Applications of Supply Tunneling in Unified Power Format 4.0 for Mixed Signal Design

Daniel Cross
Cadence Design Systems
+1-512-342-5560
12301 Research Blvd Ste 200
Austin, TX 78759
danielcr@cadence.com

***Abstract*- A new UPF HDL supply tunneling concept was introduced in IEEE 1801-2024. It allows power supply networks defined in IEEE 1801 to be represented in simulation by HDL-native nettypes, which can carry more power supply information than the UPF supply net. Applications and limitations of UPF HDL supply tunneling are presented, including situations in which UPF HDL supply tunneling is not supported. Design examples and figures with sample UPF code support the discussion. New features of IEEE 1801-2024 that help the user control tunneling behavior are shown.**

I. Introduction

Within the recent IEEE 1801-2024 standard [1] (henceforth termed Unified Power Format (UPF) 4.0 in this writing), the new concept of HDL supply *tunneling* was introduced. Tunneling of HDL (Hardware Description Language) supply nets allows for simulation of supply nets using analog representations even though those supply nets were declared as part of the power intent in a UPF file.

UPF HDL supply tunneling supports a design flow in which analog functional "islands" are embedded in a top level design netlist constructed in an HDL language such as SystemVerilog. This flow is to be contrasted with a flow in which all of the analog design is collected into a single partition, with its netlist governed generally by a schematic drawn in a graphic tool, including the supply network. With the analog island flow, the supply network can be defined in UPF as is typical in all-digital design flows.

The UPF 4.0 standard also introduces Value Conversion Methods (VCMs) that replace and expand the concept of Value Conversion Tables (VCTs). VCMs govern the interaction between Analog representations of supply nets in HDL and UPF-defined supply nets (which are usually represented by the UPF supply net type). Analog representations of nets – such as custom User Defined Nettypes (UDNs) in SystemVerilog – are generally more expressive than the UPF supply net type, which has only supply net state and voltage components. Thus, an analog supply connection through the UPF supply network would potentially lose valuable signal information. The concept of UPF HDL supply tunneling was introduced to eliminate this loss of information.

Other methods for preserving signal information generally have one or more of the following drawbacks:

- They are non-standard or bespoke solutions that often do not carry over to new designs or other design tools.
- They involve the creation of additional connections directly in the netlist, in the testbench, or outside the design elaboration altogether.
- They offer less supply configuration verification coverage by leaving important connections outside of the unified UPF verification flow.

The UPF HDL supply tunneling features introduced in UPF 4.0 overcome all these weaknesses by providing a standard method for forming analog supply connections using the UPF-centric design flow.

The concept of UPF HDL supply tunneling is simple in principle, but the many possibilities of different design configurations can create confusion about its proper application. The goal of this paper is to elucidate the intent of the IEEE 1801 Working Group about the intended function of UPF HDL supply tunneling using progressively more detailed design examples, exposing the flexibility, usefulness, and limitations of the concept. This paper solely represents the views of the author, and does not necessarily represent a position of either the IEEE P1801 Working Group, the IEEE DASC Standards Committee, IEEE or the IEEE Standards Association.

II. SIMPLE CASES

A. Simple HDL supply tunneling with no UPF connection

A block diagram for this case is shown in Figure 1. Sample UPF code for this example is in Figure 2. The design example of Figure 1 shows an analog supply source such as a linear regulator UA1 that supplies power to another analog functional block UA2. The supply net VDDA that will connect the two blocks is created by the UPF command in line 1 of Figure 2. The connection between net VDDA and the two blocks is created by the UPF command in line 2. Since the supply net is created with the "-tunneling force" option, simulation will proceed by representing the net VDDA using the custom UDN, and no translation to the UPF supply net type will occur. Even though a VCM is specified in the UPF command, as an optimization the tool need not place any VCMs. (See Section V for occasions when it might be an advantage to place an optional VCM.) Thus, information expressible by the custom UDN about the power needs of UA2 can be transported to UA1 via the custom UDN and will not be lost during translation to the UPF supply net type and back to the custom UDN.

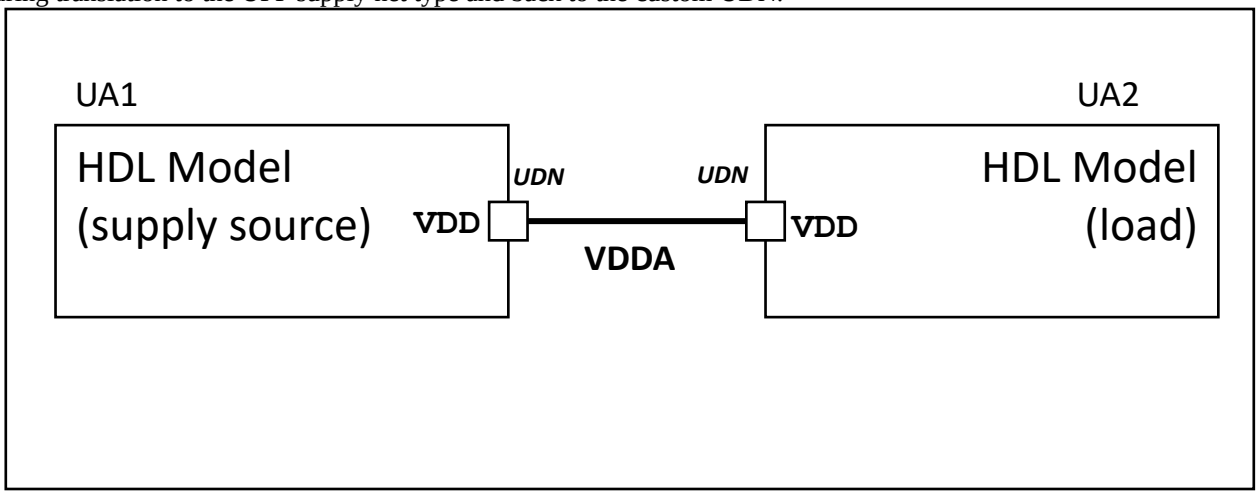

Figure 1: Simple HDL supply tunneling design example

```
1:  create_supply_net VDDA -tunneling force -resolve unresolved

2:  connect_supply_net VDDA -ports {UA1/VDD UA2/VDD} -vcm UDN2UPF
```

Figure 2: UPF code for simple HDL supply tunneling design example

B. Simple HDL supply tunneling with connection to UPF

Figure 3 shows a block diagram of a design example that uses HDL supply tunneling to connect two power-aware analog blocks and also connects to non-power-aware digital blocks. Figure 4 contains the UPF code for this example. Referring to Figure 3, analog source UA1 provides power on net VDDA for analog block UA2, but

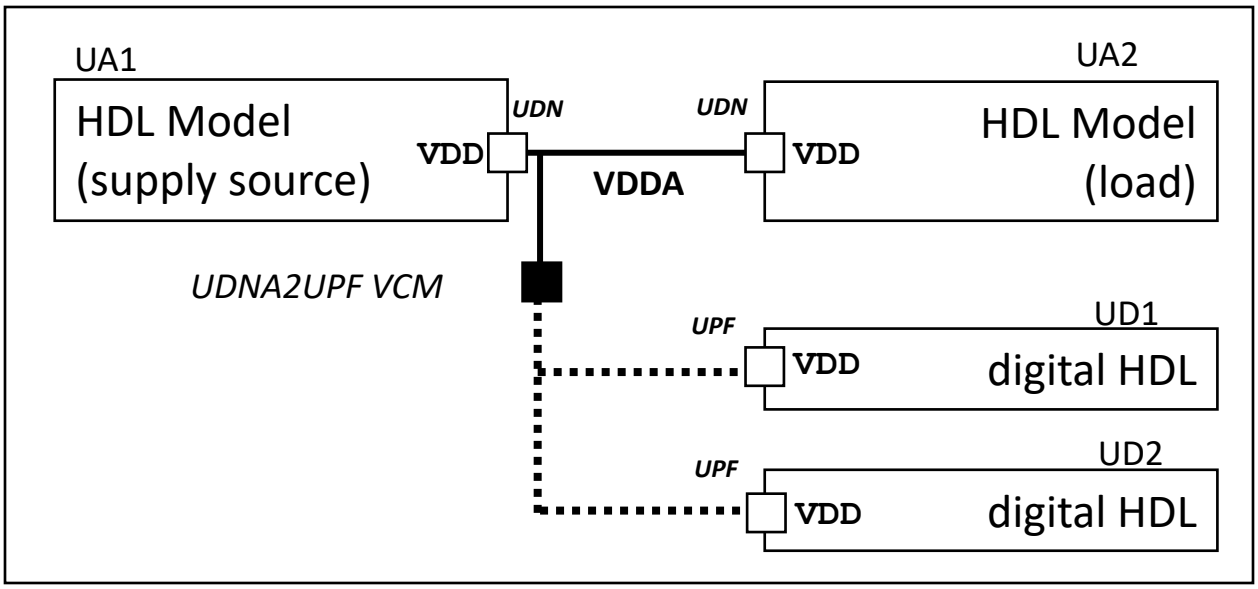

Figure 3: Design example for simple HDL supply tunneling with UPF connection

also for digital blocks UD1and UD2 via UPF supply ports VDD created on lines 3 and 6 of Figure 4. In contrast with the previous example, in this case since the connection to the UPF supply type port on UD1 and UD2 is required, the placement of a VCM is mandatory. A UDNA2UPF VCM (which has a direction of hdl2upf) is selected in line 8. Due to the direction of the VCM, any change in the value of VDDA will be propagated to the VDD port of UD1 and UD2.

```
1: create_supply_net VDDA -tunneling force -resolve unresolved
2: set_scope UD1
3: create_supply_port VDD
4: set_scope ..
5: set_scope UD2
6: create_supply_port VDD
7: set_scope ..
8: connect_supply_net VDDA -ports {UA1/VDD UA2/VDD UD1/VDD UD2/VDD} -vcm UDNA2UPF
```

Figure 4: UPF code for simple HDL supply tunneling with UPF connection example

Figure 5 shows a similar case with a single digital block UD1. A UPF2UDNA VCM, which has a direction of

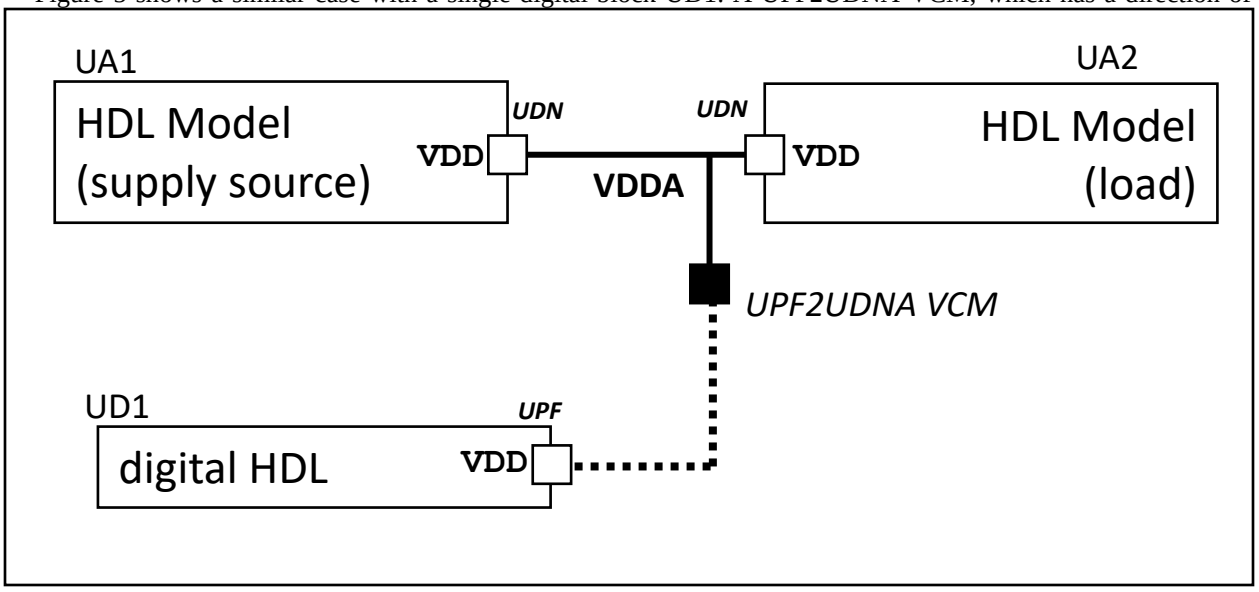

Figure 5: Design example for HDL supply tunneling with UPF driver

upf2hdl, was selected by the user. HDL supply tunneling from UA1 to UA2 occurs, but the VCM becomes an additional driver on the portion of VDDA represented as UDNA. The net value delivered to UA2 will be the result of resolving the drivers from UA1 and the VCM using the defined resolution mechanism for UDNA. Code for this example is in Figure 6.

```
1: create_supply_net VDDA -tunneling force -resolve unresolved
2: set_scope UD1
3: create_supply_port VDD
4: set_scope ..
5: connect_supply_net VDDA -ports {UA1/VDD UA2/VDD UD1/VDD} -vcm UPF2UDNA
```

Figure 6: UPF code for HDL supply tunneling with UPF driver

III. Complex Cases

A. HDL supply tunneling with multiple UDNs

If the simulation tool has a defined method for merging two or more nets declared with different nettypes, HDL supply tunneling through UPF can still occur. The HDL nets should be merged using the defined method and simulation should proceed with the merged HDL net connecting the blocks as specified in the UPF. If no connection to UPF is required, then no VCM is needed (except as noted in Section V).

B. HDL supply tunneling, multiple UDNs and UPF

If connection to UPF is required, the design example shown in Figure 7 and the UPF code of Figure 8 is relevant. Instances UA1 and UA2 are both sources that drive VDDA, each through a port of a different type UDN. Loads on VDDA include analog block UA3 and digital block UD1. As in the previous example, the merging of dissimilar

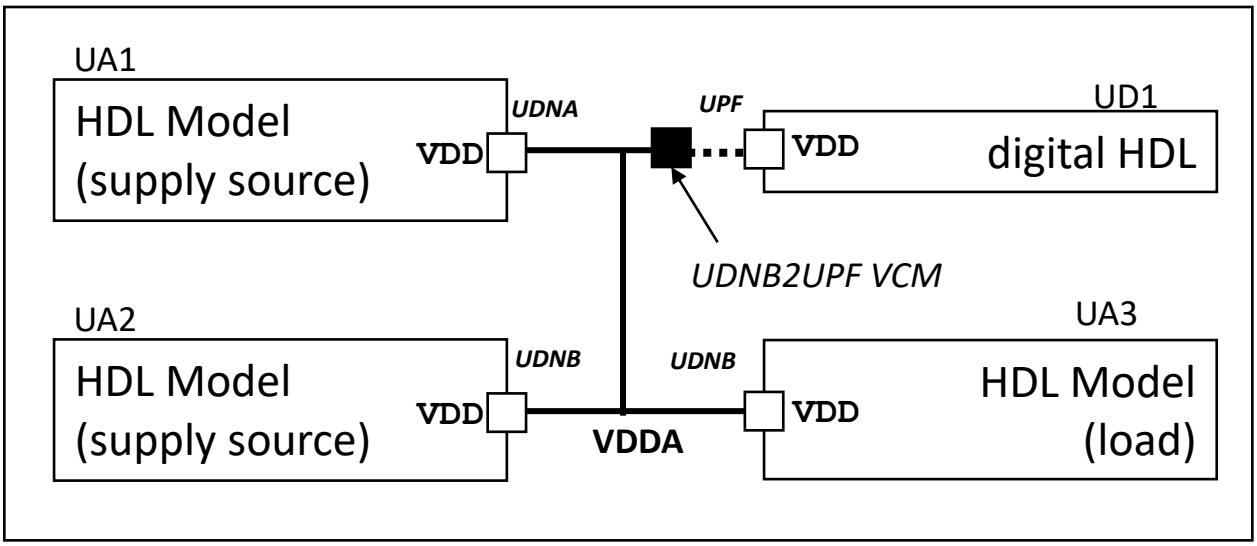

Figure 7: Design example for tunneling multiple UDNs with UPF connection

HDL nettypes will be handled in the HDL simulator via the defined method. Connection to UPF will be via the VCM requested in line 5 of Figure 8, which is of type UDNB2UPF. The simulator will have to resolve the drive to VDDA from UA1 and UA2, and pass this resolved value to the input of the VCM expressed as a net of type UDNB (the type of UA2 and UA3). This UDNB value will also be tunneled to the supply port on UA3.

```
1: create_supply_net VDDA -tunneling force -resolve unresolved
2: set_scope UD1
3: create_supply_port VDD
4: set_scope ..
5: connect_supply_net VDDA -ports {UA1/VDD UA2/VDD UA3/VDD UD1/VDD} -vcm UDNB2UPF
```

Figure 8: UPF code for tunneling multiple UDNs with UPF connection

Note that the connection in line 5 could have requested a VCM of type UDNA or UDNB. The simulator has the responsibility to pass the resolved value of the net to the VCM in the form the VCM expects it. Similarly, if UA3 had a supply port of type UDNA, the tunneled value delivered to UA3 by the simulator would have to be typed UDNA.

IV. HDL Supply Tunneling and Supply Switches

A. Connecting analog block supplies through a UPF power switch (HDL supply tunneling not possible)

Figure 9 shows a design example in which an analog source UA1 provides power for another analog block UA2

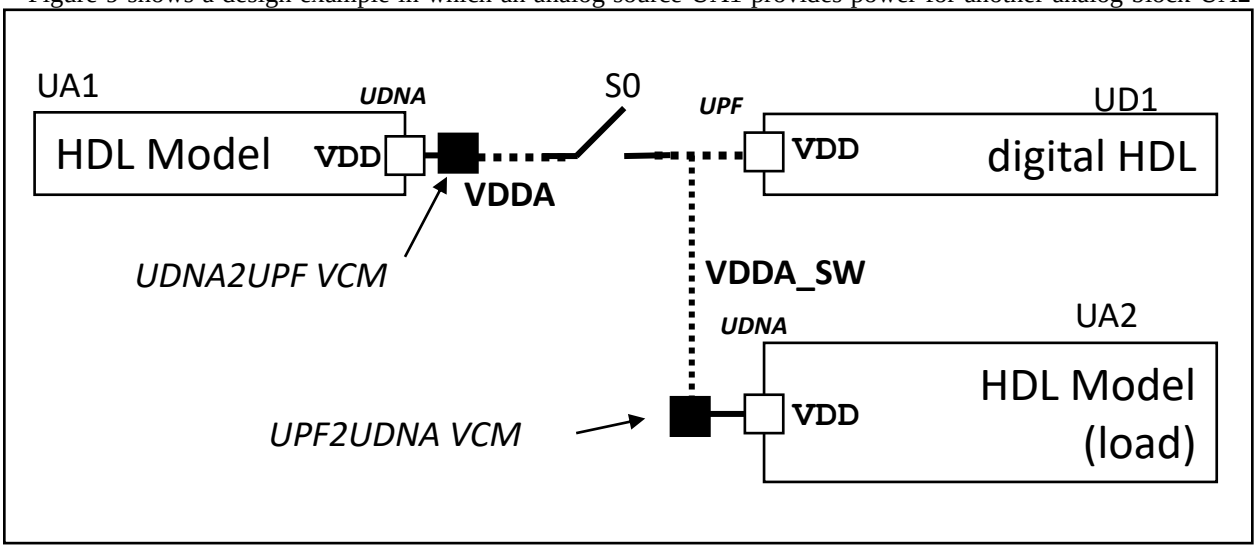

Figure 9: Design example including a UPF power switch

as well as a digital block UD1, but the power supply network contains a UPF power switch S0. The power switch is a UPF object which by definition has input and output ports of UPF supply net type. Thus, signals driven by the UDNA type port of UA1 must be converted to the UPF supply net type by a VCM in order to pass them to the input of S0. Similarly, the UPF supply net type signal at the output of S0 must be converted to UDNA before it can be passed to the input port of UA2. Direct HDL supply tunneling from the port of UA1 to the port of UA2 would bypass the switch and lose information about the state of the switch. Since this information is very important for simulating proper circuit function, this is therefore not an application for HDL supply tunneling. UPF code for this design example is shown in Figure 10.

```
1: create_supply_net VDDA
2: create_suppy_net VDDA_SW
3: create_power_switch S0 -input_supply_port VDDA  \
4:     -output_supply_port VDDA_SW -control_port {EN vddaEn} \
5:     -on_state {on VDDA {EN}} -off_state {off VDDA {!EN}}
6: set_scope UD1
7: create_supply_port VDD
8: set_scope ..
9: connect_supply_net VDDA -ports {UA1/VDD S0/VDDA} -vcm UDNA2UPF
10: connect_supply_net VDDA_SW -ports {UA2/VDD UD1/VDD S0/VDDA_SW} -vcm UPF2UDNA
```

Figure 10: Code for design example with a switch

Note that VDDA and VDDA_SW are two independent nets that do not have "-tunneling" defined. Even if these nets were created with "-tunneling force," HDL supply tunneling would not occur from UA1 to UA2, and the resulting possible loss of signal information is inevitable. The sections below provide two methods for retaining the UDN signal information.

B. Connecting an analog supply to a UPF power switch with HDL supply tunneling

One method to allow HDL supply tunneling in a design with a switch is to include in UPF an explicit connection directly from an analog source to its analog load. This allows for preservation of detailed UDN signal information, but discards information about the state of the switch and possibly does not represent the actual circuit design. An example is shown in Figure 11, with UPF code in Figure 12.

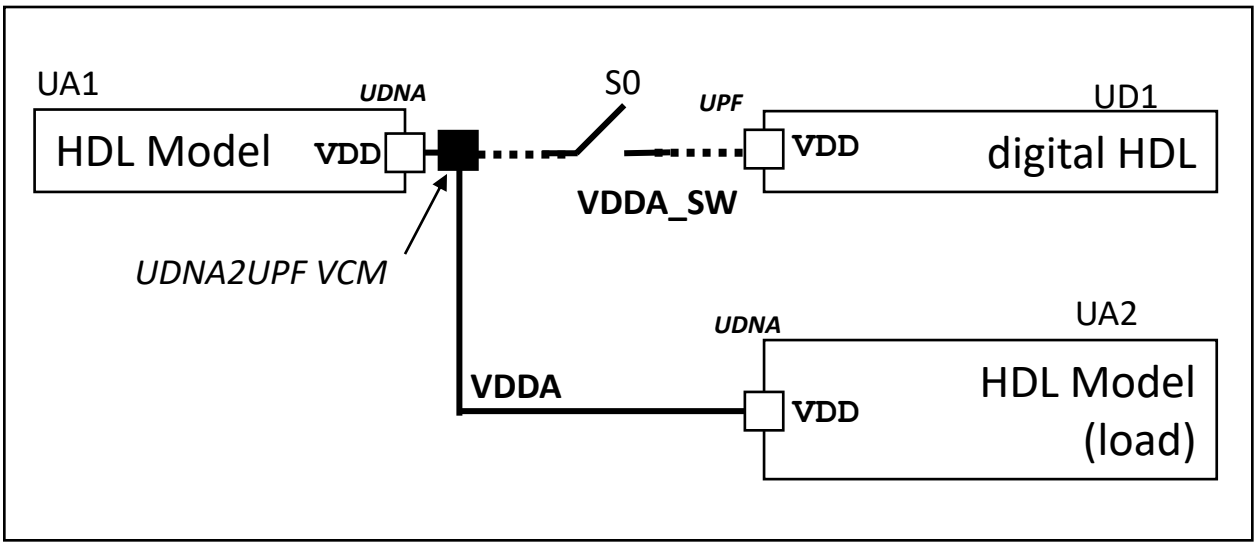

Figure 11: Design example of tunneling bypassing a switch

```
1: create_supply_net VDDA -tunneling force
2: create_suppy_net VDDA_SW
3: create_power_switch S0 -input_supply_port VDDA  \
4:     -output_supply_port VDDA_SW -control_port {EN vddaEn}
5: set_scope UD1
6: create_supply_port VDD
7: set_scope ..
8: connect_supply_net VDDA -ports {UA1/VDD S0/VDDA UA2/VDD} -vcm UDNA2UPF
9: connect_supply_net VDDA_SW -ports {UD1/VDD S0/VDDA_SW}
```

Figure 12: UPF code for example of tunneling bypassing a switch

C. Use of an HDL power switch with UPF

Another method of dealing with switched power networks in simulation without losing detailed signal or switch state information is to use an HDL switch model, hand-instantiated in the netlist. A diagram of a design example showing this method is in Figure 13. UPF code for the example is in Figure 14.

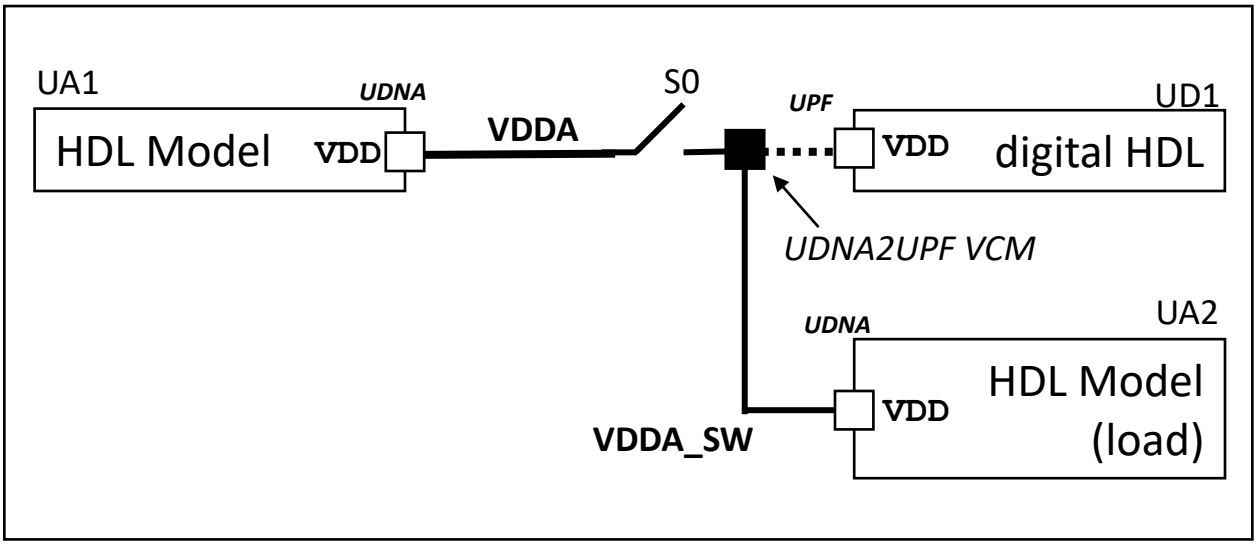

Figure 13: Design example for using an HDL switch model

```
1: create_supply_net VDDA -tunneling force
2: create_suppy_net VDDA_SW -tunneling force
3: set_scope UD1
4: create_supply_port VDD
5: set_scope ..
6: connect_supply_net VDDA -ports {UA1/VDD S0/VDDA}
7: connect_supply_net VDDA_SW -ports {UD1/VDD S0/VDDA_SW UA2/VDD} -vcm UDNA2UPF
```

Figure 14: UPF code for example using an HDL switch model

A drawback of this method is that UPF cannot be used to place the switch as is common practice in digital design flows. Another drawback is that the control signal must be manually routed to the switch instance in the netlist. However, if the HDL switch model has ports typed with the custom UDN and proper behavioral code, it can pass through all the detailed signal information carried by the UDN, while also correctly modeling the switching behavior and its impact on downstream modules.

V. Special Situations

A. Placement of optional VCMs

If during a simulation an Information Model function such as supply_on(), supply_off(), or set_supply_state() will be used to interact with a tunneled net, the simulation tool may choose to place an optional upf2hdl VCM on that net even when it has no connection to any other UPF object. The desired UPF supply net will then be attached to the VCM but have no other connections. The Information Model function, when called, will then have the effect of changing the drive to the UDN during the simulation. If there is no VCM with a upf2hdl direction, the tool may instead issue a warning or error rather than create the useless UPF object.

B. HDL supply tunneling priority and supply net resolution

The tunneling attribute attached to a UPF supply net when it is created can be inherited by other UPF supply nets that are connected to it. In case of conflicts, the tunneling attribute for the joined nets will be resolved according to the following priority:

- *INHIBIT / FORCE*
- *AUTOMATIC*
- *UNSPECIFIED*

Attempts to connect two nets, one of which has tunneling of INHIBIT and the other of which has tunneling of FORCE, shall result in an error.

Any attempt to tunnel through a UPF supply net that has any resolution declared shall result in an error. This is why the supply nets in our examples above generally have “-resolve unresolved” to make this explicit.

VI. Summary

UPF HDL supply tunneling provides integrated circuit designers with a powerful way to describe on-chip power networks using the well-known UPF language, but simulate them with much more expressive HDL nettypes. Mixing of tunneled and non-tunneled pathways on the same power network is supported, allowing detailed simulation of complex design configurations. Provisions are made for optimizing the representation to balance accuracy and simulation performance. The UPF HDL supply tunneling feature is defined in the recently released IEEE 1801-2024 (UPF 4.0) standard.

Acknowledgment

The author thanks his colleagues of the P1801 Working Group for their creativity in helping to devise many of the design examples of this paper.

References

[1] "IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems," in IEEE Std 1801-2024 , (Publication pending).