What's New in IEEE 1801 and Why?

Progyna Khondkar
Cadence Design Systems
Austin Texas
progyna@cadence.com

Abstract- ***The IEEE-1801 standard or Unified Power Format (UPF) is the ultimate abstraction necessary to enable the low power methodologies today. It provides the concepts and the artifacts of power management architecture, power aware verification and low power implementation mechanisms for any design! However, adoption of UPF and adherence to the semantics for a given design is not straightforward. Specifically the diversified power reduction schemes, different design characteristics and target applications of the design, as well various versions (releases) and variations of UPF standards makes UPF adoption painfully difficult. Interestingly a new UPF standard release does not necessarily prohibit the previous release and all variations of UPF (Accellera 1.0, IEE1801 2.0, 2.1, 3.0 and 3.1) are widely used in semiconductor industries and embraced by EDA tools. UPF is used pervasively throughout the flow from linting, verification (simulation, emulation), logic equivalence checking and implementation (synthesis, P&R). Obviously, the questions arises why so many versions of standards (LRMs)? This is mainly because of correction, amendment, extensions of semantics and addition of new syntax-semantics to enable new functionality. The changes are there because either the previous version was incomplete, misleading, error prone, easily mis-interpretable or found lacking while adopting/implementing in verification tools. This paper attempted to explain why a new standard is necessary, by pointing out the exact deficiency of previous LRMs (predecessors) with examples, as well show how the upcoming new LRM UPF 4.0 fulfilling those inadequacies based on issues found on UPF 3.1 or other older releases.***

Keywords—IEEE 1801, UPF, LRM, Semantical Syntactical changes, new commands options.

I INTRODUCTION

The Unified Power Format (UPF) also known as IEEE-1801 is not just a language to denote low power intents or power specifications for a design – it's a complete command set for designing and verifying such low power designs. The UPF is the ultimate abstraction necessary for today's low power methodologies. It provides the concepts and the artifacts of power management architecture, power aware verification and low power implementation for any design.

UPF is tcl based language that is defined separately from the HDL but has strong correlation with underlying HDL of the design. It provides the notions of power architecture from very early stage of design abstraction. Now it's the industry standard for lowering static and some special cases dynamic power dissipation in every digital design. Overwhelmingly UPF stands out as the primary choice of lowering power dissipation when fabrication process technology advanced below 25nm. UPF based low power verification and design implementation are reality today.

As the design abstraction phase's progress through implementation phases (from RTL to PG-netlist at P&R stage), UPF plays vital roles in guiding how the physical structure of the design will adopt the low-power entities. For example, during simulation or emulation, the UPF guides power states of supply set, power domain's primary, simstate, and modeling of power shutoff. UPF defines the locations & types of isolation, level-shifter, and retention cells in the design. During P&R the UPF further guides the power switch cascaded implementation, and complete power, ground, bias supply connectivity of all low-power cells (like power switch, isolation, level-shifter etc.). There are several static and dynamic power reduction schemes, like power gating, body biasing, low power standby, multi-voltage design, dynamic and/or adaptive voltage scaling etc. that fall within the realm of UPF. Further the power specification of each designs or intents are governed by its construction, architecture and by the target applications of the design, like CPU cores, IPs, SoCs, ASICs, IoTs that all have their own perspective to adopt one or multiple power reduction schemes. Despite its vital roles in overall power managements of VLSI designs, adoption of UPF and adhering to the semantics for a given design is not straightforward.

Part of this complexity is that the UPF LRM (language reference manual) itself has different versions and variations of releases from UPF 1.0 (Accellera initiative in 2007), UPF 2.0 (IEEE 1801-2009), UPF 2.1 (IEEE 1801-2013), UPF 3.0 (IEEE 1801-2015) and UPF 3.1(IEEE 1801-2018). These variations have evolved over time to meet the demands of new low power designs. Each release is designed to build on its predecessors but are not necessarily backward compatible and can vary widely semantically and moderately syntactically. Interestingly a new release does not unavoidably prohibit the previous release and all variations of UPF - 1.0 through 3.1 are widely used in industries and embraced by various EDA tools(linter, verification, logic equivalence checker or implementation tool).

Furthermore, UPF 4.0 (IEEE 1801-2024), will be released towards the end of this year 2024 or early 2025. UPF 4.0 obviously has numerous changes, specifically from UPF 3.1 as well from all previous LRMs.

This paper attempted to explain why a new standard is necessary, by pointing out the exact deficiency of previous LRMs (predecessors) with examples, as well show how the upcoming new LRM UPF 4.0 fulfilling those inadequacies based on issues found on UPF 3.1 or other older releases.

A. Motivation and Contribution of this Paper:

The primary objective of this paper is to simplify UPF adoption for any design despite of considering every complex decision process like which UPF LRM version should be picked, does it have impact on target design application, should this UPF release supports all the power reduction schemes, does front-end linting, verification (simulation and emulation), and back-end (synthesis and P&R) implementation, logic equivalent checker tools have consistent UPF support and so on. We clearly articulate how to simplify the adoption of UPF for any design, and design abstraction phase and any tool. We started our discussion from historical perspective and release timeline of different UPF versions from 1.0 to upcoming 4.0, along with comprehensive examples to compare prior standards solutions or gaps to newer improvements. We have explained every requirements of new syntax & semantics as well high-light backward compatibility issues and adoption challenges so that the paper could help audience more easily absorb the key aspects of UPF. We hope this paper, will serve as reference point for UPF adoption and implication of highly efficient low power verification and implementation platform.

B. Organization of this Paper:

This paper is organized in the following structure. Section I formulate the criticality of adoption of UPF and adherence to the semantics for a given design. Section II shows the historical perspective and timeline of UPF releases. These section also illustrate why a new LRM is required through explaining missing feature of ancestor and new capabilities of successor. Section III focus on UPF commands and options from upcoming UPF 4.0 and shows highlevel views of all the 6 LRMs. The final sections IV draws the conclusion and points out which release to adopt for a design. The references are shown at the end.

II HISTORICAL PERSPECTIVE AND TIMELINE FOR UPF RELEASE

A. The Concept of UPF and Simple Power Intent: UPF 1.0

In early 2007, the Accellera Systems Initiative introduced the UPF, also known as UPF 1.0. The fundamental concept of UPF 1.0 is to allow users to define and manage power for any design without any direct interference on the design itself- that is the golden HDL references. UPF or the power intent or the power specification can directly be overlaid on the HDL designs through EDA tools. The methodological approach of UPF abstractly model the exact power supply networks, voltage values, power area distributions, voltage protection circuitry and corresponding power states for the design. These methodologies are based on power specification or intent at the RTL and allow designers to manually refine the power network distributions of voltage areas (power domain boundaries) throughout the entire design implementation flow. Specifically manual changes are required at the post synthesis and post P&R levels of design abstraction with UPF 1.0.

The early UPF 1.0 initiatives were mostly driven from the physical design perspective in terms of explicit power supply networks - the supply ports, supply nets, their port states and possible combinations of supply states. These physical entities are mostly absent at the RTL or higher levels of design abstraction, unless the design has already been rolled out for synthesis and P&R. That means, supply ports (VDD, VSS), nets (vdd, vss), voltage values (VDD 1.2 volt), port states (ON, OFF), voltage protection circuitry (Isolation, Level-shifter, Retention, Power Switch etc.), extent of domain boundary (extents of hierarchical instances), control ports (Iso enable, Save-Restore ports), combination of supply mode, etc. are the minimum 'objects' required for developing power intents (UPF files) based on UPF 1.0. Hence, it is distinctly clear that most of these UPF objects needed to be collected from post-synthesis gate-level netlist, Liberty libraries and even P&R power-ground (PG) netlist.

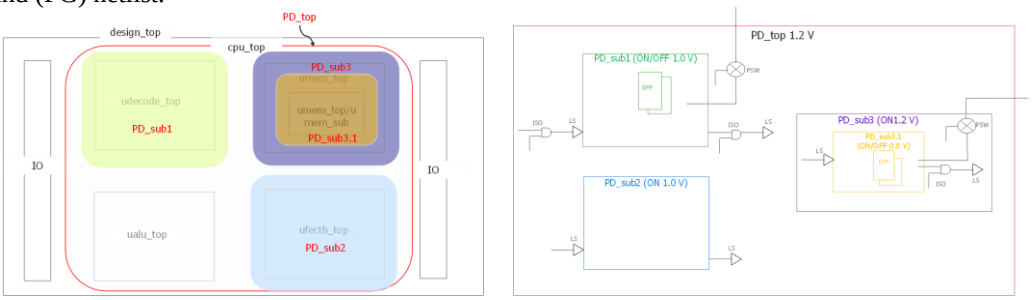

Figure 1 Concept of UPF Overlaying on HDL Blocks and Simple Power Intent UPF 1.0

So, UPF 1.0 contradicts the original power management and verification objectives – that intended to start from RTL. It pose unwanted dependency on backend design flows - that possibly should begin independently at least from the RTL. Explicit requirements of supply-port, supply-net and possible values of supply port also known as power state (actual terminology is power state table or PST) was targeted to deliver a system. This approach appeared more practical for post

synthesis power management. Because PST states are defined based on 'supply net state' only, which requires knowledge from liberty and coordination with post synthesis designs. Nevertheless, UPF (or UPF 1.0) became very popular and successful for introducing low power objects in a design without touching or modifying the HDL design. It worth to mention that before UPF, any low power implication on a design required direct modification in HDL. Table 1 shows the categories and actual syntax of UPF commands that became available in UPF 1.0.

Table 1 UPF 1.0 Syntax for Power Management, Verification and Implementation

Navigation	Supply Nets	Power Domains	Power States	Strategies	Implementation	HDL Interface	Management
set_scope set_design_top	create_supply_port create_supply_net connect_supply_net create_power_switch	create_power_domain set_domain_supply_net	add_port_state create_pst add_pst_state	set_retention set_retention_control set_isolation set_isolation_control set_level_shifter	map_retention_cell map_isolation_cell map_level_shifter_cell map_power_switch_cell	bind_checker create_hdl2upf_vct create_upf2hdl_vct	upf_version load_upf save_upf

B. The IEEE Standard and Concept of Supply Set: UPF 2.0

In 2009, the IEEE Standard Organization published the IEEE-1801-2009 or UPF 2.0, the first standard methodology for implementing power intent, truly targeted for RTL but also equally applicable to the rest of the design abstraction levels (e.g. Synthesis and P&R level). UPF 2.0 brought three major phenomenal changes from UPF 1.0. The first one was the introduction of a concept known as "supply set" – which is actually a collection of supply nets that provide a power source. UPF 2.0 introduced the **create_supply_set** & **associate_supply_set –handle** commands, which defines the supply set. It also abstracts **power**, **ground** and bias (**nwell**, **pwell**) functions and allows referencing to supply set and its handles. The second one was allowing progressive refinement of previously defined UPF objects with **–update** in most categories of UPF commands. The –**update** eventually eliminated the manual refinement process required in UPF 1.0 when design abstraction phases transitioned from RTL to Synthesis to P&R. The third one was replacing power state table PST - that comprise of a combination of three commands- **add_port_state, create_pst, add_pst_state** with a single **add_power_state** command. The **add_power_state** not only concise the sets of PST commands, but also allows to denote power states of a supply set (PD.**primary** in terms of –**supply_expr{}**), power domains (PD in terms of –**logic_epxr{}**), refine previously defined power states through –**update** as well allows to capture complex power state relationships like hierarchical states dependencies, which were evidently missing in UPF 1.0. All these new concepts accommodate smooth power management schemes from RTL - much better than that of UPF 1.0.

Figure 2 shows the basic concept and construction of supply set that comprise of a bundle of six predefined power supply function - **power, ground, nwell, pwell, deepnwell** and **deeppwell**.

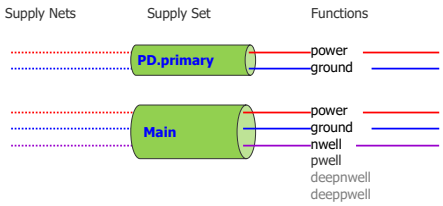

Figure 2 Basic Construct of Supply Set

Supply sets are created in a logic scope using **create_supply_set** command. Local supply set handles (local supplies) are created for power domain using **create_power_domain** PDTop **–supply {}**. The abstract supply set network is formed by associating global supply sets with local supply set handles. The **primary** supply set handle is automatically created for each power domain e.g. PDTop.**primary**. In addition, supply set for isolation and retention is also created for PDTop.**default_isolation** and PDTop_**defult_retention**. However, they are deprecated later UPF 3.0 due to semantical conflicts hence they are not recommended to use anymore. User can create additional supply set handles within a power domain as shown in snippet below.

Example 1: Supply set handles in power domain

```
11 create_supply_set aon_ss -update -function {nwell}
12 create_power_domain  PDTop -elements inst1\
...
21 -supply { ret_ss backup_ss . . .} \
22 -supply { backup_ss  aon_ss}
```

It is possible to add additional **supply_set** handles like PDTop.**backup_ss** and PD_Top.**ret_ss** using –**supply** as shown in line 21 and the associate PDTop_backup with another **supply_set** aon_ss as shown in line 22. In general, a supply set handle is considered "a local supply set" and the handle can be referenced where needed within the power domain. The **supply_set** created is also available to other child domains. Apart from supply set, Table 2a and 2b shows the categories

and actual syntax of UPF commands that became available from UPF 2.0 (green color text) on top of the existing UPF 1.0 commands (black color text).

Table 2a UPF 2.0 Syntax for Power Management Power Management, Verification and Implementation

Navigation	Supply Nets	Power Domains	Power States	Strategies	Implementation
set_scope set_design_top	create_supply_port create_supply_net connect_supply_net create_power_switch	create_power_domain **-update** set_domain_supply_net **create_composite_domain -update**	add_port_state create_pst add_pst_state **add_power_state -update** **describe_state_transition**	set_retention set_retention_control set_isolation set_isolation_control set_level_shifter **set_retention_elements** all with -update	map_retention_cell map_isolation_cell map_level_shifter_cell map_power_switch_cell **use_interface_cell**

Table 2b UPF 2.0 Syntax for Power Management Power Management, Verification and Implementation

HDL Interface	Code Management	Supply Sets	Attributes	Control Logics	Simstate
bind_checker create_hdl2upf_vct create_upf2hdl_vct	upf_version load_upf save_upf load_upf_protected load_simstate_behavior find_objects	create_supply_set supply set handles associate_supply_set connect_supply_set all with -update	set_port_attribute set_design_attribute HDL and Liberty Attributes	create_logic_port create_logic_net connect_logic_net	**set_simstate_behavior** **add_power_state**

The addition of supply set (**create_supply_set, associate_supply_set, connect_supply_set** etc.), new power state (**add_power_state**) and refinements of previously defined UPF objectives through –**update** in UPF 2.0 enriched power management adoption schemes and verification mechanism on any designs. However, many of the semantics were premature and semantics and (or) syntax were incomplete to deploy UPF in every level of design abstraction coherently. It was evident at this point that UPF is essential to manage or mitigate power on any design however UPF was still evolving for its maturity.

UPF 2.0 is backward compatible with UPF 1.0 (create_supply_set, supper set of supply port, net). It has supports for IP development, refinement (add_power_state, create_logic_port, and –update). Following are notable changes in UPF 2.0 from UPF 1.0 – to start verification from RTL

- Introducing create_supply_set & associate_supply_set –handle
- Abstraction of power, ground and bias (nwell, pwell) functions
- Reference to supply set and handles
- Introducing -update in many UPF commands
- Refinement of UPF objects as progressing through design phase transition
- Replacing Power State Table with add_power_state command
- add_port_state, create_pst, add_pst_state replaced with add_power_state

All these helps to start power management from higher level of design abstraction (RTL)

C. The Evolution of IEEE 1801 Standards: UPF 2.1

The actual evolution of UPF specifically started from IEEE Standard 1801-2013 or UPF 2.1 published in May 2013. Because UPF 2.1 brought more semantical clarification that were missing or incomplete in UPF 2.0 and UPF 1.0, than introducing new set of commands and options. As a consequence, UPF 2.1 also labelled some UPF 1.0 & 2.0 commands (and options) as "obsolete" or "legacy" (red color texts) and discourage their usage on semantical ground to avoid conflicts and confusion.

Table 3a UPF 2.1 Syntax for Power Management Power Management, Verification and Implementation

Navigation	Supply Nets	Power Domains	Power States	Strategies	Implementation
set_scope set_design_top	create_supply_port create_supply_net connect_supply_net create_power_switch	create_power_domain -update **set_domain_supply_net** create_composite_domain -update	**add_port_state** **create_pst** **add_pst_state** add_power_state -update describe_state_transition	set_retention **set_retention_control** set_isolation **set_isolation_control** set_level_shifter set_retention_elements all with -update set_repeater	map_retention_cell **map_isolation_cell** **map_level_shifter_cell** map_power_switch_cell use_interface_cell

Table 3b UPF 2.1 Syntax for Power Management Power Management, Verification and Implementation

HDL Interface	Code Management	Supply Sets	Attributes	Control Logics	Simstate	Power Management Cell
bind_checker create_hdl2upf_vct create_upf2hdl_vct	upf_version load_upf save_upf load_upf_protected load_simstate_behavior find_objects **begin_power_model** **end_power_model** **apply_power_model**	create_supply_set supply set handles associate_supply_set connect_supply_set all with -update **set_equivalent**	set_port_attribute set_design_attribute HDL and Liberty Attributes	create_logic_port create_logic_net connect_logic_net	set_simstate_behavior add_power_state	**define_always_on_cell** **define_diode_clamp** **define_isolation_cell** **define_level_shifter_cell** **define_power_switch_cell** **define_retention_cell**

Following major changes came in UPF 2.1

- Clarify Repeater insertion and verification (set_repeater) Vs SPA –repeater_supply
- Also create_power_domain –available_supply for Verifying Supply Constraints

- Atomicity of power domains for soft IP (create_power_domain –atomic)
- Hard Macro Modeling, Integration & Verification
- (begin_power_model~end_power_model, apply_power_model)
- Protection (Isolation) cell at Hard Macro Boundary (HighConn, LowConn Concept)
- Supply Equivalence supply_equivalent
- Power Management Cell modeling commands (define_* Vs SPA)
- Precedence of Strategies ret, repeater, iso, ls
- Retention Semantics (Balloon Vs Master-Slave)
- PST became legacy

In UPF 2.0, modeling hard IPs or macros was difficult – it requires power domain, related supplies, etc. Such spec were directives for verification, as well implementation –which is deviation from ultimate UPF objectives. UPF 2.1 introduces **begin/end_power_model** and that defines the power intents or UPF for hard IP to provide the boundary concepts for hard macros. **begin/end_power_model** explicitly became descriptive to imply that, it describe the power intent of hard IP for validating in the SoC environment. At implementation this will automatically ignore the commands within a power model. UPF 2.1 also introduces, **apply_power_model** for easy association of the model to design binds to an instances in the design and connects the interface supply set handles and logic ports of a previously loaded power model.

Another important changes came in 2.1 is **set_equivalent**, which defines when supply set or supply net are electrically or functionally equivalent. UPF strategies needs to match -source -sink filtering of supplies for strategies like ISO, LS etc. There was no such semantics for supply matching in UPF 2.0 that explicitly defines supply equivalence. **set_equivalence** is the rules of matching of supplies and make tool consistent on strategies like **set_isolation, set_level_shifter** and **set_repeater** will by default use supply equivalence that helps to determining which ports be isolated when filtering is specified with a **-sink** or **-source** or **-diff_supply_only** filters, shown in Figure 3.

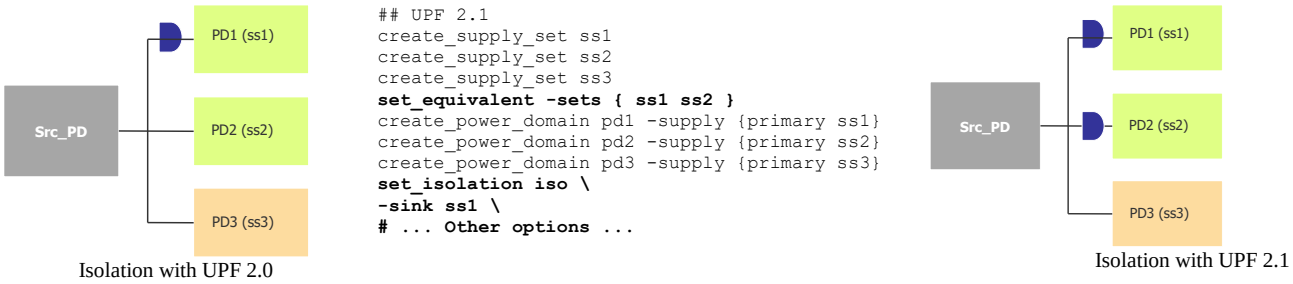

Figure 3 **set_equivalnce** correctly determines isolation for PD1 and PD2

UPF 2.1 also introduces definitions for power management cells like isolation, level shifter, power switch, as well always on and diode cell as UPF commands (define_always_on, define_diode_cell etc. as shown in Table 3a). In previous release with UPF 2.0, **set_port_attribute** provides some liberty attribute but not all. Hence user required liberty files (.lib) at front end RTL simulation to specify any liberty attributes. However, these defined cells do not alter the existing library cell definitions. When the liberty libraries are available, these commands can be leveraged by verification tools to perform accurate verification, that closely matches the implementation results.

D. The Information Model of IEEE 1801 Standards: UPF 3.0

UPF 3.0 standard was released in December 2015 which clarifies and enhances UPF 2.1 features (new semantic of power model), adds a few new capabilities (SPA is_analog, -parameter for bind_checker, UPF Info Model). Table 4a and 4b highlights new syntax came in UPF 3.0. UPF information model (UPFIM) introduces a concept model that captures "Power-management Information" from "UPF semantics" on a "design".

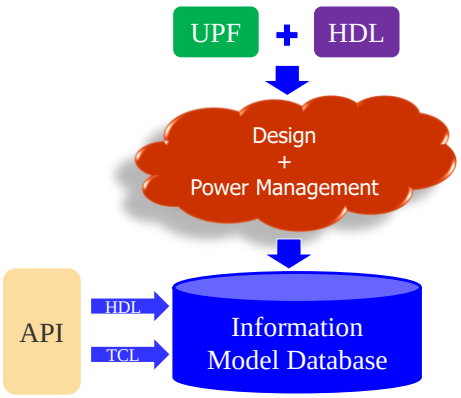

Figure 4 UPF 3.0. introduces UPF information model (UPFIM)

The UPFIM contains information from UPF objects and user Design (i.e. HDL Objects). Note that UPF power domain, supply sets, design groups, design models, design instances are collectively known as UPF Objects. The model allows to access (read/write) the Information through HDL and Tcl API. These Tcl/HDL Package API functions allows to develop variety custom, productive and intuitive of PA debug & verification applications. These applications can be run both at static time, to get static information or post sim to get both static and dynamic data.

Following shows some example applications on UPFIM that helps to catch low power bugs at RTL:

- Source of Corruption/Retentions of a Signal (static properties)
- Find the current state of a power domain (dynamic properties)
- Custom Debug and Verification Reporting
- Coverage using HDL Package Functions (Random Directed Coverage)
  - e.g. Isolation/retention/level-shifter control signal coverage
  - e.g. Power States and Power State Transitions coverage
- Assertions for error scenarios
  - e.g. if two power domains are in mutually exclusive states
  - Based on UPF 3.0 Property types; User functions need to be created

Table 4a UPF 3.0 Syntax for Power Management Power Management, Verification and Implementation

Navigation	Supply Nets	Power Domains	Power States	Strategies	Implementation	Variation/Correl.
set_scope set_design_top	create_supply_port create_supply_net connect_supply_net create_power_switch	create_power_domain -update **set_domain_supply_net** create_composite_domain -update	**add_port_state** **create_pst** **add_pst_state** add_power_state -update describe_state_transition **add_supply_state** **add_state_transition** **create_power_state_group**	set_retention **set_retention_control** set_isolation **set_isolation_control** set_level_shifter set_retention_elements all with -update set_repeater	map_retention_cell **map_isolation_cell** **map_level_shifter_cell** map_power_switch_cell use_interface_cell	**set_correlated** **set_variation**

Table 4b UPF 3.0 Syntax for Power Management Power Management, Verification and Implementation

HDL Interface	Code Management	Supply Sets	Attributes	Control Logics	Simstate	Power Management Cell
bind_checker create_hdl2upf_vct create_upf2hdl_vct	upf_version load_upf save_upf load_upf_protected load_simstate_behavior find_objects begin_power_model end_power_model apply_power_model **add_parameter**	create_supply_set supply set handles associate_supply_set connect_supply_set all with -update set_equivalent	set_port_attribute set_design_attribute HDL and Liberty Attributes	create_logic_port create_logic_net connect_logic_net	set_simstate_behavior add_power_state	define_always_on_cell define_diode_clamp define_isolation_cell define_level_shifter_cell define_power_switch_cell define_retention_cell

Following are major Changes from UPF 2.1 to UPF 3.0

- New semantic of power model with begin/end_power_model represent
- Both Hard or Soft macro with {UPF_is_hard/soft_macro TRUE} attributes.
- New power model semantics also provides boundary concept for both Hard and Soft macros
  - For both macros – parents context will not be allowed to modify UPF inside the macros
  - Because for Hard macro – its obvious, already implemented.
- But for Soft macro – UPF is intended for separate implementation, and therefore the soft macro interface is still treated as a hard boundary
- So, in UPF 3.0 - a power model instantiated in a design can only be modified by the parent context when UPF_is_hard_macro or UPF_is_soft_macro both are FALSE.

In addition UPF 3.0 also added new syntax and obviously new semantics for) **create_power_state_group** – which defines a group name used in conjunction with the add_power_state command to give controllability/flexibility. A power state group is used to collect related power states defined by add_power_state. The -group (<group_name>) is defined in the current scope. The power state group defines legal combinations of power states in current scope and/or from the descendant subtree represents combinations of power states that can be active at the same time. Following Figure 4 shows a generic SoC and UPF example showing **create_power_state_group** for the SoC.

```
# Grouping of legal Power States
create_power_state_group PD_SOC
add_power_state -group PD_SOC \
 -state {RUN -logic_expr {primary==ON  && PD_L2==RUN && PD_COREA==RUN}} \
 -state {DMT -logic_expr {primary==OFF && PD_L2==RUN && PD_COREA==SHD}} \
 -state {SHD -logic_expr {primary==OFF && PD_L2==SHD && PD_COREA==SHD}}
 -state {RET -logic_expr {primary==OFF && LP_RET1N == 1'b1 PD_COREA == OFF}} \
 -state {OFF -logic_expr {primary==OFF && LP_RET1N == 1'b0 PD_COREA == OFF}}
add_power_state -group PD_SOC -update \
 -state "RET.PD_CPUA0_ACT -logic_expr {PD_COREA.PD_CPUA0.ACT} -illegal" \
 -state "OFF.PD_CPUA0_ACT -logic_expr {PD_COREA.PD_CPUA0.ACT} -illegal"
```

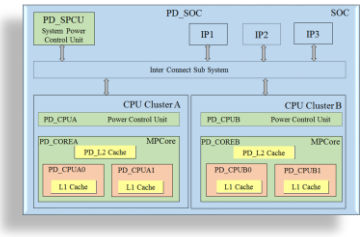

Figure 4 UPF 3.0 example showing **create_power_state_group** for an SoC.

UPF 3.0 also introduces **bind_checker** – a mechanism by embedding the binding of the design and checker within the UPF/Tcl file through the bind_checker command. The bind_checker allows to model assertions that are distinctively different from SystemVerilog assertions in that they can access all the UPF objects – i.e. UPF power supply, power states etc.

E. More Evolution of of IEEE 1801 Standards: UPF 3.1

UPF 3.1 standard was released on September 2018 mostly adhering to correcting and enhancing UPF 3.0 features like path based strategies from port base, new precedence rules, as well adding new capabilities in UPFIM, power model, hard and soft macro, terminal boundary and find_objects. Not all these changes necessarily turned into new syntax but only few to name like new commands for verification (simulation & emulation control) shown in Table 5c.

Table 5a UPF 3.1 Syntax for Power Management Power Management, Verification and Implementation

Navigation	Supply Nets	Power Domains	Power States	Strategies	Implementation	Variation/Correl.
set_scope set_design_top	create_supply_port create_supply_net connect_supply_net create_power_switch	create_power_domain -update **set_domain_supply_net** create_composite_domain -update **boundary_supplies**	**add_port_state** **create_pst** **add_pst_state** add_power_state -update describe_state_transition add_supply_state add_state_transition create_power_state_group	set_retention **set_retention_control** set_isolation **set_isolation_control** set_level_shifter set_retention_elements all with -update set_repeater	map_retention_cell **map_isolation_cell** **map_level_shifter_cell** map_power_switch_cell use_interface_cell	set_correlated set_variation

Table 5b UPF 3.1 Syntax for Power Management Power Management, Verification and Implementation

HDL Interface	Code Management	Supply Sets	Attributes	Control Logics	Simstate	Power Management Cell
bind_checker create_hdl2upf_vct create_upf2hdl_vct	upf_version load_upf save_upf load_upf_protected load_simstate_behavior find_objects begin_power_model end_power_model apply_power_model add_parameter	create_supply_set supply set handles associate_supply_set connect_supply_set all with -update set_equivalent	set_port_attribute set_design_attribute HDL and Liberty Attributes literal_supply **exclude_model** **exclude_elements**	create_logic_port create_logic_net connect_logic_net	set_simstate_behavior add_power_state	define_always_on_cell define_diode_clamp define_isolation_cell define_level_shifter_cell define_power_switch_cell define_retention_cell

Table 5c UPF 3.1 Syntax for Power Management Power Management, Verification and Implementation

Verification Control
sim_assertion_control Control the behavior of assertions during low-power verification
sim_corruption_control Provides the ability to disable the corruptions of a specific set of design elements or types of design elements
sim_replay_control Specify initial blocks to be replayed when a domain powers up (mostly for simulation).

Following are major features changes in UPF 3.1from UPF 3.0

- Clarifies and enhances UPF 3.0 features (Path base strategy association, precedence rules)
- Adds a few new capabilities (UPF Info Model, Power Model, Hard & Soft Macro)
- **begin/end_power_model** (became legacy) and
    - New **define_power_model** introduced
    - **apply_power_model** remains with newly introduced option **-port_map**

These two binds power model to design and connects the interface supply set handles and logic ports of a previously loaded power model. The **-port_map** defines how the interface logic port or supply ports of the instance scope connects with logic nets or supply nets in the current scope respectively. The apply_power_model command supports two modes of operation:

- For system level IP power modelling:
  - Binds a system level IP power model to a cell instance within a design and binds parameter defined within that power model to objects within the environment
- For supply set connections:
  - Describes the connections of the interface supply set handles of a previously loaded power model with the supply sets in the scope where the corresponding macro cells are instantiated.

UPF 3.1 also addresses SV LRM IEEE1800 violations of 3.0 handle **upfHandleT** data type by mapping it from **chandle** to System Verilog int. UPF 3.0 defines **upfHandleT** to access objects and properties in the UPFIM upfHandleT is constructed as a member of "struct { }" in HDL API.

Why IEEE 1800 (SV) LRM Violation Occurred in UPF 3.0 LRM usage?

- While representation of upfHandleT in Native HDL in UPF 3.0 LRM upfHandleT to HDL mapping is done with '**chandle'** data type for SV.
- chandle data type represents storage for pointers passed using the DPI.
- IEEE 1800 imposes restrictions on where chandle type can be used, for e.g. **chandle** cannot be used as ports, in sensitivity lists or event expressions, in continuous assignments etc.
- This in turn limits the intended use cases of the SV HDL API functions to develop apps that will get expected data from UPFIM.

Table 6 UPF 3.1 addresses issues of 3.0 with **upfHandleT**

UPF 3.0	UPF 3.1
struct { upfHandleT handle; upfPowerStateObjT current_state; } upfPdSsObjT	struct **packed**{ upfHandleT handle; upfPowerStateObjT current_state; upfSimstateE **current_simstate;** } **upfPdSsObjT**

Here packed array allows int type, which makes implementation independent packing of members and easy access. In addition by extending **upfPdSsObjT** as native HDL type, 3.1 also allows HDL API to qualify for getting dynamic objects for simstate.

F. Furthermore Evolution of IEEE 1801 Standards: Coming in UPF 4.0

UPF 4.0 which is probably coming in early 2025, is mainly known to have a completely new feature for analog and mixed signal (AMS) for connecting and modeling supply nets across the analog/digital boundary. Specifically HDL supply tunnelling and value conversion mechanism (VCM). There are other with several changes in digital or regular part, such as 'Retention Strategies', 'Power Models', 'Supply Set/Net', 'Precedence Rules' etc. which are mainly based on implementing additional requirements that were revealed by real world usage of the earlier versions of the standard. Table 7 shows changes in retention strategy. The retention strategies are a great example of adapting to changing requirements. The UPF3.x semantics could not describe the more complex and varied behavior of the plethora of modern retention cells. The UPF4.0 semantics were created from the ground up to provide a more complete and accurate representation of these cells.

Table 7 UPF 4.0 Changes in Retention Strategy

Newly Added Options in UPF 4.0	Obsolete/Legacy from UPF 4.0
[-applies_to <flop | latch | any>] [-save_event_condition] [-restore_event_condition] [-powerdown_period_condition] [-restore_period_condition] [-async_set_reset_effect <ignored | retained_value | output_value>]	[-retention_power_net net_name] [-retention_ground_net net_name] [-save_condition] [-restore_condition] [-retention_condition]

On the other hand, there are several completely new UPF commands added in 4.0 as listed in Table 8 below.

Table 8 UPF 4.0 New Commands

Newly Added Commands in UPF 4.0	Brief Outline

create_abstract_power_source	Defines abstract model of power source of supply sets
create_vcm	AMS value conversion methods for connecting UPF supply nets and HDL ports
create_upf_library	A container to create_vcm
load_upf_library	Execute commands from specified UPF library
use_upf_library	Relevant to UPF library to make UPF object visible to current scope
map_retention_clamp_cell	Allows list of liberty cells used for constraints of clamps on clock path

Similarly, there are more commands and options added for adopting new concepts like refinable macros, virtual supplies etc. Table 9 summarizes major focus of each releases which in fact gives clear understanding how each releases are different from each other as well how they are interconnected or interdependent.

Table 9 Brief overview of different versions of the UPF Releases (Language Reference Manual) and its Focus

Release	Year	Standard	Major Focus on the Release
UPF 1.0	Feb 2007	Accellera	- Focused on adding power intent to HDL (VDD, VSS i.e. supply net and supply port) - Relatively simple concepts and commands (add_port_state, add_pst_state, create_pst for PST)
UPF 2.0	March 2009	IEEE 1801	- Backward compatible with UPF 1.0 (create_supply_set, supper set of supply port, net) - Supports IP development, refinement (add_power_state, create_logic_port, and -update)
UPF 2.1	March 2013	IEEE 1801	- Clarifies and enhances UPF 2.0 features (set_repeater, master/slave 0 pin RET, set_equivalent) - Adds a few new capabilities (define_always_on_cell, define_diode_clamp_cell & others ISO)
UPF 3.0	Dec 2015	IEEE 1801	- Clarifies and enhances UPF 2.1 features (new semantic of power model) - Adds UPFIM & new capabilities (SPA is analog, -parameter for bind checker)
UPF 3.1	Sept 2018	IEEE 1801	- Clarifies and enhances UPF 3.0 features (Path base strategy association, precedence rules) - Adds a few new capabilities (UPF Info Model, Power Model, Hard & Soft Macro)
UPF 4.0	TBD	IEEE 1801	- Clarifies and enhances UPF 3.1 features (Retention, Refinable macro, Virtual supply, power model) - Adds a few new capabilities (AMS, VCM,)

CONCLUDING REMARKS

We started this paper with simple question: why so many versions of standards (LRMs)? We have clearly came to a conclusion that the obvious answer of the questions must be - this is mainly because of correction, amendment, extensions of semantics and addition of new syntax-semantics which were previously either incomplete, misleading, error prone, easily mis-interpretable or found impossibilities while adopting/implementing in verification tools and so on. This paper attempts to explain why a new standard is necessary, by pointing out the exact deficiency of previous LRMs (predecessors) with examples, as well show how the upcoming new LRM UPF 4.0 fulfilling those inadequacies based on issues found on UPF 3.1 or other older releases. As the IEEE standard itself evolved overtime, providing additional abstraction flexibility for the other rudimentary parts of design elements, like supply set, supply networks, and supply states for design groups, models, and instances, collectively known as objects. Some of the amendments are reflected in IEEE Standard 1801-2013 or UPF 2.1 published in May 2013, while the latest update is available in IEEE Standard 1801-2015 or UPF 3.0 published in December 2015.

However, it is important to note that each amendment is not necessarily backward compatible and each updated version is usually a superset of semantic and syntactic expressions from its predecessors. The abstraction of UPF 3.0 or UPF 3.1 in general, methodically consists of power specifications that categorically itemize the design elements for a particular power block, known as a power domain (PD), along with the PD boundaries. Further, it also specifies the power supply and supply states of the power domains or supply sets, whether the supply is in On, Off, or other potential states. Depending on the power management and reduction techniques the specification adopts for the final chip, the list further extends to specify the requirements of boundary strategies for intra or inter-domain communications. These strategies may include isolations (ISO), level-shifters (LS), enable level-shifter (ELS), always-on buffers (AOB), feed through buffers or repeaters (RPT), diode clamps, retention flops (RFF), power switches (PSW), and their corresponding supply network and locations details. In short, the UPF is a precise map that models the power specification of the design and converts the same to a power aware design.

Although UPF is very well defined through IEEE 1801 LRM, it is often difficult to comprehend many primitive and inherent features of individual UPF commands-options or relations between different varieties of UPF commands-options. The semantic context between most of the UPF commands are orthogonal. However, fundamental constituent parts of UPF that buildup the power management architecture are inherently linked because of their transitive nature — specifically the UPF commands that establish the links with DUT objects; like instances, ports, and nets, etc. In this paper, we provide a simplistic approach to find inherent links between UPF commands-options through their transitive nature. We also explain how these inherent features help to foster and establish exact relationships between UPF and DUT objects in order to develop UPF for power management and implementation as well as conduct power aware verification.

REFERENCES

[1] Progyna Khondkar, “Low-Power Design and Power-Aware Verification”, Hard Cover ISBN: 978-3-319-66618-1, October, 2017, Springer International Publishing.

[2] Progyna Khondkar, et al., “How UPF 3.1 Reduces the Complexities of Reusing Power Aware Macros" March, DVCon 2020.”

[3] Progyna Khondkar, et al., “Low Power Coverage: The Missing Piece in Dynamic Simulation”, February March, DVCon 2018.

[4] Progyna Khondkar, et al., “Free Yourself from the Tyranny of Power State Tables with Incrementally Refinable UPF”, February March, DVCon 2017.

[5] Design Automation Standards Committee of the IEEE Computer Society, “IEEE Standard for Design and Verification of Low-Power,

Energy-Aware Electronic Systems", IEEE Standards 1801-2015, 5 December 2015.
[6] 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.