Stepping into UPF 2.1 world: Easy solution to complex Power Aware Verification

Amit Srivastava
Mentor Graphics
amit_srivastava@mentor.com

Madhur Bhargava
Mentor Graphics
madhur_bhargava@mentor.com

***Abstract*—The increasing complexity and growing demand for energy efficient electronic systems has resulted in sophisticated power management architectures. To keep up with the pace, the power formats have also evolved over the years. With the recent release of the IEEE P1801-2013 (UPF 2.1), several new features have been added along with improving clarity on existing features. It has also bridged the gap between UPF and CPF to provide much needed convergence. However, it has also posed some questions about the compatibility, differences, and challenges related to migration and its impact on verification. In this paper, we will provide an in-depth analysis and relevant examples of all the new features introduced by the UPF 2.1 along with highlighting any semantics differences with the earlier versions to help the user easily migrate to the new standard.**

Keywords—Power Management, Power Aware Verification, IP, UPF.

I. INTRODUCTION

A. Power Management

The growing demand for energy efficient electronic systems has resulted in sophisticated power management architectures. The constant need to minimize energy consumption to increase battery life for portable devices, and reduce heat dissipation for non-portable devices to minimize cooling costs ensure that power management is critical part of any electronics designs. Designers employ a variety of advanced techniques ranging from clock gating and power gating to multiple voltages and dynamic scaling of voltages and frequency. These techniques affect the functionality of the system if not executed correctly. Hence, it becomes important to verify the power management to ensure functional correctness of designs.

The power management consideration starts as early as the system design phase to achieve maximum benefits. It gets refined at various phases of the design cycle. Thus it becomes important to verify the power management at every stage in the design flow so that any functional bugs are rectified.

The traditional Hardware Description Languages (HDL) were not designed to consider the power related information in the description. Power intent specification formats were introduced to address this limitation. These formats allow the user to express the power intent related to power management, which can be overlaid on top of HDL description without requiring any change in normal design functionality. This specification can be used by various tools to perform verification and implementation of power managed designs.

B. Verification complexity

Power management in today's complex SoCs involves various techniques. The result is the modification of the original design and insertion of special power management structures like isolation, level shifters, retention, etc. at various places in the design. Due to the complex interaction of these structures with the normal design functionality, it poses a serious challenge to verification. To add to it, the various IPs with their own power management need proper verification to remove any integration issues related to power management. To aid the verification process, power intent specification formats can share the burden by defining clear and consistent semantics enabling tools to automate various tasks related to power management.

C. Evolution of UPF

Back in 2007, the first version of UPF was developed and released by Accellera. UPF 1.0 introduced the basic concepts relating to power management, i.e. corruption, isolation, retention and level shifting. In March 2009, IEEE released UPF 2.0 [1] and introduced various concepts like Supply Sets and power states to ease specification at higher levels of abstraction. Although, it was a significant upgrade from UPF 1.0 and provided lot more automation capabilities, the adoption has been slow over the years. Lack of clarity and preciseness on semantics didn't help the adoption further. The P1801-2013 (aka UPF 2.1) [2], recently released in May 2013, has addressed the issues of the previous version and also introduced new concepts to provide greater precision, accuracy and fidelity of power intent expression.

D. Why Migrate To UPF 2.1

The new UPF 2.1 has taken leaps in addressing most of the verification challenges that the former versions of UPF failed to address. This paper highlights some of these verification challenges and demonstrates the capability of UPF 2.1 by relevant examples. It also highlights the limitations with the earlier UPF versions, along with providing migration tips which will help users easily migrate to UPF 2.1 standard.

II. Basic Concepts of UPF

Some of the important concepts and terminology used in power intent specification are the following:

- Power domain: A collection of HDL module instances and/or library cells that are treated as a group for power management purposes. The instances of a power domain typically, but do not always, share a primary supply set and typically are all in the same power state at a given time. This group of instances is referred to as the extent of a power domain.
- Power state: The state of a supply net, supply port, supply set, or power domain. It is an abstract representation of the voltage and current characteristics of a power supply, and also an abstract representation of the operating mode of the elements of a power domain or of a module instance (e.g., on, off, sleep).
- Power state table (PST): A table that captures the legal combinations of power states for a set of supply ports and/or supply nets.
- Isolation Cell: An instance that passes logic values during normal mode operation and clamps its output to some specified logic value when a control signal is asserted. It is required when the driving logic supply is switched off while the receiving logic supply is still on.
- Level Shifter: An instance that translates signal values from an input voltage swing to a different output voltage swing.
- Retention: Enhanced functionality associated with selected sequential elements or a memory such that memory values can be preserved during the power-down state of the primary supplies.
- Repeaters: If the distance between driver and receiver is long, special buffers may be required to boost the strength of the signal, or to ensure that it stabilizes within the required time. These buffers are typically called repeaters.
- Supply net: an abstraction of a power rail.
- Supply set: an abstraction of a collection of supply nets that in aggregate provide all the supply connections required by a given logic element. The individual nets are referred to as functions of a supply set.
- Driver supply (source supply): the supply set providing power to the logic that is the ultimate source (driver) of a net.
- Receiver supply (sink supply): the supply set providing power to the logic that is the ultimate sink (load) of a net.
- Soft IP: a synthesizable module in HDL such as SystemVerilog or VHDL. It is designed to be implemented using logic synthesis and place-and-route tools.
- Hard IP: an IP which is pre-implemented and has power management already built into it. These IPs are already verified for power management at the IP level.

III. Verification Challenges

A. *Repeater insertion*

In typical designs, it is possible that there are signal crossings that span across several power domains, some of which can be switchable. These crossings require special buffers to boost the strength of the signal. These buffers, typically called repeaters, can be placed anywhere along the path depending upon the fanout and load requirements. It becomes important to use the correct supplies for these repeaters as incorrect switching of supplies may affect the functionality of the design.

The repeaters are typically inserted automatically by implementation tools late in the design flow. Due to late insertion of these buffers, the verification process done at an earlier stage is ignorant of their existence and hence cannot verify the intent properly. In some cases, designers are required to direct tools to pick specific supplies for the insertion of these repeaters according to their unique requirements.

1) UPF 2.0 Specification

UPF 2.0 has some capability to allow users to specify the supply of repeaters by specifying the `-repeater_supply` attribute.

UPF Code

```
set_port_attributes -domain pd_sw \
  -ports $DOMAIN_PORTS \
  -repeater_supply always_on_ss
```

a) Limitations

Due to the lack of proper semantics in UPF for this attribute, the option is not widely used. It only restricts insertion of buffers at the outputs and doesn't clearly define the placement of the buffers. It fails to define the interaction of the buffers with other power management cells like isolation or level shifter cells. Moreover, it fails to provide sufficient information for tools to perform verification and implementation of repeaters.

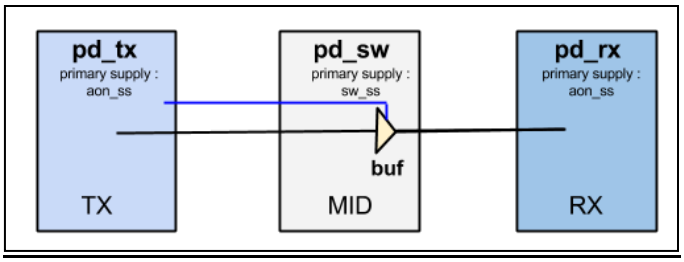
These limitations have resulted in users relying on proprietary commands or semantics to express the intent. This causes some significant problems to verification tools to mimic the behavior and achieve effective verification early in the design flow.

2) UPF 2.1 Solution

UPF 2.1 has introduced a new strategy command, `set_repeater`, similar to existing strategy commands for isolation and level shifter. This allows the user to define a strategy for the insertion of repeaters with greater control over placement. The repeaters inserted for these commands act as new source/sink for the determination of isolation and level shifter strategies hence providing a well-defined semantics for the effect of placement of repeater cells.

Similar to other strategies, the set_repeater command inherits the well-known semantics like filters, precedence rules and predictable placement of cells. There is also an ability to provide existing repeater instances in the same way as other existing special cells like an isolation cell.

Design Example



A signal originates in transmitter domain (`pd_tx`), span across switchable domain (`pd_sw`) and terminates in receiver domain (`pd_rx`). To boost the signal strength, repeaters are required to be inserted in switchable domain `pd_sw`

UPF Code

```
create_power_domain pd_tx \
  -elements {tx} \
  -supply {primary aon_ss}
create_power_domain pd_sw \
  -elements {mid} \
  -supply {primary sw_ss}
create_power_domain pd_rx \
  -elements {rx} \
  -supply {primary aon_ss}
# Insert repeaters powered by always on supply
 set_repeater rep_sw \
  -domain pd_sw \
  -repeater_supply_set pd_tx.primary \
  -source pd_tx.primary \
  -sink pd_rx.primary
```

a) Verification Impact

With the dedicated strategy for the insertion of repeaters, verification tools can easily understand the location and properties of repeaters. This will help them to mimic their behavior in simulation to catch errors related to incorrect supplies and placement of these repeater cells.

3) Migration Tips

Users who were using proprietary commands need to translate them to the appropriate `set_repeater` strategies for consistent behavior across various tools.

B. Modeling soft IP power management constraints

A large number of SoC designs use configurable soft IPs. These IPs typically define constraints related to power management which should be followed for the proper functioning of the IP. Some of these constraints are following:

- Certain regions should always belong to specific power domains and should be powered by the same supply of the power domain.
- The isolation cells inserted in the IP should follow specific clamping requirements as determined by the IP provider.
- If the IP has retention, the critical registers should always be retained during the power down period.

It becomes important for an IP integrator to ensure that these constraints are not violated when the soft IP is integrated into their system and configured according to their requirement. Hence, it is important for verification tools to validate the constraints of the soft IPs and catch any scenarios that fail to honor these constraints. As a result, the power intent should be able to express these constraints to enable tools to interpret and validate them.

1) UPF 2.0 Specification

UPF 2.0 can be used to express the constraints for the soft IP as demonstrated in the DVCon 2012 paper titled" Low Power SoC Verification: IP Reuse and Hierarchical Composition using UPF"[3].

UPF Code

```
#---------------------
#soft_ip.upf
#---------------------

#Power Domains for Soft IP
create_power_domain pd_softIP \
  -include_scope \
  -supply { cpu_ss } \
  -supply { mem_ss  }

#Isolation Constraints
set_port_attributes -ports $PORTS \
  -clamp_value 1

#Retention Constraints
set_retention_elements critical_regs \
  -elements { reg_a reg_b } \
  -retention_purpose required

#... Other Constraints ...

#---------------------
#soc.upf
#---------------------

# Load UPF of Soft IP
load_upf soft_ip.upf \
  -scope softIPinst

# Connect Supplies
associate_supply_set pd_SoC.primary \
  -handle softIPinst/pd_softIP.cpu_ss
associate_supply_set pd_SoC.mem_ss \
  -handle softIPinst/pd_softIP.mem_ss

# Update Supply Constraints
add_power_state pd_SoC.primary ..

# Connect Logic Controls
connect_logic_net ...

#---------------------
```

a) Limitations

Although, UPF 2.0 can be used to model the constraints for the IP but it still lacks clear semantics in some areas and this can easily be missed by the verification semantics.

The regions within the power domain can easily be carved out and added into another power domain. The semantics of UPF provide no mechanism to catch such a scenario.

UPF Code

```
soc.upf
---------------
load_upf soft_ip.upf \
  -scope softIPinst

#.... Other UPF Commands ...

#Potential Problem
create_power_domain pd_other \
  -elements { softIPinst/child }
```

It could be possible that an IP integrator has accidently configured the IP in such a way that an element inside the soft IP (`softIPinst/child`) is added into another power domain (`pd_other`) powered by a different supply (`other_ss`) and the tools will not be able to catch such a scenario.

The semantics of `set_retention_elements` are not properly defined for tools to interpret it and validate the retention constraint. Although, there is some indication that there should be error checking to ensure retention behavior but lack of a proper description makes it difficult for tools to have consistent behavior.

2) UPF 2.1 solution

UPF 2.1 introduced the concept of atomic power domains (by using command `create_power_domain -atomic`). It allows the user to add constraints to the IP that once an atomic power domain is created it cannot be further split during implementation. To enforce this, atomic power domains have the highest precedence and are created first before non-atomic power domains.

UPF 2.1 has also clarified the semantics of the command `set_retention_elements`. It allows users to create an atomic list of critical registers all of which needs to be retained, if the IP has retention capability.

UPF Code

```
#---------------
#soft_ip.upf
#---------------

#Power Domains for Soft IP
create_power_domain pd_softIP \
  -elements {.} \
  -supply { cpu_ss } \
  -supply { mem_ss  } \
  -atomic

#Isolation Constraints
set_port_attributes \
  -ports $PORTS \
  -clamp_value 1

#Retention Constraints
set_retention_elements critical_regs \
  -elements { reg_a reg_b } \
  -retention_purpose required

#... Other Constraints ...

#---------------------
#soc.upf
#---------------------

#.... UPF Commands ...

# Load UPF of Soft IP
load_upf soft_ip.upf \
  -scope softIPinst

# Connect Supplies
associate_supply_set pd_SoC.primary \
  -handle softIPinst/pd_softIP.cpu_ss
associate_supply_set pd_SoC.mem_ss \
  -handle softIPinst/pd_softIP.mem_ss

# Update Supply Constraints
add_power_state pd_SoC.primary ..

# Connect Logic Controls
connect_logic_net ...


#.... Other UPF Commands ...
#ERROR
~~create_power_domain pd_other \~~
~~  -elements { softIPinst/child }~~

# Adding retention only on reg_a
# ERROR: Retention not added on reg_b
~~set_retention ret_soft \~~
~~  -domain softIPinst/pd_softIP~~
~~  -elements { softIPinst/reg_a }~~

#---------------
```

a) Verification Impact

If `pd_softIP` has been defined as atomic, then adding any child instances that belong to it into some other power domain will cause errors in UPF processing. This avoids any scope of accidently carving out of regions within any soft IP atomic power domains. If only some of the registers of `set_retention_elements` list are retained and rest of the registers are not retained, then it will cause an error in UPF processing.

3) Migration Tips

Users have to modify the existing power domains of the IP and make them atomic. If the IP provider wants to specify additional power domains within an atomic power domain, they have to use `-exclude_elements` during the creation of atomic power domains. The changes related to `set_retention_elements` are mostly in terms of clarification so there isn't any need to change anything if they are already using this command in their UPF 2.0 code.

However, if they are using `-expand` option of the command then they need to remove it as it has been marked as deprecated in the 2.1 standard. Removing this option will not impact their power intent.

C. Modeling of Hard IP

In some cases, the SoC directly reuses an IP that is pre-implemented and the power management is already built into it. These IPs, called hard IPs, are already verified for power management at the IP level. It becomes important for the IP integrator to ensure that the IP has been connected to proper supplies and that the boundary of IP are properly protected with respect to the environment in which the IP is present.

Although the power intent of Hard IP is already verified, it is important for the SoC environment to be aware of the power management of hard IP. This will ensure that verification tools can validate the power management in the SoC environment and can catch issues related to it early in the design flow. The following information is necessary for the integration of hard IP:

- Information about related supplies of boundary ports of hard IP
- Information about external and internal supplies of the hard IP and their characteristics
- Information about isolation/level shifter cells already implemented for the hard IP and the supplies.
- Information about system power states of the hard IP

1) UPF 2.0 Specification

UPF 2.0 has capabilities to specify the power intent of a hard IP.

UPF Code

```
#---------------
#hard_ip.upf
#---------------
#Power Domains for Hard IP
create_power_domain pd_hardIP \
  -include_scope \
  -supply { backup_ssh } \
  -supply { primary }

#Related Supply Constraints
set_port_attributes -domain pd_hardIP \
  -applies_to outputs \
  -driver_supply pd_hardIP.primary
set_port_attributes -ports portA \
  -driver_supply pd_hardIP.backup_ssh
set_port_attributes -domain pd_hardIP \
  -applies_to inputs \
  -receiver_supply pd_hardIP.primary

#Internal switchable supply
create_power_switch ...
# Isolation/level shifter and retention cells
set_isolation ...
set_level_shifter ...
set_retention ...

# System states for hard IP
add_power_state pd_hardIP ...
```

```
#... Other Constraints ...

#---------------------
#soc.upf
#---------------------

# Load Hard IP UPF
load_upf hard_ip.upf \
  -scope hardIPinst

# Connect Supplies
associate_supply_set pd_SoC.primary \
  -handle hardIPinst/pd_hardIP.primary
associate_supply_set pd_SoC.backup \
  -handle hardIPinst/pd_hardIP.backup

# Update Supply Constraints
add_power_state pd_SoC.primary ..

# Connect Logic Controls
connect_logic_net ...

#---------------
```

a) Limitations

The UPF specification is directive in nature as it directs verification and implementation tools to apply the necessary changes required for power management. Since, power management for the hard IP is already implemented; the UPF specification should only be used to describe the power intent of hard IP for validating in the SoC environment. This implies that in the presence of an UPF for hard IP, tools should be able to detect the hard IP instance and ensure that the UPF specification doesn't end up having re-implemented.

2) UPF 2.1 Solution

In order to create a well-defined boundary for hard IP power intent, new dedicated commands (`begin_power_model`, `end_power_model`) have been added in UPF 2.1 standard.

Along with these commands a new command (`apply_power_model`) to provide easy association of the model to design has also been provided. The semantics of UPF commands within `begin_power_model` and `end_power_model` has been explicitly changed to be descriptive in nature, implying that implementation will automatically ignore the commands within a power model.

UPF Code

```
#---------------
#hard_ip.upf
#---------------

# Power Model for Hard IP
begin_power_model hardMacro

#Power Domains for Hard IP
create_power_domain pd_hardIP \
  -include_scope \
  -supply { backup_ssh } \
  -supply { primary }

#Related Supply Constraints
set_port_attributes -domain pd_hardIP \
```

```
  -applies_to outputs \
  -driver_supply pd_hardIP.primary
set_port_attributes -ports portA \
  -driver_supply pd_hardIP.backup_ssh
set_port_attributes -domain pd_hardIP \
  -applies_to inputs \
  -receiver_supply pd_hardIP.primary

#Retention Constraints
set_retention_elements critical_regs \
  -elements { reg_a reg_b } \
  -retention_purpose required

#Internal switchable supply
create_power_switch
# Isolation/level shifter and retention cells
set_isolation ...
set_level_shifter ...
set_retention ...

# System states for hard IP
add_power_state pd_hardIP ...

# ... Other Constraints ...

end_power_model

#---------------------
#soc.upf
#---------------------
# Load the Power Models.
load_upf hard_ip.upf

# Apply the Power model for hard Macro
# and connect supplies
apply_power_model hardMacro \
  -supply_map {
    { pd_hardIP.primary pd_SoC.primary } \
    { pd_hardIP.backup_ssh pd_SoC.backup_ssh }
\
  }

# Update Supply Constraints
add_power_state pd_SoC.primary ..

# Connect Logic Controls
connect_logic_net ...

#---------------
```

a) Verification Impact

Verification tools can now understand the power intent of hard macros and catch any integration issues by validating the constraints.

3) Migration Tips

If users are already following the methodology described in DVCon 2012 paper titled "Low Power SoC Verification: IP Reuse and Hierarchical Composition using UPF" [3] then they can simply add `begin/end_power model` at the beginning and end of interface UPF.

If the information about hard IP is captured in liberty or other formats then they need to be translated into equivalent UPF commands and added into the power model corresponding to the IP.

D. Isolation cells at Hard Macro boundary

UPF allows insertion of power aware cells only at the power domain boundaries.

1) UPF 2.0 Specification

In order to apply isolation/level-shifter/buffer cells at a Hard IP boundary, these IPs need to be explicitly added as extent to a power domain.

UPF Code

```
# Power domain for DUT
create_power_domain pd_dut \
  -include_scope \
  -supply {primary always_on_ss}

# Hard IP Modeling: Boundary constraints
set_scope hard_ip

# Some output pins have switched supply
# as driver supply
set_port_attributes \
  -ports {pin1 pin2}\
  -driver_supply {sw_ss}

# Some output pins have always on supply
# as driver supply
set_port_attributes \
  -ports { pin3 pin4}\
  -driver_supply {always_on_ss}

# Just to isolate the boundary of HardIP,
# will have to create power domain for
# Hard IP
create_power_domain pd_hardIP \
  -include_scope
set_isolation iso \
  -domain pd_hardIP \
  -applies_to both
```

a) Limitations

Creating additional power domains for Hard IPs leads to unnecessary complexities and burden on the designer to ensure UPF is matching the intent. They need to explicitly put `-no_isolation` on ports that share the same supply or define strategies in such a way that redundant isolation cells are not inserted because of power domain boundary. The implementation tools need to tackle such power domains in a special way so that it doesn't cause problems in implementation.

UPF Code

```
# Redundant isolation placed on pin3 and
# pin4 as source and sink both are
# always_on_ss

# Specify -no_isolation on pin3 and pin4
set_isolation iso1 -domain pd_hardIP  \
  -elements {pin3 pin4}
  -no_isolation
```

2) UPF 2.1 solution

UPF 2.1 no longer requires user to add Hard IPs to extent of power domain just for creating domain boundary. According to UPF 2.1, lower boundary of a domain consists of the HighConn side of each port on each child instance that is in some other power domain or is a port of a macro cell instance that is powered differently from the rest of the domain.

UPF Code

```
# No need for domain boundary for hardIP
# Apply isolation on lower boundary of
# pd_dut

set_isolation dut_iso \
  -domain pd_dut \
  -elements {hard_ip}

# Pins pin1 and pin2 of hard_ip
# constitutes lower boundary of pd_dut as
# these are powered by different supply
# sw_ss

# Isolation not applied on pin3 and pin4
# as these are powered by always_on_ss
# which is also primary supply of pd_dut
```

a) Verification Impact

This takes the burden from user to create unnecessary power domains. Users will now have to use fewer commands to express the power intent, thus reducing the burden on verification and minimize the bugs introduced by incorrect power intent.

3) Migration Tips

Some IPs may already have isolation capability inside them. Users will have to take extra care when integrating such IPs in a SOC. As these Hard IPs now constitute lower boundary of a domain, it could potentially lead to redundant isolation at boundary of macro cell. To avoid such redundant isolation cells, user will need to use define the strategies carefully by using `-source/sink` filters or explicitly specify `-no_isolation` on lower boundary of hard macro.

E. Verifying supply constraints

It could happen that the power intent which was successfully verified at the RTL stage was rejected by backend tools as it did not meet the constraints of power rails availability at the backend. This is because certain cells got placed at a location by UPF where the supply rail to power these cells is missing. This causes several problems to backend tools where designers are forced to either make costly ECOs or do some significant modifications in power intent to make the power intent successfully pass through backend tools.

1) UPF 2.0 Specification

UPF 2.0 doesn't have any capability to express these constraints. As a result, users have to rely on proprietary ways to specify this constraint which causes portability issues.

UPF Code

```
create_power_domain pd_aon \
  -elements {feedthru_inst}
# pd_aon doesnot have availability of ss_sw
set_port_attributes \
  -ports {feedthru_inst/out} \
  -repeater_supply ss_sw
```

a) Limitations

Lack of constraint specification in UPF makes it difficult for verification tools to do constraint checking early in the design cycle.

Repeater cell powered by switched supply `sw_ss` got placed in an always on domain `pd_aon` where this switched supply is not available. Such issues are caught only at later stages during implementation.

2) UPF 2.1 solution

UPF 2.1 has introduced a new concept of supply availability to specify the constraints about which supplies are available in a power domain.

The predefined supply set handles of a power domain and the supply sets identified by options of the strategies (`set_isolation`, `set_retention`, `set_repeater`, `set_level_shifter`) associated with the domain are referred to as the locally available supplies of that domain. These locally available supplies can be used by tools to power cells inserted into a domain.

In addition to the locally available supplies, the UPF command

`create_power_domain -available_supplies`

specifies whether any additional supplies are also available for use, and if so, which supplies are available for use by tools to power cells inserted into the power domain.

UPF Code

```
create_power_domain pd_aon \
  -elements {feedthru_inst} \
  -available_supply {ss_sw}
set_repeater -domain pd_aon \
  -elements {feedthru_inst/out} \
  -repeater_supply {ss_sw}
```

a) Verification Impact

Verification tools can do constraint checking and ensure that supplies used to power the inserted cells are available in the domain where these cells get inserted. Any incorrect usage is caught immediately at RTL stage itself.

3) Migration Tips

If users are using proprietary syntax to express the same information, they need to translate it to the new UPF 2.1 syntax.

F. Supply Equivalence

Power management cells like isolation and level shifter are often dependent upon the supplies powering the logic driving and receiving a signal.

Isolation is required when the source supply is switched off while the sink supply is still on. Similarly, level shifters are required when source and sink supplies operate at different voltage levels when they are on.

As a result, the source and sink supplies are often used as filters in selecting the ports needed for isolation and level

shifters. This requires tools to match the source and sink supplies with descriptions present in power intent.

For the above usage of selection of ports based on `-source/sink` supply filters; proper matching of supplies needs to be done.

1) UPF 2.0 Specification

UPF 2.0 didn't explicitly define the definition of supply equivalence. This resulted in inconsistent interpretation and discrepancies between different tools over the definition of matching of supplies.

UPF Code

```
#---------------
# UPF 2.0
#---------------
# ss1 and ss2 are equivalent
create_supply_set ss1 \
  -function { power vdd1 } \
  -function { ground vss }
create_supply_set ss2 \
  -function { power vdd1 } \
  -function { ground vss }
create_supply_set ss3 \
  -function { power vdd2 } \
  -function { ground vss }

create_power_domain pd1 \
  -supply {primary ss1}
create_power_domain pd2 \
  -supply {primary ss2}
create_power_domain pd3 \
  -supply {primary ss3}

# Will match only pd1.primary as
# no equivalence semantics defined
# in UPF 2.0
set_isolation iso \
  -sink ss1 \
  # ... Other options ...

# set_level_shifter only allows \
# power domains in -source/-sink
# Will match only pd1 as sink
set_level_shifter ls \
  -domain pd_Src \
  -sink pd1 \
  # ... Other options ...
```

a) Limitations

Some tools do not honor supply equivalence and match the supplies only if they are identical. This may result in incomplete matching and thus may miss some power aware cells when using -source/-sink filters.

For the UPF strategies using (`set_isolation -diff_supply_only`) some tools may insert redundant cells even if source/sink supplies were equivalent as they didn't consider supply equivalence; thereby these redundant cells taking extra area on the chip.

2) UPF 2.1 solution

UPF 2.1 has explicitly defined the rules for matching of supplies by introducing the concept of supply equivalence.

Two supply ports or nets are said to be electrically equivalent if the two objects are electrically connected somewhere. Similarly, two supply sets are electrically equivalent if the two are directly associated with each other using the UPF command (`associate_supply_set`) or all the corresponding functions and their associated supply nets are respectively equivalent.

In certain cases the electrical connection may not be evident in the design. UPF 2.1 has defined a new command (`set_equivalent`) to explicitly state the supply equivalence.

UPF Code

```
#---------------
# UPF 2.1
#---------------
create_supply_set ss1
create_supply_set ss2
create_supply_set ss3

set_equivalent \
  -sets { ss1 ss2 }

create_power_domain pd1 \
  -supply {primary ss1}
create_power_domain pd2 \
  -supply {primary ss2}
create_power_domain pd3 \
  -supply {primary ss3}

# Will match both pd1.primary
# and pd2.primary
set_isolation iso \
  -sink ss1 \
  # ... Other options ...

# set_level_shifter extended to
# allow both power domains or
# supply sets.
# Will match both pd1 and pd2
# as pd1 will default to pd1.primary
set_level_shifter ls \
  -domain pd_Src \
  -sink pd1 \
  # ... Other options ...

# Will match only pd1
set_level_shifter ls \
  -domain pd_Src \
  -sink pd1.primary \
  -use_equivalence FALSE \
  # ... Other options ...

#---------------
```

a) Verification Impact

Power management strategy commands like `set_isolation`, `set_level_shifter` and `set_repeater` will by default use supply equivalence in determining which ports are to be isolated when filtering is specified with a `-sink/-source` or `-diff_supply_only` filters.

For the cases where the user wants the supply matching to be done only when these are identical, they can specify it by setting `-use_equivalence` to be `FALSE`.

These clarified semantics will result in consistent behavior across various tools and will improve interoperability between tools.

3) Migration Tips

The default behavior as per UPF 2.0 was to match the supply only if the two are same, however this has been changed in UPF 2.1. In the later version, the two supplies will match if these are electrically equivalent.

So in certain cases when the user wanted the power aware cells insertion only for identical supplies, they will have to explicitly use `-use_equivalence FALSE`.

G. Interactions of power management strategies

In a large SoC design comprising of several power domains, each with a number of strategies defined on them, can often result in a scenario where multiple strategies affect a domain crossing. This can result in the placement of multiple isolation/level shifter and repeater cells in the path. The relative ordering of these cells becomes critical as functional and electrical bugs may be introduced if the placement is incorrect.

1) UPF 2.0 Specification

There are no clear semantics defined to address the relative placement of power management cells. In such situations, it is left to the tools to make their own interpretation of relative placements which can lead to discrepancies between verification and implementation.

UPF Code

```
create_power_domain pd_tx \
       -elements {tx} \
       -supply {primary sw_ss}
create_power_domain pd_sw \
       -elements {mid} \
       -supply {primary sw_ss}
create_power_domain pd_rx \
       -elements {rx} \
       -supply {primary always_on_ss}
set_port_attributes -domain pd_sw \
       -applies_to outputs \
       -repeater_supply always_on_ss
set_isolation iso_tx -domain pd_tx \
       -sink pd_rx.primary \
       -applies_to outputs
set_isolation iso_rx -domain pd_rx \
       -source pd_tx.primary \
       -applies_to inputs
```

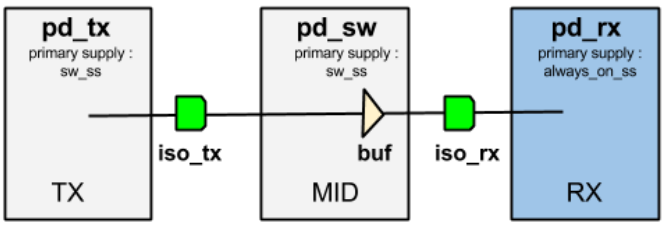


Image1

a) Limitations

Different tools have interpreted this in different ways, causing the problems related to interoperability.

2) UPF 2.1 Solution

The new standard has clearly defined how the power management strategies interact with each other and has defined a precedence order in which these strategies apply.

Strategies are implemented in the following order: 1) retention strategies, 2) repeater strategies, 3) isolation strategies, and 4) level-shifter strategies. Each strategy may affect the driving or receiving supply of the port and thus affect the –source/sink filters of a subsequently applied strategy.

```
#Same upf code as specified in UPF 2.0 section
#iso_rx does not get placed as source is now
changed to always_on_ss
```

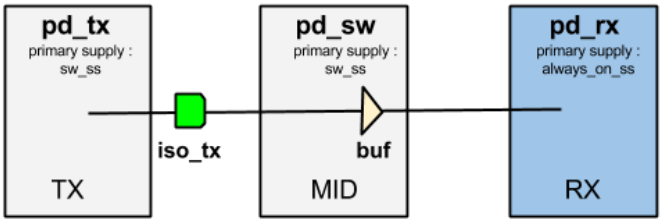


Image2

a) Verification Impact

Clarified semantics will result in consistent behavior across various tools and will improve interoperability

3) Migration Tips

Users may have to update their strategies to account for the new semantics. This will help them get consistent behavior across various tools and avoid the chance of mismatch between verification and implementation.

H. Power States

Power states play a significant role in describing the power intent. There will be a tremendous impact if the power states are not defined properly as they are used for both verification and implementation of power management. The verification tools use it for checking the power management structures whereas implementation tools use it to insert the power management logic.

1) UPF 2.0 Specification

UPF 2.0 defines two separate styles of capturing power state information.

a) Power State Tables (PST)

Power State Tables (PST) are inherited from UPF 1.0 which allowed specification of power states in terms of supply nets/ports and their possible combinations of values. It is a tabular representation of the possible state combinations for the given supplies. The tabular representation makes it easier for users to comprehend the power state dependencies and possibilities. However, this style has several limitations which prompted UPF 2.0 to introduce a new way of representing power state information.

UPF Code

```
#---------------
PST Example
#---------------

# Port States
add_port_state vdd1 \
      -state { OFF off } \
      -state { ON_1d0V  1.0 }

add_port_state vdd2 \
      -state { OFF off } \
      -state { ON_1d0V  1.0 }

# PST Definition
create_pst PST_IP \
  -supplies     { vdd1    vdd2    }
add_pst_state -pst PST_IP \
  IP_ONH -state { ON_1d0V ON_1d0V }
add_pst_state -pst PST_IP \
  IP_STB -state { ON_1d0V OFF     }
add_pst_state -pst PST_IP \
  IP_OFF -state { OFF     OFF     }
#---------------
```

Limitations

The dependency on supply port/nets makes it difficult for designers to capture state information early in the design process where the supply information is not available.

Tabular representation causes an explosion of states in large designs that have a large number of supplies and states defined on them. This forces users to split the PST information into multiple smaller PSTs that are easier to manage but at the cost of a loss of information about all state combinations.

The lack of hierarchical composition capability in PSTs forces the user to rely on tools to automatically merge the PSTs together to compose a larger table for the whole system to get required information. The UPF LRM doesn't define any rules or semantics for merging the PST and hence tools have to depend on proprietary algorithms for the merging. This causes incompatibility across tools aggravating the verification problem.

b) Power States

To overcome the shortcomings of PST representation, UPF 2.0 introduced a completely new style of expressing power state information via the `add_power_state` command. This allows users to define power states on supply sets and power domains.

The `add_power_state` command requires users to express power states in terms of boolean expressions via `-logic_expr` and `-supply_expr` switches. Although, this is very powerful in capturing more complex relationships including hierarchical dependencies, it has the potential of causing problems if not used properly. The absence of sufficient restrictions and guidelines in the LRM implies that tools cannot catch improper usage. The hierarchical composition capability is built right into the command itself and promises to reduce the state explosion. However, the lack of methodology and semantics imply that users are slow in adopting `add_power_state` for representing power states.

UPF Code

```
#---------------
# Power State Example
#---------------

# Power States on Supply Sets
add_power_state pd_IP.primary \
  -state IP_SS_ON { \
    -supply_expr { (pd_IP.primary.power ==
{FULL_ON, 1.0}) \
     && ( pd_IP.primary.ground == {FULL_ON,
0.0 } ) \
    } \
  } \
  -state IP_SS_OFF { \
    -supply_expr { pd_IP.primary.power == OFF
} \
  }

add_power_state pd_IP.gpu_ssh \
  -state GPU_SS_ON { \
    -supply_expr { (pd_IP.gpu_ssh.power ==
{FULL_ON, 1.0}) \
     && ( pd_IP.gpu_ssh.ground == {FULL_ON,
0.0 } ) \
    } \
  } \
  -state GPU_SS_OFF { \
    -supply_expr { pd_IP.gpu_ssh.power == OFF
} \
  }

# System Power States on Power Domain
add_power_state pd_IP \
  -state PD_IP_ONH { \
    -logic_expr { (pd_IP.primary == IP_SS_ON )
\
     && ( pd_IP.gpu_ssh == IP_GPU_SS_ON_1d0V )
\
     } \
    } \
  -state PD_IP_STB { \
    -logic_expr { ( pd_IP.primary == IP_SS_ON
) \
    && ( pd_IP.gpu_ssh == IP_GPU_SS_OFF) \
    } \
  } \
  -state PD_IP_OFF { \
    -logic_expr { (pd_IP.primary == IP_SS_OFF
) \
    && ( pd_IP.gpu_ssh == IP_GPU_SS_OFF ) \
    } \
  }
#---------------
```

Limitations

UPF 2.0 doesn't define proper semantics about how the states are handled when supply sets are associated with other supply sets or handles. UPF 2.0 does not restrict the transfer of power states in the supply set associations. So the power states defined on one power domain may incorrectly affect some other power domains which are associated with the same supply set.

Insufficient restrictions may lead to circular dependence of power states and also makes these difficult for methodical usage.

UPF Code

```
add_power_state PD.SS -state OFF {\
    -supply_expr {power == OFF} \
    -logic_expr {PD==OFF}\
}
add_power_state PD -state OFF {\
    -logic_expr {PD.SS == OFF}\
}
```

2) UPF 2.1 Solution

UPF 2.1 has clarified a lot of semantics to enable greater adoption of the `add_power_state` command. The standard has marked PSTs as legacy – implying that the command, although present in standard, is only present for backward compatibility and is not recommended for future use. The legacy commands will not be considered for future extensions of the standard hence their use should be discouraged and users are advised to migrate to `add_power_state` command. The following are the clarifications and new additions to this command:

- Clarified that supply set handles are local supply sets. Power states do not get transferred. Power states added to supply set handles are a property of supply set handles and not its associated supply set.
- Added `-supply`, `-domain` and `-complete` to `add_power_state`
- Addition of another simstate "`CORRUPT_STATE_ON_ACTIVITY`"

A number of restrictions have also been imposed on the `add_power_state` command as follows:

- For power states added on supplies
  - `-supply_expr` can refer to supply ports/nets or its own functions. It cannot refer to functions of another supply set;
  - `-logic_expr` can refer to logic ports/nets, interval functions and its own power states however it cannot refer to functions of a supply set, power states of another supply set or power states of a domain
- For power states added on power domains:
  - It cannot have a `-supply_expr`.
  - `-logic_expr` can refer to logic ports, logic nets, interval functions, power states of supply sets or supply set handles, or power states of other power domains.
  - It is an error if `-logic_expr` refers to supply ports, supply nets, or functions of a supply set or supply set handle.

UPF Code

```
#---------------
# UPF 2.1 Example
#---------------

# Power States on Supply Sets
add_power_state pd_IP.primary \
  -supply \
  -state IP_SS_ON { \
    -supply_expr { (power == {FULL_ON, 1.0}) \
    && (ground == {FULL_ON, 0.0 } ) \
    } \
  } \
  -state IP_SS_OFF { \
    -supply_expr {power == OFF } \
  }

add_power_state pd_IP.gpu_ssh \
  -supply \
  -state GPU_SS_ON { \
    -supply_expr { power == {FULL_ON, 1.0}) \
    && (ground == {FULL_ON, 0.0 } ) \
    } \
  } \
  -state GPU_SS_OFF { \
    -supply_expr {power == OFF } \
  }

# System Power States on Power Domain
add_power_state pd_IP \
  -domain -complete \
  -state PD_IP_ONH { \
    -logic_expr {(primary==IP_SS_ON) \
    &&(gpu_ssh == IP_GPU_SS_ON) \
    } \
  } \
  -state PD_IP_STB { \
    -logic_expr { ( primary == IP_SS_ON ) \
    && ( gpu_ssh == IP_GPU_SS_OFF) \
    } \
  } \
  -state PD_IP_OFF { \
    -logic_expr { (primary == IP_SS_OFF ) \
    && ( gpu_ssh == IP_GPU_SS_OFF ) \
    } \
  }
#---------------
```

a) Verification Impact

The clarified semantics helps the user define power states in a much better way and in less amount of code avoiding creation of explicit supply sets. Proper restrictions promote better methodology for power intent modeling.

3) Migration Tips

As the states added on supply set do not get transferred to its handles, users will need to explicitly define power states on supply set handles if they are different than default states.

Due to various restrictions added in the 2.1, the user may have to modify the power state definitions if it is not complying with the restrictions.

I. Retention Semantics

Retention registers come in various types depending on how the retained value is stored and retrieved. There are at least two types of retention registers, as follows:

a) Balloon-style retention: In a balloon-style retention register, the retained value is held in an additional latch, often called the balloon latch. In this case, the balloon element is not in the functional data-path of the register.

b) Master/slave-alive retention: In a master/slave-alive retention register, the retained value is held in the master or slave latch. In this case, the retention element is in the functional data-path of the register.

1) UPF 2.0 Specification

UPF 2.0 mainly defines the semantics of balloon-style retention.

UPF Code

```
#---------------
# UPF 2.0
#---------------
# Balloon Latch

set_retention ret_balloon \
  -domain pd \
  -save_signal { ret high } \
  -restore_signal { ret low } \
  # ... Other options ...

#---------------
# Master/Slave Alive Latch

set_retention ret_ms \
  -domain pd \
  -save_signal { ret high } \
  -restore_signal { ret low } \
  -retention_condition { ret }
  # ... Other options ...

# Explicitly specify a behavioral model
# to provide master-slave alive retention
# behavior.
map_retention_cell ret_ms \
  -domain pd \
  -lib_model_name master_slave_model { \
    -port CP UPF_GENERIC_CLOCK \
    -port D UPF_GENERIC_DATA \
    -port SET UPF_GENERIC_ASYNC_LOAD \
    -port VDDC pd.primary.power \
    -port VDDRET pd.default_retention.power \
    -port VSS pd.primary.ground  \
    -port RET ret \
  }
```

a) Limitations

When using this new style of retention, users have to explicitly use simulation models to enable early verification of retention or delay the verification to a later stage when retention registers have been inserted in the design.

2) UPF 2.1

UPF 2.1 provides special syntax and clear semantics to model master/slave alive retention flops.

For master-/slave alive implementations, the `-save_signal/-restore_signal` should not be specified in the `set_retention` command. The retention behavior of this style is specified through the `-retention_condition`

UPF Code

```
#---------------
# UPF 2.1
#---------------
# Balloon Latch

set_retention ret_balloon \
  -domain pd \
  -save_signal { ret high } \
  -restore_signal { ret low }
  # ... Other options ...

#---------------
# Master/Slave Alive Latch
set_retention ret_ms \
  -domain pd \
  -retention_condition { ret }
  # ... Other options ...
#---------------
```

a) Verification Impact

Verification tools can identify the retention registers that are behaving as master/slave and can do an early verification because of the pre-defined behavior for them present in the UPF LRM. This ensures that verification at an early stage is much closer to the actual implementation improving the verification effectiveness.

3) Migration Tips

Users will need to change proprietary uses to new UPF 2.1 commands.

J. Power Management Cell modeling commands

For a complete and accurate verification of power intent, it is important to define the characteristics of the instances of power management cells that get inserted during the implementation of a power management, e.g. isolation, level-shfiting, always on cells, and retention cells.

1) UPF 2.0 Specification

UPF 2.0 doesn't define any dedicated commands that capture the specific details of the power management cells that get inserted. Although, the `set_port_attributes` command can be used to express some of the attributes, it still is not sufficient for completely modeling the power aware cell so that it can be properly implemented and verified.

a) Limitations

The user has to rely on library formats to specify this information to the verification tools. Sometimes libraries are out of date and incomplete in capturing power information which needs to be communicated to the verification tools.

Changing the library becomes difficult as there can be severe side-effects. Moreover some verification tools are unaware of library specification formats resulting in inconsistent interpretation of power intent and implementation

2) UPF 2.1 Solution

UPF 2.1 has introduced dedicated commands to model these power aware cells. Inspired by CPF, this command removes the dependency of UPF on external library formats thereby providing a comprehensive specification of power intent. These commands also provide capability to perform much more accurate verification of power aware cells that get inserted during the implementation. The semantics are defined

such that users can use these commands in conjunction with any library formats to model non-power aware behavior. The commands will automatically override any incorrect information related to power present in external library formats, without requiring users to go through the costly and risky process of modifying the library specification.

UPF Code

```
#---------------
# Power Management Cell Modeling
#---------------

# Isolation Cell Model
define_isolation_cell \
  -cells mbit_iso1 \
  -pin_groups { \
  { datain1 dataout1iso1 } \
  { datain2 dataout2 iso2 } \
  }
  -power VDD -ground VSS \
  -valid_location sink

# Level Shifter Cell Model
define_level_shifter_cell
  -cells LSHL \
  -input_voltage_range {{1.0 1.0}} \
  -output_voltage_range {{0.8 0.8}} \
  -direction high_to_low \
  -input_power_pin VH -ground G

# Switch Cell Model
define_power_switch_cell \
  -cells 2stage_switch \
  -stage_1_enable !I1 \
  -stage_1_output O1 \
  -stage_2_enable I2 \
  -stage_2_output !O2 \
  -type header

# Retention Cell Model
define_retention_cell \
  -cells My_Ret_Cell \
  -power VDDC -ground VSS \
  -power_switchable VDD \
  -save_check {!clk} \
  -restore_check {!clk} \
  -save_function {save negedge}
#---------------
```

a) Verification Impact

These commands can be leveraged by verification tools to perform accurate verification that closely matches the implementation results, without any need to depend on external library formats to get information.

3) Migration Tips

Users may need to capture the power intent present in library formats in terms of UPF commands if the verification tool doesn't interpret the library formats.

IV. BACKWARD COMPAPITIBILITY

One of the biggest challenges faced whenever a new standard comes out is ensuring backward compatibility and providing for the reuse of existing IPs. UPF 2.1 deprecated some of the UPF commands present in earlier versions and also changed the semantics and syntax of few existing commands and options. Although, these changes are intended to simplify the concepts in the standard, it has already caused some concerns among users regarding reuse of existing IPs.

1) Semantic compatibility

It can be possible that legacy UPF files of existing IPs may be semantically incompatible with UPF 2.1. Reusing such IP's in a UPF 2.1 the environment will require editing the UPF files to resolve the semantic differences.

Please refer to table-1 for the list of semantic difference's which are significant for verification.

2) Syntax compatibility

The existing IP whose UPF commands are syntactically compatible with the latest version will **not** require any change and will work as it is.

However the UPF of some IP's may not be syntax compatible with respect to UPF 2.1. In order to reuse such IP's in UPF 2.1 environment will require either of following:

- Verification tools will need to support syntax of all the UPF versions and interpret all of them correctly.
- User will need to modify the UPF to make it syntax compatible with latest UPF 2.1 standard.

a) Deprecated Commands

Refer to UPF 2.1 (IEEE_1801_2013), Annex D "**Replacing deprecated and legacy commands and options**" for list of deprecated and legacy commands. It is recommended to avoid using deprecated commands in the new code and replace the usage of deprecated commands in legacy UPF with corresponding UPF 2.1 commands.

b) Syntax Changes

Please refer to table-2 in the Appendix for the list of syntax differences between UPF 2.0 and UPF 2.1.

REFERENCES

[1] IEEE Std 1801™-2009 for Design and Verification of Low Power Integrated Circuits. IEEE Computer Society, 27 March 2009.

[2] IEEE Std 1801™-2013 for Design and Verification of Low Power Integrated Circuits. IEEE Computer Society, 29 May 2013.

[3] Amit Srivastava, Rudra Mukherjee, Erich Marschner, Chuck Seeley and Sorin Dobre : "Low Power SoC Verification: IP Reuse and Hierarchical Composition using UPF", DVCon 2012.

[4] Freddy Bembaron, Rudra Mukherjee, Amit Srivastava and Sachin Kakkar : "Low Power Verification Methodology using UPF.", DVCon 2009.

[5] Rudra Mukherjee, Amit Srivastava, Stephen Bailey: "Static and Formal Verification of Low Power Designs at RTL using UPF", DVCon 2008.

[6] Stephen Bailey, Amit Srivastava, Mark Gorrie, Rudra Mukherjee: "To Retain or Not to Retain: How do I verify the states of my low power design", DVCon 2008.

V. APPENDIX

Table 1 – Semantic differences between UPF 2.0 and UPF 2.1

S. No	Difference	UPF 2.0	UPF 2.1	Migration Guidelines	Reference
1	Default value of unconnected supply ports and nets	`UNDETERMINED`	`OFF`	Do not rely on default values, instead use UPF HDL package functions `supply_on / supply_off` to change the state and voltage of supply ports and nets.	IEEE 1801-2013 Section 9.2.1

2	Default supply of isolation cell	Isolation cell gets powered from the `default_isolation` supply set of the domain on which the isolation strategy is associated	Isolation cell gets powered from `default_isolation` supply set of the domain in which the isolation cell gets placed	If the strategy was defined in a way that `default_isolation` supply set was used, then it needs to be modified to list the supply set in `-isolation_supply_set` option of the strategy.
				e.g.
				`set_isolation iso \`
				`-domain pd \`
				`-isolation_supply_set\`
				`pd.default_isolation`
					IEEE 1801-2013 Section 6.41

3	Power states on supply set handle.	Supply set handles are considered as reference to the associated supply set. This implies that all properties of the supply set will be inherited by the supply set handle – including the power states defined on the associated supply set.	Supply set handles are local supply sets instead of a reference. Power states applied to a supply set handle are local and do not get transferred.	If power states are applied on only one supply set handle assuming it will get transferred to associated supply set, then this needs to be changed and explicitly applied to all the supply set handles as these will get not transferred.	Section III.H of this paper, "Power States"
4	HDL delays in assign statements	No information about delays. Interpretation is tool dependent	Delays are not considered as drivers/receivers	No change required if delay is not expected as driver/receiver. If delays are to be treated as driver/receiver, then the user needs to create a direct cell instantiation in HDL to be considered as drivers.	IEEE 1801-2013 Section 4.3.2
5	User defined functions in `create_supply_set`	Allowed to use user defined functions	User defined functions are removed from `create_supply_set`	Avoid using user defined function. If user defined functions are used in a supply set then it must be split into multiple supply sets by converting user defined functions into predefined functions.	IEEE 1801-2013 Section 6.22
6	Restriction in `add_power_state`	No restrictions in `-supply_expr` and `-logic_expr` of `add_power_state`	Restrictions added with respect to `-supply_expr` and `-logic_expr`	Follow the restrictions to make the UPF compatible	Section III.H of this paper, "Power States"

7	`set_simstate_behavior DISABLED …`	Was an error if supply port is not present in verification model	Allows simstate to be disabled even if supply port is not present in verification model	No change required	IEEE 1801-2013 Section 6.53
8	`set_design_top`	Used to specify root of the design. It is not clear that it accepts instance path or module name.	Used to specify module for which the subsequent UPF commands are written in the UPF file.	May need to change the root name from instance path to module name.	IEEE 1801-2013 Section 6.38
9	`set_isolation \ -force_isolation`	No clear semantics	It was clarified that without these options the tool may optimize away the isolation cells if there is no impact on functionality. Hence, if user intends to place them, they have to use `–force_isolation`	May need to add `–force_isolation` in some cases depending on need.	IEEE 1801-2013 Section 6.41

10	`upf_version`
	
	`load_upf –version`
	
	`load_upf_protected -version`
		The command allows tools to change the version as specified in the command and interpret the subsequent commands as per the specified version.	The command is now used for documentation purpose and tools will not change the version if it is specified to be lower than 2.1.	The UPF version control has been left to tool specific control. Hence, it is advised to avoid using commands and syntax that has difference with the latest UPF 2.1 standard.	IEEE 1801-2013 Section 6.54

11	Default value of `set_level_shifter \ -input_supply_set \ -output_supply_set`	The default is the primary supply set of the domain containing the source of the level-shifter input/output when the source is within the logic design starting at the design root.	The default is the supply of the logic driving the level-shifter input/output.	Avoid relying on default values.	IEEE 1801-2013 Section 6.43
12	Default value of `set_isolation –clamp_value`	The default value is “any”.	The default value if “0”.	Avoid relying on default values.	IEEE 1801-2013 Section 6.41

Table 2 – Syntax differences between UPF 2.0 and UPF 2.1

S.No	UPF 2.0	UPF 2.1	Details
1	Syntax Error: `set_isolation ~~-applies_to~~ -source -sink`	Valid Syntax: `set_isolation -applies_to -source -sink`	It gives more flexibility to select only inputs/outputs for source/sink considerations to avoid the placement of redundant isolation cells at both input and output of the domain for a feed-through path. Without this, user need to explicitly specify `-no_isolation` for one of the boundary ports.
2	`add_power_state pd` **`-state sleep {`** `-logic_expr { pd.primary == SLEEP } }`	`add_power_state pd -supply` **`-state { sleep`** `-supply_expr {pd.primary == SLEEP } }`	In UPF 2.0, the syntax was to specify state name outside the curly brace. This has been changed to specify state name within the curly braces as shown in the example.
3	`add_power_state pd -state sleep {-logic_expr {ctrl == 1}` **`-update`** `}`	`add_power_state pd -state sleep {-logic_expr {ctrl == 1}` **`} -update`**	In UPF 2.0, -update can be specified inside the curly brace and selectively for different state definitions. In UPF 2.1, -update can only be specified outside the curly brace and is applicable for all the state definitions mentioned in the particular command.
4	`add_power_state pd -state standby {-logic_expr {ctrl == 1}`**`} -illegal`** `add_power_state pd -state standby {-logic_expr {ctrl == 1}`**`} -legal`**	`add_power_state pd -state standby {-logic_expr {ctrl == 1}` **`-illegal }`** `add_power_state pd -state standby {-logic_expr {ctrl == 1} -legal` **`}`**	In UPF 2.0, -legal/-illegal can be specified outside the curly brace which can be applicable for all the states defined for the particular command. In UPF 2.1, -legal/-illegal can only be specified inside the curly brace for individual states.
5	`create_supply_set set_name [-function {func_name` **`[`**`net_name`**`]`**`}]*`	`create_supply_set set_name [-function {func_name net_name}]*`	Function needs to mandatorily associated with corresponding `net_name`.
6	`set_isolation isolation_name [-diff_supply_only <TRUE | FALSE>]`	`set_isolation isolation_name [-diff_supply_only` **`[`**`<TRUE | FALSE>`**`]`**`]`	It is not mandatory to specify the boolean value if user intends to specify `TRUE`.
7	`-transitive <TRUE | FALSE> For various UPF Commands`	`-transitive [<TRUE | FALSE>] For various UPF Commands`	It is not mandatory to specify the boolean value if user intends to specify `TRUE`.
8	`create_composite_domain [-supply {supply_set_handle [supply_set_ref]}]*`	`create_composite_domain [-supply {supply_set_handle [supply_set_ref]}]`	In UPF 2.0, `-supply` option on can be used to specify any supply sets visible in active scope. In UPF 2.1, `-supply` option can only be used to specify the primary supply set. It is not possible to specify any other supply sets on a composite power domain.
9	`map_retention_cell [-lib_model_name name` **`{-port port_name net_ref}`**`*]`	`map_retention_cell [-lib_model_name name` **`-port_map {{port_name net_ref}*}`**`]`	In UPF 2.0, `-port` was used to provide port mappings. In UPF 2.1, `-port` option has been changed to `-port_map` with a different syntax. This is consistent with other commands which accept port mapping.

10	use_interface_cell **[-map {{port net_ref}*}]**	use_interface_cell **[-port_map {{port net_ref}*}]**	In UPF 2.0, -map was used to provide port mappings. In UPF 2.1, -map option has been renamed to -port_map. The syntax remains same as the UPF 2.0 version. This is consistent with other commands which accept port mapping.
11	set_design_attributes **[-attribute name value]***	set_design_attributes **[-attribute { name value }]***	In UPF 2.0, -attribute option accepted two separate arguments, name and value. In UPF 2.1, -attribute option accepts a single argument which is a Tcl list having name value pair.
12	set_retention_elements retention_list_name [**{**-applies_to <required | not_optional | not_required | optional>**}**]	set_retention_elements retention_list_name [-applies_to <required | not_optional | not_required | optional>]	Mandatory curly { } braces have been removed from syntax in UPF 2.1

Table 3 – New Features and other differences

S.No	Difference	Description	Reference
1	`find_objects -object_type model`	Enables user to search all instances of a particular model in the design.	IEEE 1801-2013 Section 6.26
2	`add_power_state -supply, -domain, -complete`	Addition of `-supply/-domain` option makes UPF more clear in intent and readable. The `-complete` option indicates that all the power states have already been defined and hence becomes a constraint that no new states can be added during the IP integration.	IEEE 1801-2013 Section 6.4
3	`create_power_switch -instance`	Allows inferring power switches which are already present in the design.	IEEE 1801-2013 Section 6.18
4	`create_power_switch -update`	Allows the addition of `-instance`.	IEEE 1801-2013 Section 6.18
5	`set_port_attributes -feedthrough -unconnected`	Allows users to explicitly specify ports as feed through or unconnected on a hard macro, if it is not inferred by verification tools automatically.	IEEE 1801-2013 Section 6.46
6	`set_simstate_behavior -elements -exclude_elements`	Provides more fine grain control of disabling of simstate semantics.	IEEE 1801-2013 Section 6.53
7	Using dot in various UPF commands to refer to current scope. `create_power_domain pd -elements {.}`	Allow users to easily access current scope.	
8	`set_isolation -exclude_elements`	Allows user to filter out some ports/instances to which this strategy does not apply	IEEE 1801-2013 Section 6.41
9	`set_level_shifter -exclude_elements`	Allows user to filter out some ports/instances to which this strategy does not apply	IEEE 1801-2013 Section 6.43
10	`connect_logic_net -reconnect`	Allows a port that is already connected to a net to be disconnected from the existing net and connected to new net.	IEEE 1801-2013 Section 6.10
11	`set_design_attributes -is_leaf_cell -is_macro_cell`	Allows users to explicitly mark a design element as `leaf_cell` and/or a `macro_cell` so that corresponding power aware semantics can be applied to these.	IEEE 1801-2013 Section 6.37
12	`set_isolation/set_level_shifter strategy_name [-source <source_domain_name | source_supply_ref >] [-sink <sink_domain_name | sink_supply_ref >]`	Allows user to specify power domain in `-source/-sink` option in addition to supply sets. In that case, it automatically expands to `<domain_name>.primary`.	IEEE 1801-2013 Section 6.41
13	Return value of UPF Commands changed to “empty” string, if successful	The return value of most of the commands that have deferred effect was changed to empty string for consistency across various commands. However, some commands return a valid string, e.g. `find_objects`, `set_scope`, `upf_version`. If users relied upon return values of other commands, they need to update their UPF and use other mechanism for getting the information.	IEEE 1801-2013 Section 6
14	`set_repeater`	Allows user to specify buffer strategy	IEEE 1801-2013 Section 6.48
15	`create_power_domain -atomic`	Allows user to create atomic power domains	Section III.B of this paper, “Modeling soft IP power management constraints”
16	Macro cell modeling commands: `begin_power_model` `end_power_model` `apply_power_model`	Allows user to create power model for macro cells	Section III.C of this paper: “Modeling of Hard IP”

17	`create_power_domain –available_supplies`	Allows user to specify supply constraints	Section III.E of this paper: "Verifying supply constraints"
18	`set_equivalent` [`-function_only`] [`-nets` *supply_net_name_list*] [`-sets` *supply_set_name_list*]	Allows user to explicitly define the equivalent supplies	Section III.F of this paper: "Supply Equivalence"
19	`-use_equivalence` in `set_isolation, set_level_shifter, set_repeater`	Allows user to select the criteria to consider supply equivalence or not	Section III.F of this paper: "Supply Equivalence"
20	New simstate `"CORRUPT_STATE_ON_ACTIVITY"`	Use this simstate when the power level is sufficient to power normal functionality for combinatorial logic but insufficient for powering the normal operation on state elements when there is any activity on the state element.	IEEE 1801-2013 Section 9.4.6
21	Power Management Strategy commands `define_always_on_cell` `define_diode_clamp` `define_isolation_cell` `define_level_shifter_cell` `define_power_switch_cell` `define_retention_cell`	Use these commands to define the characteristics of the instances of power management cells used to implement and verify the power intent of given design. These commands provide shorthand to specify various low power attributes for a special power aware cells. They can also be used to override the power aware information specified in a generic library specification of the cell.	Section III.J of this paper: "Power management cell modeling commands"