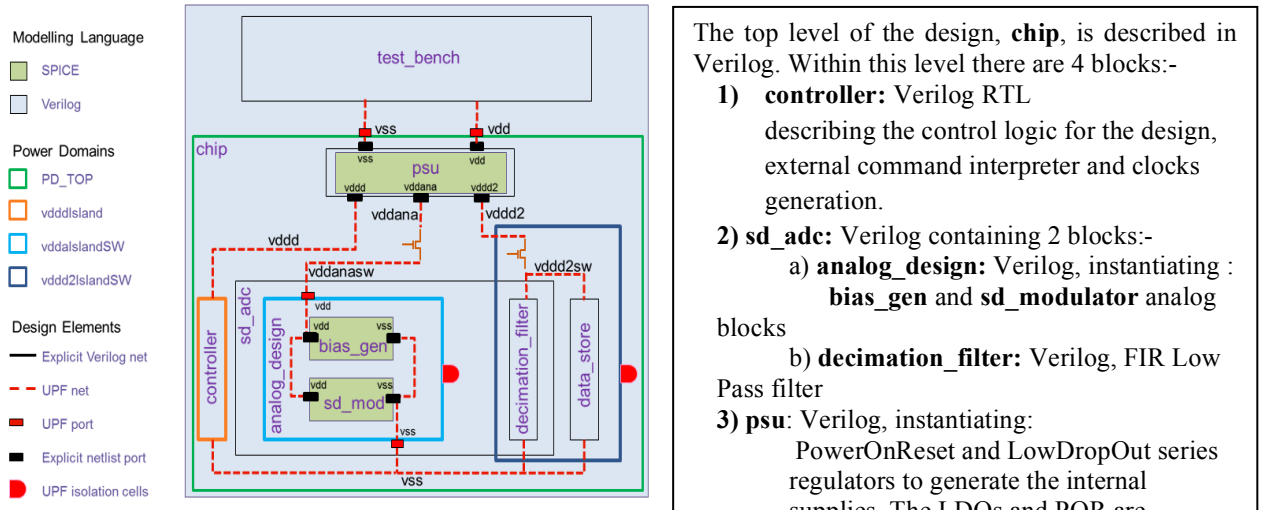
The standard co-simulation flow manages the partitioning of design blocks to the relevant simulator engine, insertion of interface elements between the digital and analog blocks, as well as analog simulator accuracy settings. The tool will, for example, automatically insert interface elements to convert a real number generated in a Verilog model, or by the UPF NLP data, into an electrical voltage for use in the SPICE block.

While this paper discusses the verification of UPF based mixed signal designs, the intent is, that exactly the same UPF can be used for implementation. For implementation, the various mixed-signal blocks would typically be represented as hard macros, described as Liberty .lib models.

I. EXAMPLE DESIGN

The paper describes the setup and use of this flow on an example design, in this case a Sigma Delta ADC with an on-chip Power Supply and control logic that can respond to external commands. The majority of the design is Verilog, with the power intent described in UPF. Some of the design blocks are analog in nature. These could be modelled as Verilog Real number, VerilogA or SPICE. The paper has taken the approach of using the SPICE description for these blocks. This would most likely have the lowest simulation performance, but perhaps the most utility from a verification perspective. It allows the functionality of the true transistor level implementation to be verified within the context of the full chip, including power supply controls e.g. does the analog block behave as expected as its supplies are turned off/on? In addition, from an example perspective, this is the most complicated scenario and so is perhaps the most useful to the end user. Replacing the SPICE with Verilog Real Number or VerilogA models should present no additional challenges.

A block diagram of the example design, with power domains, is presented in Figure 1 below.



The top level of the design, **chip**, is described in Verilog. Within this level there are 4 blocks:-

1) **controller:** Verilog RTL describing the control logic for the design, external command interpreter and clocks generation.
2) **sd_adc:** Verilog containing 2 blocks:-
   a) **analog_design:** Verilog, instantiating : **bias_gen** and **sd_modulator** analog blocks
   b) **decimation_filter:** Verilog, FIR Low Pass filter
3) **psu**: Verilog, instantiating: PowerOnReset and LowDropOut series regulators to generate the internal supplies. The LDOs and POR are represented as SPICE.
4) **Data store**: Verilog model of RAM for storing converted values.

Figure 1. Block diagram of Sigma Delta ADC

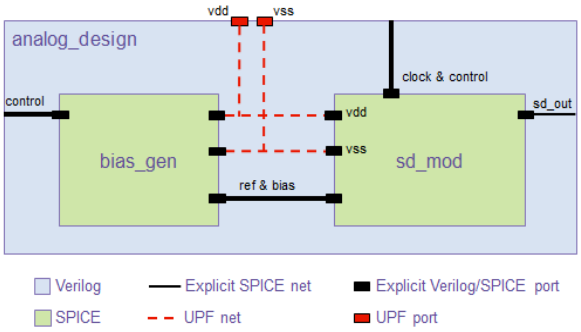
The design has a regulated 'always on' supply to the control logic, the **vdddIsland** domain. In addition, there should be separate power domains for the analog_design and for the decimation_filter/data_store, **vddaIslandSW** and **vddd2IslandSW** respectively. It was also required that both of these power domains could be controlled separately.

II. UPF FOR THE EXAMPLE DESIGN

The steps used to create the UPF description of this power intent, are detailed below. For this example UPF 1.0 has been used. However, UPF 2.0 syntax is equally applicable.

A. Create sub-UPF for analog_design

The analog top level block is written in Verilog and contains instantiations of the two SPICE blocks.
This structure favors the use of hierarchical UPF i.e. a separate sub-UPF file for the analog block. This sub-UPF file can then loaded from the top level UPF file (described in sections B to L). The sub-UPF contains all of the power connectivity for the analog block and so the Verilog netlist should not contain any power ports or nets. The sub-UPF for analog_design is shown below. Firstly, UPF power ports and power nets called **vdd** and **vss** are created. These power nets are then connected to the power ports of the two lower level SPICE blocks.
Figure 2 shows the hierarchy of analog_design and the power net connections that result from the sub-UPF.




```
create_supply_port vdd
create_supply_net vdd
connect_supply_net vdd -ports {vdd I0/vdd I1/vdd}
create_supply_port vss
create_supply_net vss
connect_supply_net vss -ports {vss I0/vss I1/vss}
```

Figure 2: Block diagram of analog_design

This hierarchical UPF approach has some advantages. In particular, it is very flexible from a design implementation perspective. If the whole analog_design block is implemented using an analog design flow (e.g. Custom Designer™), then it will represented as a black box for place and route (PnR) and only the top level UPF file will be required to connect up the power structures at the SOC level. If however the Verilog for analog_design is passed to the PnR tool for implementation and the two lower level blocks are treated as black boxes, then the sub-UPF can be included so that the power supplies within analog_design are correctly handled by the PnR tool.

B. Create power domain definitions in top level UPF.

The power supply behavior of each power domain can be defined and controlled separately. In this case we have four power domains.

1) A top level domain (always required) called **PD_TOP**
2) **vdddIsland**, containing the controller.
3) **Vddd2IslandSW** containing the decimation filer and the memory (dataStore).
4) **vddaIslandSW** containing the analog block.

```
create_power_domain PD_TOP –elements {chip1}
create_power_domain vdddIsland -elements {chip1/control1}
create_power_domain vddd2IslandSW -elements {chip1/store1 chip1/sd_adc1/decimation_filter_1x16bit_1}
create_power_domain vddaIslandSW -elements {chip1/sd_adc1/analog_design_1}
```

C. Define the supply nets that will be used in the design

For this example, in order to improve the clarity, all of the power domains use the net vss (defined as part of the top level domain PD_TOP) as the ground net. The "-reuse" switch indicates that this top level net, which has already been defined, will be used. For the positive supplies, the following power nets are defined:-

1) **vdd** in domain **PD_TOP**. This is the external supply to the system.
2) **vddd, vddana and vddd2** in domain **PD_TOP**. These are internal supplies created by the psu block.
3) **vddanasw** in the domain **PD_TOP**. The net **vddanasw** will be connected via a power switch to **vddana**
4) **vddd2sw** in the **vddd2IslandSW** domain**.** The net **vddd2sw** will be connected via a power switch to **vddd2**

```
create_supply_net vdd -domain PD_TOP
create_supply_net vddanasw -domain PD_TOP
create_supply_net vddd2 -domain PD_TOP
create_supply_net vss -domain PD_TOP
create_supply_net vss -domain vddd2IslandSW –reuse
create_supply_net vddana -domain PD_TOP
create_supply_net vddd -domain PD_TOP
create_supply_net vddd2sw -domain vddd2IslandSW
create_supply_net vss -domain vdddIsland -reuse
```

D. Define the primary power nets that will drive each power domain.

Each power domain needs to have primary supply nets defined i.e. these will be the main nets that supply the blocks within the domain.

```
set_domain_supply_net PD_TOP -primary_power_net vdd -primary_ground_net vss
set_domain_supply_net vdddIsland -primary_power_net vddd -primary_ground_net vss
set_domain_supply_net vddd2IslandSW -primary_power_net vddd2sw -primary_ground_net vss
set_domain_supply_net vddaIslandSW -primary_power_net vddanasw -primary_ground_net vss
```

E. Connect the top level supply nets to power ports.

Where no ports exist in the netlist, virtual ports need to be defined e.g. here the ports **vdd** and **vss** are created in the **PD_TOP** power domain.

```
create_supply_port vdd -domain PD_TOP -direction in
connect_supply_net vdd -ports vdd
create_supply_port vss -domain PD_TOP -direction in
connect_supply_net vss -ports vss
```

F. Connect the power supply unit (PSU) ports to the relevant power nets.

In this design the internal supplies are generated by the PSU. The PSU itself is supplied by the net **vdd.** It then generates 3 supplies, **vddana**, **vddd** and **vddd2**. These 3 output nets, or switched versions of these, will supply the rest of the chip.

```
###inputs of PSU                                  ###outputs of PSU
connect_supply_net vdd -ports chip1/psu1/vdd     connect_supply_net vddana -ports chip1/psu1/vdda
connect_supply_net vss -ports chip1/psu1/vss     connect_supply_net vddd -ports chip1/psu1/vddd
                                                  connect_supply_net vddd2 -ports chip1/psu1/vddd2
```

G. Create power switches.

In this design, both the domain containing the SPICE blocks **vddaIslandSW** as well the digital domain **vddd2IslandSW** can be switched off i.e. there will be some power switches controlling the supplies to these blocks. In the UPF below 2 switches are defined:-

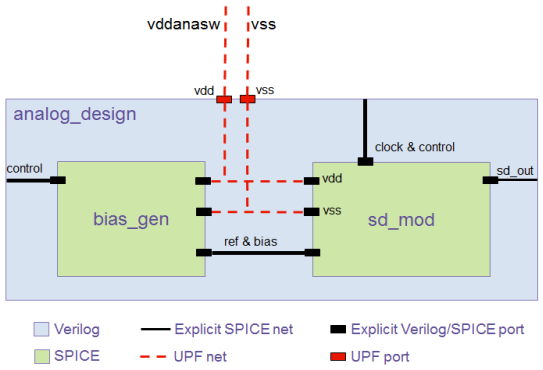
1) The **ana** switch is created in the top level domain **PD_TOP.** The input to the switch is connected to the power net called **vddana.** The output of the switch is the **vddanasw** net. The control for this switch is called **ctrlAn** and is driven by a control net with the following hierarchical name **chip1/enable_vdda.** When **ctrlAn** is logic high the switch, and therefore the **vddanasw** supply is "on". When it is logic low, this supply is "off"
2) The **dig** switch is created in the **vddd2IslandSW** domain**.** The input to the switch is connected to the power net called **vddd2.** The output of the switch is the **vddd2sw** net. The control for this switch is called **ctrlDig** and is driven by a control net **chip1/enable_vddd2.**

```
create_power_switch ana -domain PD_TOP -output_supply_port {anPout vddanasw} -input_supply_port \
{anPin vddana}  -control_port {ctrlAn chip1/enable_vdda} -on_state {active anPin {ctrlAn}} \
-off_state {inactive {~ctrlAn}}
create_power_switch dig -domain vddd2IslandSW -output_supply_port {digPout vddd2sw} -input_supply_port \
{digPin vddd2} -control_port {ctrlDig chip1/enable_vddd2} -on_state {active digPin {ctrlDig}} \
-off_state {inactive {~ctrlDig}}
```

H. Connect the correct power nets to the SPICE power ports.

The supply connections for the SPICE blocks within analog_design are already defined in the sub-UPF described in section A. Therefore the top level UPF only needs to connect the UPF power nets **vss** and **vddanasw** to the power ports at the top level of the analog_design. The sub-UPF file is loaded using the load_upf command. The -scope switch is used to specify the hierarchical path to the block where the sub-UPF should be applied.

Figure 3, shows the power net connections that result from the combination of the top level UPF and the sub-UPF.




```
load_upf /global/gtseu1/cases1/analog_design.upf  \
–scope chip1/sd_adc1/analog_design_1

connect_supply_net vddanasw -ports
{chip1/sd_adc1/analog_design_1/vdd}

connect_supply_net vss -ports
{chip1/sd_adc1/analog_design_1/vss}
```

Figure 3. power net connections from top and sub-UPF

Note: The VCS-AMS™ flow supports the direct connection of UPF power nets to SPICE i.e. the voltage value on the UPF net will be automatically converted into an electrical voltage which will drive the SPICE supply ports.

I. Isolate the inputs to and outputs from the blocks contained in the switchable domains

In this design, when the switchable power domains are turned off, we want to make sure that the blocks inside these domains are isolated. This avoids corrupt data generated by these powered down blocks being fed into other parts of the system. Therefore, some isolation cells are defined in the UPF to avoid this. When the **vddaIslandSW** and **vddd2IslandSW** domains are powered down, the isolation cells should be enabled and clamp the outputs of the domain. The clamp value in the example UPF below is specified to be logic 0. This isolation strategy will be applied to all outputs of the **vddaIslandSW** and **vddd2IslandSW** domains. The isolation logic itself is powered by the **vddd** and **vss** power nets (always on). The controls for the isolation cells also have to be defined. The isolation enable for the **vddaIslandSW** domain is the **chip1/control1/vddaIso** net, which is active high. The isolation control for the **vddd2IslandSW** domain is the **chip1/control1/vddd2iso** net.

```
set_isolation iso1 -domain vddaIslandSW -isolation_power_net vddd \
 -isolation_ground_net vss -clamp_value 0 –applies_to outputs
set_isolation_control iso1 -domain vddaIslandSW \
-isolation_signal chip1/control1/vddaIso -isolation_sense high -location parent
set_isolation iso2 -domain vddd2IslandSW -isolation_power_net vddd -isolation_ground_net vss \
-clamp_value 0 –applies_to outputs
set_isolation_control iso2 -domain vddd2IslandSW -isolation_signal chip1/control1/vddd2Iso \
-isolation_sense high -location parent
```

J. Definition of level shifters.

It is possible to specify level shifter constraints in UPF using the set_level_shifter construct. However, by default, NLP does not simulate level shifters (or absence of level shifters). This is because level shifters are automatically inferred by the implementation tools, based on the definition of the power state table (PST). This reinforces the need to properly validate the Power State Tables (PST).

K. Add port state data.

The following lines add state information to the power ports. In the case below, if vdd is between 1.8v and 2.1v it will be in state "highV". If the supply is "off" it will be in state "OFF". The always on regulated logic supply, vddd will be in state "active" if its voltage is between 1.0 and 1.3v.The output port of the digital power switch (dig/digPout) will be in state "active" if its voltage is between 1.5 and 1.8v. The output port of the analog power switch (ana/anPout) will be in state "active" if its voltage is between 1.5 and 1.8v and in state "lowV" if it is between 1.3v and 1.49v.

These port state definitions can then be used to populate the power state tables.

```
add_port_state vdd          -state {highV  1.8 2.1} -state {OFF off}
add_port_state vss          -state {active 0.0}
add_port_state vddd         -state {active 1.0 1.3} -state {OFF off}
add_port_state dig/digPout  -state {active 1.5 1.8} -state {OFF off}
add_port_state ana/anPout   -state {active 1.5 1.8} -state {lowV 1.3 1.49} -state {OFF off}
```

L. Define the power state table (PST).

In this example, there are 4 legal operating states for the design.

1) If vdd is in state "highV" and vddd, vddd2sw, vddanasw and vss are in state "active", then the chip is in overall power state "**NormalOp**".
2) If vdd is in state "highV" and vddd2sw, vddanasw are in state "OFF" and vss and vddd are in state "active", then the chip is in overall power state "**StdBy**".
3) If vdd is in state "highV", vdd, vddd2sw and vss are in state active and vddanasw is in state "lowV", then the chip is in overall power state "**PowerSv**".
4) If vdd, vddd, vddd2sw, vddanasw are in state "OFF" and vss is in state "active", then the chip is in overall power state "**PowerDwn** ".

All other combinations will result in an "illegal" overall power state.

```
create_pst chiptop_pst          -supplies          {vdd    vddd   vddd2sw    vddanasw    vss}
add_pst_state NormalOp  -pst chiptop_pst –state    {highV  active  active    active      active}
add_pst_state StdBy -pst chiptop_pst     –state    {highV  active  OFF       OFF         active}
add_pst_state PowerSv -pst chiptop_pst   -state    {highV  active  active    lowV        active}
add_pst_state PowerDwn  -pst chiptop_pst -state    {OFF    OFF     OFF       OFF          active}
```

III. MODELLING THE POWER SUPPLY UNIT (PSU).

In this design, the supplies for the majority of the chip are supplied by an on-chip PSU. The PSU controls the voltage values for the supplies to the other block in the system and controls, for example, the power on reset as the external supply to the chip powers up.

The hierarchy of the PSU block is shown in Figure 4, below.

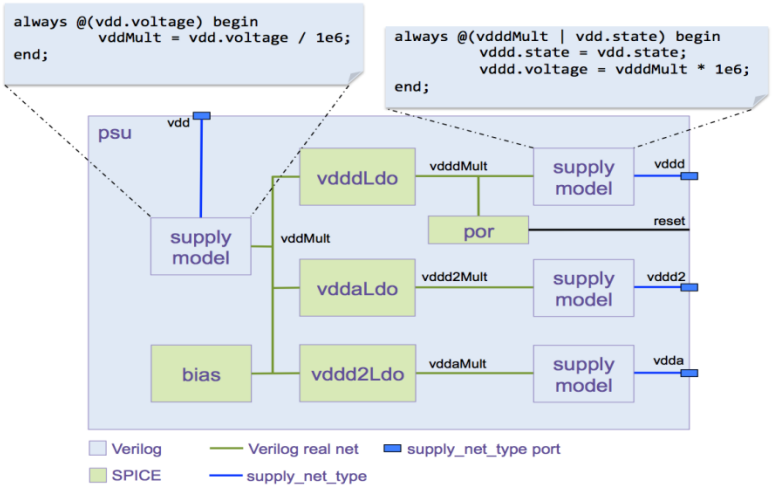


Figure 4. block diagram of PSU

In order for the model to correctly drive the UPF power nets, the outputs must use the System Verilog **supply_net_type**. This nettype is defined in the IEEE1801 UPF standard and contains both **voltage** and **state** properties.

Therefore in our PSU model we have the following definitions for the power ports that drive or are driven by the UPF.

```
module psu (clock, vddd, vddd2, vdda, reset, reset2, vddd2_control, vdda_control, enable_vddd2, vdd);
input supply_net_type vdd, vss;
output supply_net_type vdda, vddd, vddd2;
```

One of the aims of the PSU model was for the outputs of the PSU to follow the state of the input supply to the PSU, net vdd. i.e. if the external supply to the system is “OFF” then the PSU outputs should be “OFF” as well.

For this reason the state of these outputs is set to the same state as the vdd net, (see Figure 4).

In addition the voltage of the output nets should be driven to the correct voltage value. This can be achieved by modelling the PSU as a real number model.

Another approach, and the one used in this example design, is to use the SPICE implementation of the PSU circuitry to drive these output voltages.

Note: the unit of voltage used for the supply_net_type is uv. e.g. to set a net of this type to 1.2v this in fact has to be driven to a value of 1200000. Therefore, the voltages into and out of the SPICE representation of the PSU need to be scaled back to Volts. This can be achieved using System Verilog as seen in Figure 4.

IV. SIMULATION RESULTS

The testbench contains a SPICE block and Verilog section. The SPICE part of the testbench drives the ADC with an analog input signal. The Verilog part of the testbench drives a set of control signals into the chip. These control signals are decoded by the controller block and initiate a powerup sequence as well as taking the system into and out of a power save mode.

For this simulation the vdd powernet is in an "ON" state and set to 2.0v. This is controlled in the testbench by using the supply_on/off functions. These functions are defined in the UPF LRM and are made available to the testbench by including the UPF System Verilog package.

```
initial begin
        supply_on("vdd", 2.0);
        supply_on("vss", 0.0);
end
```

The output in Figure 5 illustrates that, as expected, **vdd** is "FULL_ON" and driven to 2.0v for the duration of the simulation. Since the outputs from the PSU are all driven from the SPICE implementation, there is a short power up period before these powernets, **vddd**, **vddana** and **vddd2** reach their expected values.

Powernet **vddd** reaches 1.2v and stays at this value.

The voltage values of both the **vddana** and **vddd2** power nets are programmed by the on chip controller. The **vdda_control** bus controls the value of **vddana** and the **vddd2_control** bus controls the value of **vddd2**. At approximately 180us the values of these control buses change. This leads to the voltage on **vddana** changing from 1.60v to 1.47v and the voltage on **vddd2** changing from 1.60v to 1.09v. The supply into the analog_design block is a switched version of the **vddana** signal.

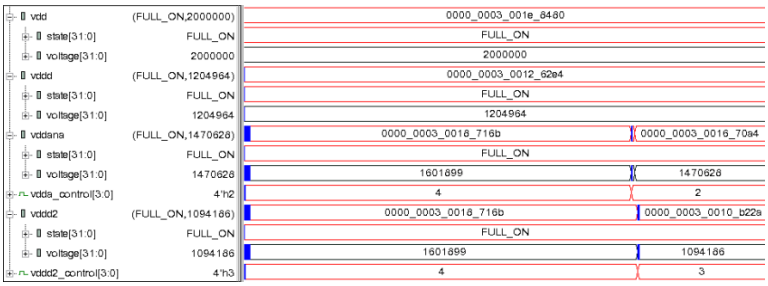


Figure 5. Top level power waveforms

The controls and resultant supply values for this vddanasw supply net are shown in Figure 6. The **ctrlAn** signal is the control for the power switch that enables the switched analog supply **vddanasw**. As the **ctrlAn** signal moves from a logic 1 to logic 0, this results in the **vddanasw** supply moving to the "OFF" state and a voltage of 0v.

The **vddaIso** signal is the enable for the isolation circuitry at the interface to/from the analog block. The timing of the control signals should be such that the **vddaIso** signal goes high some time before the analog supply is turned off and goes low again sometime after the analog supply is turned on again. This ensures that the analog block does not receive or supply corrupt data. The waveforms shown in Figure 6 indicate that this isolation circuitry is working as expected. The **sd_mod_out** signal is an output from the analog block. When the supply for the analog block turns

off, this signal moves to a z state. The isolated version of this signal is called **sd_mod_out_UPF_ISO**. It is this isolated signal that will be passed as an input to the decimation filter. As the isolation **vddaIso** signal goes high, **sd_mod_out_UPF_ISO** is clamped to logic 0, therefore avoiding a corrupted signal being passed into the filter.

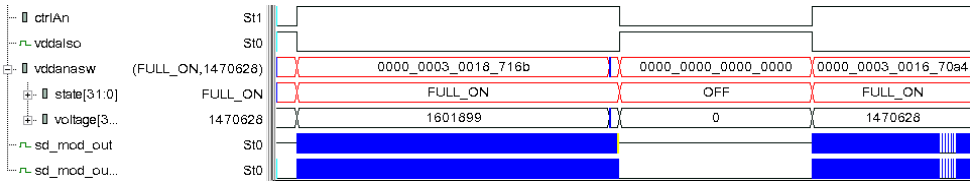


Figure 6. controls and waveforms for the analog supply

Figure 7 shows the analog input to the system (analog_in) as well as the 16bit output from the ADC (adc_out), represented as a stepped analog waveform .

The ADC output, as expected, is a good representation of the analog input.

However, since this output is isolated, it should not be corrupted to an x state as the supply (**vddd2sw)** turns off. Clearly however, this signal is driven to an x state for a short time after the isolation enable **vddd2Iso** goes low. This is something that needs to be investigated further.

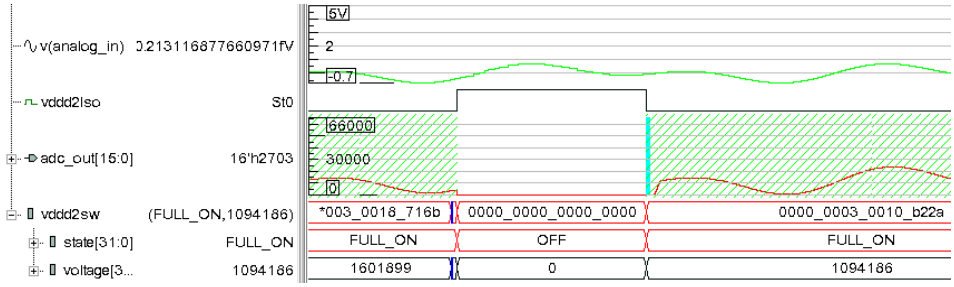


Figure 7. ADC output wavefoms

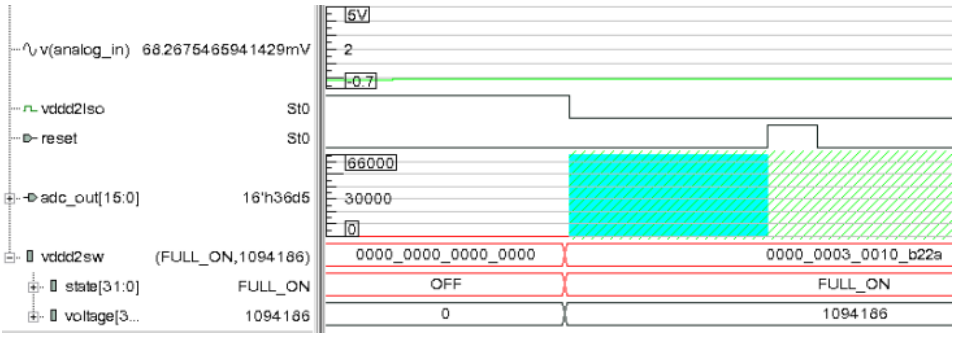
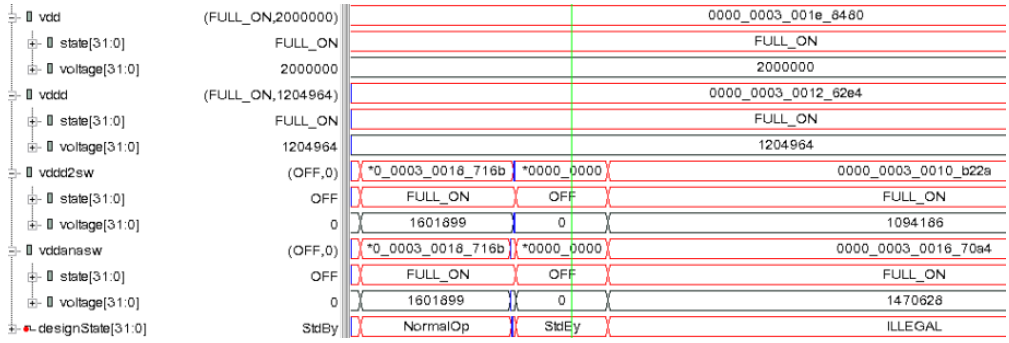


Figure 8. Isolation and reset control waveforms

Figure 8 is a zoomed in image of this period where the ADC outputs an x. The x state occurs from the time when the isolation is disabled (the falling edge of **vddd2iso**) until the decimation filter is reset ~ 2us later. This result indicates that there is a real design problem with the timing of these signals. The filter block should not come out of

isolation until it has been reset. Since this reset is generated by the PSU, which being simulated at SPICE level, this may be an indication that there is a problem with the transistor level implementation of this circuitry.

In addition to the standard waveform outputs discussed above, the power state information is also written to the waveform file. Figure 9 illustrates how this information is displayed in DVE. The voltage and state of each supply is displayed in the waveform view. In this example the system moves from "NormalOp" mode, into "StdBy" mode and then into an "illegal" state. The power state table view shows all of the legal power states. The active power state is highlighted in red. The fact that the design has gone into an "illegal" power state is something that would need to be investigated by the verification engineer. In this case the illegal state is due to the **vddd2sw** supply being too low. This could either be an issue with the control logic driving the PSU, or in fact, an issue with the design/implementation of the PSU itself. Of course this could also be an error in the PST definition. Since the SPICE netlist for the PSU was used for this simulation, this is a good check as to whether the transistor level implementation is functional.



	vdd	vddd2sw	vddanasw	vddd	vss
NormalOp	highV(1.95v)	active(1.65v)	active(1.65v)	active(1.15v)	active(0.00v)
StdBy	highV(1.95v)	OFF(off)	OFF(off)	active(1.15v)	active(0.00v)
PowerSv	highV(1.95v)	active(1.65v)	lowV(1.40v)	active(1.15v)	active(0.00v)
PowerDwn	OFF(off)	OFF(off)	OFF(off)	OFF(off)	active(0.00v)

Figure 9. Power state output

V. CONCLUSION

VCS-AMS™ enables a UPF based simulation of the power intent of a chip. Modern designs usually contain analog as well as digital blocks. As has been illustrated by the example above, the standard UPF flow can be used to control the behaviour of the supplies to both the digital and analog blocks. In addition, these analog blocks can be modelled using Verilog or SPICE. Indeed, if the supplies are generated by an on chip PSU, this methodology can help to verify that the control logic to this PSU is correct and that the transistor level implementation of the PSU produces the supply voltage values in line with UPF description (PST). This is clearly a critical aspect of Low Power design verification, since all implementation and static verification tools will consider PST as the golden specification.