Hierarchical UPF Design – The 'Easy' Way

Brandon Skaggs
brandon.skaggs@infineon.com

Chris Turman
chris.turman@infineon.com

Joe Whitehouse
joe.whitehouse@infineon.com

Cypress Semiconductor, An Infineon Technologies Company

***Abstract* -- While there are obvious benefits to reusing pre-verified, IP-centric Unified Power Format (UPF) files within a system-level power intent definition, there are also several concerns related to EDA tool support of the required LRM constructs and complications that arise from applying power intent to new design scopes. A summary of the issues encountered, approaches to adopt (or avoid), and the implications on design and verification flow requirements is presented. Recommendations for future UPF LRM enhancements to allow better support for UPF reuse in a hierarchical design are provided.**

I. Introduction and Motivation

The IEEE1801 Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems—commonly known as the Unified Power Format (UPF) standard—provides a framework for describing the power intent and supply distribution of a design for both implementation and power-aware verification tools. Hierarchies with common supply behavior can be collected into 'domains' where their supply connectivity and expected interactions with other domains (power state information) is described in an abstract way. This abstract intent can be used by functional power-aware simulators to verify correct functional behavior along with power state transitions and correct isolation and retention during power-down. This verified power intent can then be used by synthesis tools to select appropriate power cells (level-shifters, isolation cells, or retention elements), LINTing tools to verify that the implementation is sound, and layout tools to properly implement the intent.

A. Motivation

With verification schedules pressured by time-to-market concerns, there is a strong desire to leverage verification done by IP and subsystem teams. By adopting a hierarchical UPF approach, IP UPF descriptions (written with re-use in mind) can be leveraged at higher levels of integration. There are many UPF language constructs that seem to promise 'easy' integration of hierarchical UPF files; however, not all UPF language constructs are supported equally by all EDA vendors, and some language options that seem to be exactly what would be needed for this approach do not do what you would expect…

Easy hierarchical design and re-use of IP in the RTL design world has been enabled by clear definitions of scope, port maps, and *SystemVerilog* interfaces. It's long past time we enable efficient hierarchical design of power intent in the same manner.

B. Organization of this Paper

We begin with a discussion of key terms in and concepts Section II before presenting the test setup in Section III—including a simplified system that demonstrates the real-world scenario. Section IV presents the specific issues and experimental findings. Section V presents recommendations based on the findings before conclusions are presented in Section VI.

II. Key Terms & Concepts

Unified Power Format (UPF)	IEEE standard format for describing intended design power intent abstractly.
Supply Net	A supply net represents a supply or ground net within a power domain. Supply nets are the key component of supply sets, but they can also be directly routed to the supply pins of macros or supply ports.
Supply Set	A supply set is a collection of supply nets where the function of each net has been defined, and they are used to describe sets of functions within a domain or

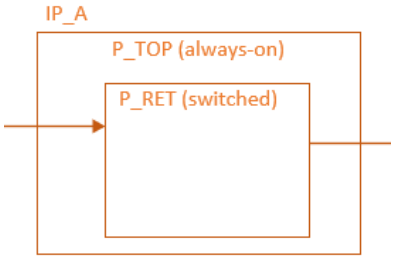
	strategy. Supply sets typically contain definitions for *power* and *ground*, but they can also contain definitions for *nwell*, *pwell*, *deepnwell*, and *deeppwell* connections. In some ways, supply sets provide the same convenience as *SystemVerilog* interfaces—which allow the description of an abstract collection of wires that can be referred to as a group.
Power Domain	A power domain defines a collection of logic that is powered in a similar way. Domains are required to define a *primary* supply set, but they can also be defined with secondary supply sets for retention or isolation. These *default_retention* and *default_isolation* supply sets are connected to the secondary/backup power of retention flops or isolation cells placed within the domain. While UPF domains are considered logical collections of instances, there are implications when it comes to the physical circuit layout. Placement tools must be provided guidance on where items of any given power domain can be placed—as it must be guaranteed that the supplies associated with the domain are available.
Scope	UPF commands are interpreted relative to a given design scope. The command *set_scope* can be used to change the active scope for any commands that follow. Also, UPF files can be loaded with a scope argument—which is equivalent to changing the active scope and design top to the scope given, applying the power intent, and reverting to the original scope and design top. UPF files can only contain references to instances at and below the current scope. Each scope brings with it a unique namespace, so the domain *PD_ACTIVE* and *u_instance/PD_ACTIVE* are seen as independent domains. Similarly; supply nets, supply ports, and supply sets defined at different scope are not implicitly connected—but instead are considered independent objects.
Equivalence	Supply nets and sets can be declared to be *functionally equivalent* and/or *electrically equivalent*. *Functional equivalence* implies parallel but independent circuitry; *electrical equivalence* implies *functional equivalence*. If a net *N* and a port *P* are connected (via *connect_supply_net* command), then *N* and *P* are electrically equivalent. The *associate_supply_set* command can also be used to declare two supply sets as functionally equivalent. Supply nets and supply sets can also both be declared equivalent with the *set_equivalent* command.
Association	A supply set can be associated with a power domain when it is created (using the 'supply' argument to *create_power_domain*)—or the supply set handle of the domain can be assigned later (using the *associate_supply_set* command). Hierarchical paths (to different scope) are allowed as arguments to *associate_supply_set* by the UPF LRM—allowing power domains and supply sets of different scope to be associated with one another.
Power State Table	Supply sets and power domains typically have power states defined—describing the allowable combination of supply net values within the design. Power states for supply nets can be defined with voltages using the *add_port_state* or they can optionally be defined without voltage (abstractly) with *add_power_state*. The power state table is used to analyze where domain boundaries exist where supplies differ in duration (more or less "ON" relatively) or voltage (higher/lower voltages between source and sink).

III. Test Setup

For our real use case, the power intent of a mature design (of a modest-sized 'wearable' SOC) was described using hierarchical design methods – rather than a top-down power intent definition. For the re-usable IP, we

focused on a highly parameterized design where the desired power intent varied based on the instantiation environment; certain portions of the IP were retained if the instance were in a switchable power domain, while certain IO were only required to be isolated when the IP were in a 'more relatively on' domain than the instantiating environment.

For the purposes of demonstration, however, a simple system shown in Figures 1-3 was constructed to demonstrate the principles discussed in the paper.



State	*P_RET* Voltage
OFF	Off
ON	1.0
RET	0.7

Figure 1. IP containing Switchable Domain and Retention strategy

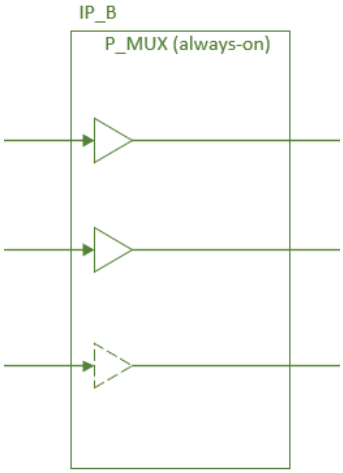


Figure 2. IP with Isolation Strategy

IP_A, shown in Figure 1, contains a top-level domain (*P_TOP*) that is relatively always-on compared to a subdomain (*P_RET*) that can be switched off—with key state elements retained. The module also defines a state (*RET*) where the retention voltage can be a lower voltage than the nominal operational voltage. Figure 2 shows *IP_B*, which contains isolation strategies for any inputs that could potentially be switched off.

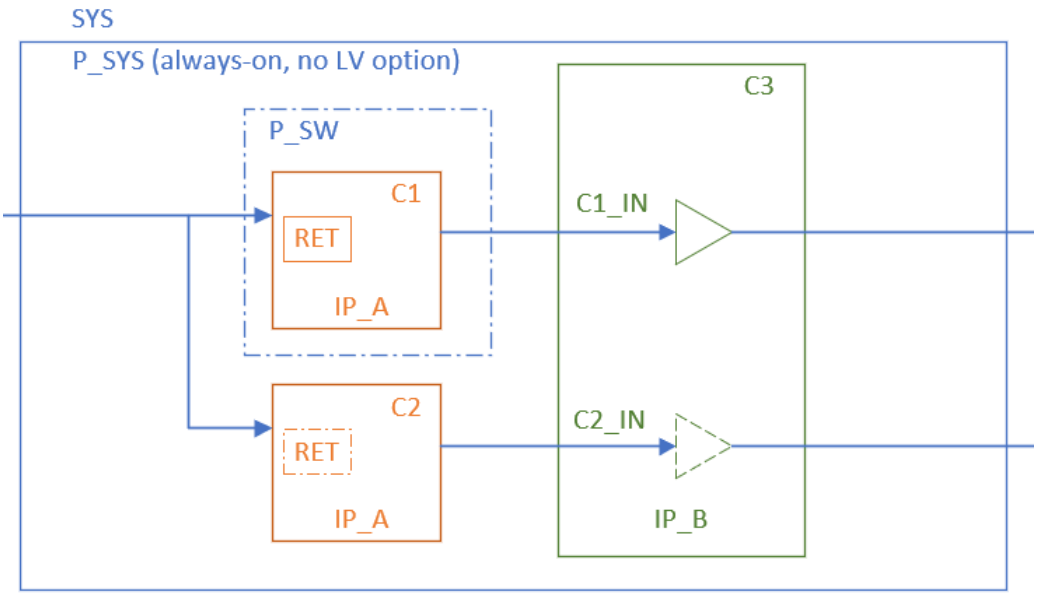


Figure 3. Hierarchical System under Test

The system shown in Figure 3 contains two instances of *IP_A* – one where the *P_RET* domain is connected to the switched off supplies (instance *C1*), and one where the switching capability isn't used. Because of this, the instance *C3* of *IP_B* needs to isolate the inputs of one but not the other.

IV. EXPERIMENTAL FINDINGS

The approach taken was to use the most abstract methods available in the UPF LRM for integrating IP power intent at the system level and evaluate the results with simulation, static multi-voltage rule checking, synthesis, and layout tools. Whenever a language construct was not accepted – or produced unexpected results—a more explicit method for integration was then applied. In this section, we'll discuss the specific language constructs attempted and the issues faced. (Note: wherever tools are mentioned, the preferred production version is discussed; notes are added where newer versions have improved support.)

A. Scoping Issues

The UPF LRM allows for the application of a power intent description to a specific instance in the hierarchy using the *load_upf* command. While this allows references (instance paths, ports, etc.) within a power intent defined at the IP scope to resolve properly when the IP is instantiated within a higher-level design, the namespace rules for UPF mean that each IP instance loading this intent creates its own scoped definition for any power domain within the UPF.

LRM Resolution Options:

The concept of combining IP domains into a higher-level domain is somewhat addressed with the *create_composite_domain* command, which was introduced in the UPF 2.1 LRM; however, as shown in Table 1, this command is not available from a practical perspective due to anemic support from EDA vendors.

	Simulator Tool A	**Synthesis Tool B**	**Formal Checker C**	**Phys Imp Tool D**
create_composite_domain	Supported	Not supported	Not supported	Not supported

Table 1. UPF Option for Combining Domains

Also, strictly-speaking, this command allows a 'super' domain to be created, but they are not meant to be physical, implementable domains, as discussed in Section 6.13 of the UPF 2.1 LRM: "*A composite power domain is a simple container for a set of power domains. Unlike a power domain, a composite domain has no corresponding physical region on the silicon…*"

Impact on Implementation:

This creates problems during implementation, because – without support for setting domain equivalence – each IP-scoped domain must be handled independently during synthesis and place-and-route.

```
...
set_attribute library_domain lib_list1 [find / -power_domain PD_SYS_AON]
set_attribute library_domain lib_list1 [find / -power_domain C1/PD_TOP]
set_attribute library_domain lib_list1 [find / -power_domain C2/PD_TOP]
...
set_attribute library_domain lib_list2 [find / -power_domain PD_SYS_SW]
set_attribute library_domain lib_list2 [find / -power_domain C1/PD_RET]
set_attribute library_domain lib_list2 [find / -power_domain C2/PD_RET]
...
```

Figure 4. Handling Scoped IP domains in Synthesis Scripts

This snippet from a synthesis setup script shown in Figure 4 shows how the proliferation of IP-scoped domains impacts basic implementation tasks. While this is not a difficult problem to resolve, the requirement to track and update all implementation task scripts each time a new IP is added is cumbersome and could be avoided if there were a way within the language to declare hierarchically-scoped domains to be treated as if they appeared at a different scope.

B. Supply Set Abstraction

Supply sets are collections of related supply nets (power, ground, nwell, pwell, etc), grouped for ease of use. Several UPF LRM commands operate on supply sets (*connect_supply_set*, *associate_supply_set*, *set_equivalent*); however, our findings were that many EDA tools either did not support these commands, or the commands were not useful for making assignments from a supply set at one level of hierarchy to another.

Figure 5 shows example syntax meant to use the 'supply set-oriented' commands to make connections between the system and IP. Table 2 shows three attempts to these commands to describe mapping between IP and system-scoped supply sets. (Note that 'N/A' entries in the table represent situations where support for the given command was not evaluated due to limitations in other portions of the flow.)

```
...
associate_supply_set P_SYS.primary -handle C1/P_TOP.primary
associate_supply_set P_SYS.primary -handle C2/P_TOP.primary
associate_supply_set P_SYS.default_retention -handle C1/P_TOP.default_retention
...
set_equivalent C1/P_TOP.primary P_SYS.primary
set_equivalent C2/P_TOP.primary P_SYS.primary
...
```

Figure 5. Attempting to Use Supply Set-Oriented Commands for Hierarchical Connections

	Simulator Tool A	**Synthesis Tool B**	**Formal Checker C**	**Phys Imp Tool D**
associate_supply_set	Partial: top-down [a]	Partial: domain [b]	N/A	Not supported [c]
connect_supply_set	N/A	Not supported	Not supported	N/A
set_equivalent	N/A	Not supported [d]	Supported	Not supported

Table 2. EDA tool support for Supply-set Oriented Commands

So, while supply sets are often thought of as analogous to *SystemVerilog* interfaces – in that they are meant to allow connections at a higher level of abstraction – they do not appear to be useful for hierarchical design because of limitations in command support and how EDA vendors expect the commands that are supported to be used.

C. Specifying 'Optional' Strategies

Another key component of hierarchical UPF design is the ability to write re-usable IP power intent specifications that become active based on the instantiating environment; however, inconsistency in UPF language capabilities and the lack of EDA tool support for native parameterization in UPF complicates matters.

For example, in the system described in Figure 3, re-usable *IP_B* instance *C3* needs to be able to isolate *C1_IN*, but does not need isolation to *C2_IN* since the *C2* instance's *P_RET* domain will be connected to a relatively-on supply. Similarly, there is no need to build the contents of C2's *P_RET* domain with retention flops.

UPF Language Inconsistency:

Isolation strategies *can* be written with the '*-diff_supply_only*' option – to allow the insertion of isolation cells only when required; however, there is no equivalent option for retention strategies. Experimental results show that the presence of these strategies was enough for retention cells to be inserted, even if—as specified by the power state table—**the related power domain is never expected to be powered off**.

Because of this limitation, the retention strategy for IP_A has to be kept in a separate UPF file that is only loaded for instances placed within switched domains (as shown in Figure 6). However, it would be preferable to allow retention strategies to be specified with an '*-if_switched*' option – so integration teams do not have to choose between manage these files auxiliary/retention UPF files or possibly getting redundant retention logic.

[a] Production version had a bug preventing bottom-up assignments; fixed in newer release.

[b] Only allowed assignments of supply sets associated with IP (sub-scope) domains – not auxiliary supply sets commonly used for port attribution.

[c] Not supported with any form of command.

[d] Command is supported for supply nets; however not for setting equivalence of supply sets.

```
...
load_upf ip_a.upf C1

load_upf ip_a.upf C2
load_upf ip_a_ret-strategy.upf C2
...
```

Figure 6: Managing 'Optional' Retention Strategies with File Segregation

'Parameterizable' Power Intent:

Parameters in UPF have been supported natively since the 2009 (i.e. 2.0) UPF LRM (via *load_upf_protected*)—a command which was later deprecated in favor of additional options added to *load_upf* in the 2015 (3.0) UPF. However, support for these commands within EDA tools remains uneven—rendering their use impractical. Table 3 shows the EDA support issues faced with the tools available for the project in question.

	Simulator Tool A	**Synthesis Tool B**	**Formal Checker C**	**Phys Imp Tool D**
load_upf_protected	Supported	Version-dependent	Not Supported[e]	Not Supported
load_upf (3.0)	Supported	Not Supported	Not Supported	Not Supported

Table 3. EDA tool support for Parameters with Loaded UPF Files

Another approach would be to provide parameters via normal TCL function; for example, IP could provide parameterization via TCL variables; however, this has a few drawbacks:

- All required TCL variables must have safely-defined default values, however…
- This creates the possibility of namespace collisions; any TCL variables used by an IP power intent would share a namespace with the sourcing (top-level) UPF. Newly-defined variables at top- or IP-level could have unintended impact on unrelated portions of the design.

D. IP Power State Re-use

In an ideal hierarchical UPF flow, the system-level power state table would be built up from the submodule power state tables; support for hierarchical power states has been in UPF since the UPF 2.0 LRM.

However, experimentally, difficulties arise when highly-parametrized IP provide states within their IP power intent definitions that are not required within the context of the target system. For our example system in Figures 1-3, *IP_A* defines a possible state *RET* that has a lower voltage on the retention supply; however, this state is not used in our system—where the retention voltage is not lowered. This scenario created errors in our Formal Checker tool:

```
// Error: (1801_PST_STATE_DROPPED_ROOT) Power state specified at root level is not consistent
with all the power state tables and is being ignored (occurrence:1)
```

Figure 7: Error indicating conflict with 'missing' IP power state

In other situations, during analysis—when conflicts existed between the system and IP power state tables or supply sets—the result was a bidirectional 'dropping' of both IP and system states—resulting in incorrect crossover analysis. The language for specifying which states should be applied in a conflict (parent or IP-provided state definition) was not available—from the UPF or the tool command options. The recommendation from the vendor was to ignore the IP power states by disabling them via tool configuration option.

The lack of support for hierarchical power state definitions meant that the most straightforward approach was to re-define the system power states and ignore IP-based definitions…

[e] Command is supported, but 'param' option is not.

V. RECOMMENDATIONS

While a hierarchical UPF design was achieved in the end, the methods required for integration by contemporary EDA tools required explicit supply port connections, explicit IP-to-system power domain mapping during synthesis and place-and-route, and re-definition of system power states – without leveraging IP-provided states.

For IP design, UPF language options that *should* have allowed a power intent to be written for multiple, different instantiation environments did not work—preventing a common IP UPF file to work for multiple instances.

From this exercise, there are several recommendations that should be considered:

- The UPF LRM should provide a mechanism for explicitly setting domain equivalence between parent and submodule domain definitions.
- UPF constructs that operate on supply nets should be supported by EDA vendors in a manner that allows connectivity – not just equivalence for power state table analysis.
- EDA vendors must support UPF language constructs that provide a mechanism for parameterized intent definitions that avoid namespace conflicts.
- The UPF LRM should include an 'if necessary' option to retention strategies to allow optional retention strategies within a switched supply.
- The UPF LRM should provide more flexibility when defining power state tables – to enable IP to specify which states can be dropped/overridden at integration level and which should be considered essential. Additionally, the LRM should define clear rules for defining hierarchical states and how conflicts should be handled.
- EDA vendors should strive to support commands related to supply set and domain equivalence.

VI. CONCLUSIONS

Design complexity and time-to-market requirements continually push for more efficient leveraging of design and verification of subcomponents, and practical hierarchical low-power design methodologies are critical for accomplishing this. However, while the components of this methodology are in place, limited support by EDA vendors for the more 'contemporary' LRM concepts—as well as some (albeit minor) inconsistencies in the UPF LRM itself—stand in the way.

As we approach the tenth anniversary of the LRM revision that provided the constructs and methodology to make it possible, this paper identifies that the practical hierarchical reuse of power intent—while possible—is a long way from being 'easy'…

REFERENCES

[1] Design Automation Standards Committee of the IEEE Computer Society, "IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems", IEEE Std. 1801-2013, 29 May 2013.

[2] Design Automation Standards Committee of the IEEE Computer Society, "IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems", IEEE Std. 1801-2015, 5 December 2015.

[3] Design Automation Standards Committee of the IEEE Computer Society, "IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems", IEEE Std. 1801-2018, 27 September 2018.