IEEE STANDARDS ASSOCIATION

IEEE Standard for Design and Verification of Low-Power Integrated Circuits—Amendment 1

IEEE Computer Society

Sponsored by the
Design Automation Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 1801a™-2014
(Amendment to
IEEE Std 1801™-2013)

IEEE Std 1801a™-2014
(Amendment to
IEEE Std 1801™-2013)

IEEE Standard for Design and Verification of Low-Power Integrated Circuits—Amendment 1

Sponsor

Design Automation Committee
of the
IEEE Computer Society

Approved 21 August 2014

IEEE-SA Standards Board

Abstract: The set of changes required to address technical and editorial errors that have been identified in IEEE Std 1801-2013 are specified in this amendment. In addition this amendment also specifies a few changes and enhancements to remove some ambiguities and inconsistencies related to the semantics of power states, power supplies, precedence rules, and location of power management cells.

Keywords: amendment, corruption semantics, IEEE 1801™, IEEE 1801a™, interface specification, IP reuse, isolation, level-shifting, power-aware design, power domains, power intent, power modes, power states, progressive design refinement, retention, retention strategies

---

3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 978-0-7381-9273-4 STD98763
Print: ISBN 978-0-7381-9274-1 STDPD98763

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at http://ieeexplore.ieee.org/xpl/standards.jsp or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this standard was submitted to the IEEE-SA Standards Board for approval, the UPF: Standard for Design and Verification of Low Power Integrated Circuits (C/DA/UPF) Working Group had the following membership:

John Biggs, *Chair*
Erich Marschner, *Vice Chair*
Sushma Honnavara-Prasad, *Secretary*

Paul Bailey
Gustav Bjorkman
Conor Byrne
Louis Cardillo
Shir-Shen Chang
David Cheng
Cyril Chevalier
John Decker
Ayon Dey
Stephan Diestelhorst
Shaun Durnan
John Ellis
Nick English
Alan Gibbons
Josefina Hobbs
Colin Holehouse
Mohit Jain
Fred Jen
Tim Jordan
James Kehoe
Tim Kogel
Rick Koster
Kaowen Liu
Gene Matter
Debajani Majhi
Don Mills
Abigail Moorhouse
Lawrence Neukom
Shreedhar Ramachandra
Judith Richardson
Frederic Saint-Preux
Richard Scales
Guido Schlothane
Krishna Sekar
Amit Srivastava
James Su
Prasad Subbarao
Ajay Thiriveedhi
Yatin Trivedi
Gaurav Varshney
Yossi Veller
Venki Venkatesh
Vita Vishnyakov
Ken Wagner
Qi Wang
Jon Worthington
Vojin Zivojnovic

The following members of the entity balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Accellera Organization, Inc.
Advanced Micro Devices (AMD)
ARM, Ltd.
Cadence Design Systems, Inc.
Cambridge Silicon Radio
Intel Corporation
Jasper Design Automation, Inc.
LSI Corporation
Marvell Semiconductor, Inc.
MEMS Industry Group
Mentor Graphics
Qualcomm Incorporated
STMicroelectronics
Synopsys, Inc.
Texas Instruments Incorporated

When the IEEE-SA Standards Board approved this amendment on 21 August 2014, it had the following membership:

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Introduction

This introduction is not part of IEEE Std 1801a-2014, IEEE Standard for Design and Verification of Low-Power Integrated Circuits—Amendment 1.

This amendment specifies the set of changes required to address technical and editorial errors that have been identified in IEEE Std 1801-2013.

In addition, this amendment also specifies a few changes and enhancements to remove some ambiguities and inconsistencies related to the semantics of power states, power supplies, precedence rules, and location of power management cells. Also the examples in Annex E have been reformatted for clarity.

Contents

IEEE Standard for Design and Verification of Low-Power Integrated Circuits—Amendment 1

NOTE—The editing instructions contained in this amendment define how to merge the material contained therein into the existing base standard and its amendments to form the comprehensive standard.

The editing instructions are shown in ***bold italic***. Four editing instructions are used: change, delete, insert, and replace. ***Change*** is used to make corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed by using ~~strikethrough~~ (to remove old material) and underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editing instructions, change markings, and this NOTE will not be carried over into future editions because the changes will be incorporated into the base standard.

4. UPF concepts

4.4.2.5.1 Explicit and automatic connections

Change the second paragraph to use body text throughout as follows:

An automatic connection connects each supply set function to ports of selected instances, based on the *pg_type* of each port, as indicated by the **UPF_pg_type** attribute (see 6.46) or the Liberty `pg_type` attribute.

5. Language basics

5.2 Conventions used

Change the square brackets example in row 8 of Table 1 as follows:

[**-ack_port** {*port_name net_name* [{~~*boolean_expression*~~*logic_value*}]}]*

Change the curly braces explanation in row 10 of Table 1 as follows:

Curly braces ({ }) indicate a parameter list that is required. ~~In some (or even many) cases, they have (or are followed by) an asterisk (*), which indicates that they can be repeated.~~ If the curly braces are followed by an asterisk ({}*), it indicates that the option can be repeated. For example, the following shows one or more control ports can be specified for this command:

5.4 Boolean expressions

Insert the following paragraph immediately after the third paragraph, which begins "A Boolean expression may also contain...":

In certain commands, logic values X, 0, 1, Z can be specified. These represent values of a predefined logic type in the relevant hardware description language. For VHDL, the predefined logic type is type ieee.std_logic_1164.std_ulogic, or any subtype thereof. For SystemVerilog, the predefined logic type is type Logic.

5.6 Attributes of objects

Delete the first paragraph and insert new paragraphs as follows:

~~HDLs include a mechanism for specifying properties of objects. These properties are called *attributes*. Certain UPF properties can be annotated directly in HDL source descriptions using attributes. The semantic for properties specified using HDL attributes is the same as the corresponding behavior defined by the UPF command alternative (see Clause 6). Table 4 enumerates the HDL attributes defined for UPF-compliant implementations.~~

UPF supports the specification of *attributes*, or properties, of objects in a design. These attributes provide information that supports or affects the meaning of related UPF commands. Such attributes can also be defined with HDL attribute specifications in design code or with Liberty attribute specifications in a Liberty model.

Table 4 enumerates the attributes that have a predefined meaning in UPF and for each attribute, the UPF command that can be used to define that attribute.

Change Table 4 column 1 title (including the continuation title on the next page) as follows:

~~HDL attribute name~~

UPF predefined attribute name

Change the second paragraph as follows:

The ~~HDL~~ attributes in Table 4 all take values that are string literals. Where a list of names is required, the names in the list should be separated by spaces and without enclosing braces (`{}`).

These attributes can also be specified using the attribute mechanism in SystemVerilog code or using attribute specifications in VHDL code. To attach a ~~UPF~~ attribute to an object in a VHDL context, the ~~UPF~~ attribute shall be declared first, with a data type of `STD.Standard.String` (or the equivalent), before any attribute specification for that attribute.

For determination of precedence (see 5.8), attributes specified in HDL code are treated as if they were implicitly specified using the UPF command **set_port_attributes -model -ports** (for port attributes) or the UPF command **set_design_attributes -models** (for design attributes).

Insert the following paragraphs immediately after the second paragraph:

Some of these attributes may also be implied by attributes in a Liberty model. Specifically, the following Liberty attributes imply definition of the corresponding UPF predefined attribute:

Liberty attribute name	implies	**UPF predefined attribute name**
pg_type		UPF_pg_type
related_power_pin		UPF_related_power_port
related_ground_pin		UPF_related_ground_port
related_bias_pins		UPF_related_bias_ports
short		UPF_feedthrough
is_macro_cell		UPF_is_macro_cell

For determination of precedence (see 5.8), attributes specified in Liberty models are treated as if the corresponding UPF attribute name were implicitly specified using the UPF command **set_port_attributes -model -ports** (for port attributes) or the UPF command **set_design_attributes -models** (for design attributes).

Certain attributes represent characteristics of a module or cell that apply universally to all instances of that module or cell. Such attributes are called characteristic attributes. The following predefined attributes are always characteristic attributes:

UPF_feedthrough
UPF_unconnected
UPF_is_macro_cell
UPF_retention

In addition, any attribute specified either explicitly or implicitly with set_port_attributes **-**model or set_design_attributes **-**models is a characteristic attribute, except for the following:

UPF_pg_type
UPF_related_power_port
UPF_related_ground_port
UPF_related_bias_ports

Non-characteristic attributes are overridable as specified by the precedence rules for attribute specifications (see 5.8). Characteristic attributes are non-overridable.

NOTE—The above definitions imply that, other than the four specific exceptions listed above, any attribute derived from a Liberty attribute or specified in an HDL model cannot be overridden by a higher precedence attribute specification in UPF (see 5.8).

Change the "Equivalent UPF command arguments" for "UPF_retention" in row 12 of Table 4 as follows:

~~set_retention_elements -retention_purpose~~
set_design_attributes -attribute {UPF_retention required}
set_design_attributes -attribute {UPF_retention optional}

Change the "Equivalent UPF command arguments" for "UPF_simstate_behavior" in row 13 of Table 4 as follows:

~~set_simstate_behavior~~
set_design_attributes -attribute {UPF_simstate_behavior ENABLE}
set_design_attributes -attribute {UPF_simstate_behavior DISABLE}

Change the attribute name in the example below the third paragraph as follows:

Attribute name: **UPF_related_bias_~~pin~~ports**

Change the last paragraph in the example showing the attribution of elements requiring retention as follows:

The same attribute can be specified in UPF, using the **~~set_retention_elements~~** **set_design_attributes** command ~~and its specific option~~ **~~-retention_purpose~~** ~~(see 6.51).~~ (see 6.37).

```
set_design_attributes -elements {my_flip} \
        -attribute {UPF_retention required}
```

5.8 Precedence

Change the precedence criteria in the third paragraph to insert item d) and item e) as follows and reletter accordingly:

a) Command that applies to part of a multi-bit port specified explicitly by name
b) Command that applies to a whole port specified explicitly by name
c) Command that applies to all ports of an instance specified explicitly by name
d) Command that applies to a port of a specified power domain with a given sink and source
e) Command that applies to a port of a specified power domain with a given sink or source
f) Command that applies to all ports of a specified power domain with a given direction
g) Command that applies to all ports of a specified power domain

Change the sixth paragraph as follows:

h) Command that automatically connects to ports of an instance ~~(~~
(e.g., `connect_supply_set -connect -elements`)

i) Command that automatically connects to ports of any instance in a given region
(e.g., `connect_supply_set -connect` or
`connect_supply_net -pg_type -domain/-cells/-elements`)

Change the eighth paragraph as follows and reletter the ordered list accordingly:

~~For attribute specifications, there is no definition of precedence to select which of several potentially applicable specifications apply. It is an error if any two UPF, HDL, or Liberty attribute specifications provide different values for the same attribute of the same object.~~

If multiple set_port_attributes commands potentially specify the same overridable attribute of a given port, whether specified explicitly in UPF or implied by HDL or Liberty attribute specifications, only the command(s) with the highest precedence will actually apply. The following criteria (listed in order from highest precedence to lowest precedence) determine the relative precedence of the commands.

The command references

a) A part of the given port, specified explicitly by name in the **-ports** list (without **-model**)
b) The whole given port, specified explicitly by name in the **-ports** list (without **-model**)
c) The given port, implied by specifying an instance name in the **-elements** list with a given direction
d) The given port, implied by specifying an instance name in the **-elements** list
e) A part of the given port of the named module or library cell, specified explicitly by name in the **-ports** list (with **-model**)
f) The whole given port of the named module or library cell, specified explicitly by name in the **-ports** list (with **-model**)
g) The given port of the instance corresponding to the current scope if none of the options **-ports**, **-elements**, **-model** are present

It is an error if the precedence rules fail to uniquely identify the value of the UPF attribute that applies to a port. In other words, it is an error if two UPF attribute specifications with the same highest precedence specify different values for the same attribute of the same port.

It is an error if a non-overridable attribute is specified with two different values for the same object, regardless of the precedence rules for attribute specifications.

5.11 Command refinement

Change the example as follows:

b) Logical configuration

```
set_isolation demo_strategy -domain pda
    -elements {a b c d}
    -clamp_value {0}
    -isolation_signal {iso_en}
    -isolation_sense {LOW}
```

6. Power intent commands

6.3 add_port_state [legacy]

Change the second paragraph as follows:

The **add_port_state** command adds state information to a supply port. If the voltage values are specified, the supply net state is **FULL_ON** and the voltage value is the single nominal value or within the range of min to max; otherwise, if **off** is specified, the ~~voltage value~~ supply net state is **OFF**.

6.4 add_power_state

Change restriction l) as follows:

l) If a logic expression is used to define a power state of a given power domain, it shall only refer to logic ports, logic nets, interval functions, power states of supply sets or supply set handles, and/or power states of ~~other~~ power domains. It is an error if such a logic expression refers to supply ports, supply nets, or functions of a supply set or supply set handle.

Insert the following additional restriction:

w) It is an error if a logic expression used to define a given power state contains a direct or indirect reference to that same state.

Change the logic expression on line 3 of the syntax example to add the curly braces as follows:

```
-logic_expr {SW_ON} -simstate NORMAL
```

6.7 associate_supply_set

Change item d) as follows:

d) The predefined supply set handles for a level-shifter strategy *level_shifter_name* (see 6.43) are *domain_name*.*level_shifter_name*.**input_supply_set** and *domain_name*.*level_shifter_name*.**output_supply_set**.

Change item e) as follows:

e) The predefined supply set handle for a retention strategy *retention_name* (see 6.49) is *domain_name*.*retention_name*.**retention_supply_set**.

6.11 connect_supply_net

Change the pg_type argument in both the syntax table summary and argument list as follows:

[**-pg_type** ~~{~~*pg_type_list* ~~*element_list*}~~]*

Insert the following option:

[**-elements** element_list]	The list of instance names.

Insert the following to the "Use the following:" list

-elements to connect all pins of the appropriate type (power or ground) on the specified instances.

Insert the following to the "The following also apply:" list as follows:

— If **-ports** is not specified, **-pg_type** and one or more of **-cells**, **-domains**, and **-elements** must be specified.

Change the fourth bullet in the "The following also apply:" list as follows:

— The **-ports** option is mutually exclusive with the **-cells**, **-domain**, **-elements**, and **-pg_type** options.

6.12 connect_supply_set

Change item c) as follows:

c) When *supply_set_ref* refers to a handle associated with a domain and -elements is not specified in the base command or any update then ~~the *aggregate_element_list* is empty,~~ all elements in the extent of the domain are added to the *aggregate_element_list*.

6.17 create_power_domain

Change the last sentence of the eleventh paragraph as follows:

However, the primary supply set shall not be implicitly connected when ~~any~~ either of the following apply:

Delete the following list item from the eleventh paragraph:

~~c) An instance is created as a result of a UPF command, e.g., isolation cells, level-shifters, power switches, and retention registers.~~

6.18 create_power_switch

Change the definition of the output_supply_port argument in the syntax table as follows:

The output supply port of the switch and, optionally, the supply net where this port connects. *supply_net_name* is a rooted name of a supply net or supply port. It shall be an error if the *supply_net_name* is not defined in the current scope.

Replace the syntax definition of -instance in the summary section of the syntax table with the following:

[**-instance** *instance_list*]

Replace the syntax definition of -instance in the argument definition section of the syntax table with the following:

-instance *instance_list*

Change the third sentence of the fifth paragraph as follows:

If not specified explicitly, the *off_state* expression defaults to the complement of the ~~conjunction~~ disjunction of all the *on_state*, *on_partial_state*, and *error_state* expressions defined for the power switch.

Change the sixth and seventh paragraphs as follows:

~~A contributing input supply port is one that has an *on_state* expression or *on_partial_state* expression that evaluates to *True* at a given time. The contributed value of a contributing input supply port is the value of the supply source connected to that input supply port. The degraded value of a contributing input supply port is the contributed value, except that if the contributed value's net state is **FULL_ON**, the degraded value's net state is **PARTIAL_ON**.~~

An *on_state* or *on_partial_state* specification for a power switch contributes a value to the computation of the power switch output port's value at any given time. If an *on_state* or *on_partial_state* Boolean expression for a given input supply port refers to an object with an unknown (X or Z) value, and that input supply port has a net state other than OFF, then the contributed value is {**UNDETERMINED,** unspecified}. If an *on_state* Boolean expression for a given input supply port evaluates to True, then the contributed value is the value of that input supply port. If an *on_partial_state* Boolean expression for a given input supply port evaluates to *True*, then the contributed value is the degraded value of that input supply port. The degraded value of an input supply port is the value of that port, except that if the port value's net state is **FULL_ON**, the degraded value's net state is **PARTIAL_ON**.

The value of the output supply port of a power switch is determined as follows. At any given time:

a) The output supply takes on the value {**UNDETERMINED**, unspecified} if
    1) any *error_state* condition is *True*, or
    2) an explicit *off_state* condition and any *on_state* or *on_partial_state* condition are both *True*, or
    3) any ~~input supply port's~~ contributed value has a net state of **UNDETERMINED**, or
    4) any two ~~input supply ports'~~ contributed values have different voltage values.

b) Otherwise, the switch output takes on any contributed value ~~the contributed value of any contributing input supply port~~ whose net state is **FULL_ON**, if there is one.

c) Otherwise, the switch output takes on any contributed value ~~the degraded value of any contributing input supply port~~ whose net state is **PARTIAL_ON**, if there is one.

d) Otherwise, the switch takes on the value {**OFF**, unspecified}.

Change the eleventh paragraph as follows:

If **-supply_set** is specified for a switch, it powers logic or timing control circuitry within the switch. ~~and powers any specified **-ack_port**s.~~ When the supply set simstate is anything other than **NORMAL**, ~~the state of the output supply port of a switch is **UNDETERMINED** and~~ the acknowledge ports are corrupted. If a supply set is not associated with a switch, ~~it shall be an error if any acknowledge ports are specified.~~ then the following shall apply:

- It shall be an error if any acknowledge ports are specified.
- The receiving supply of the control ports will be undefined.

Change the -ack_port parameters in second example in "Example 2—Two-stage switch" as follows:

```
-ack_port {ts_ack "" 1}
```

6.20 create_supply_net

Change the second paragraph as follows:

If **-domain** is specified, the supply net is created in the scope of that domain for use in the extent of the domain, subject to the supply availability defined by the **-available_supplies** option of the relevant **create_power_domain** command (see 6.17). ~~, and the supply net is available for use by tools to power cells only in the extent of the domain.~~

Delete the following note:

~~NOTE—Use **set_scope** (see 6.52) to change the scope prior to calling this command to set the current scope to the correct scope for the net.~~

6.21 create_supply_port

Change the first sentence of the third paragraph as follows:

Supply ports with direction **inout** shall be used to connect resolved supply nets (see 9.1)~~.connected to a net shall be **inout** for supply nets that have both loads and sources within that module~~.

6.24 describe_state_transition

Change the fifth paragraph as follows:

It shall be an error if the state name in a *list* does not refer to a power state of the specified supply set ~~state~~ or power domain (see 6.4).

6.26 find_objects

Change the second paragraph as follows:

By default, or if **-transitive FALSE** is specified explicitly, **find_objects** searches only the ~~current~~ specified scope of the logic hierarchy. If **-transitive TRUE** is specified, **find_objects** searches the ~~current~~ specified scope and ~~the~~ its entire ~~dependant~~ descendant subtree. If **-transitive** is specified without an argument, it is equivalent to specifying **-transitive TRUE**.

6.26.1 Pattern matching and wildcarding

Insert the following notes at the end of this subclause:

NOTE 1—Some characters used as operators in either glob-style or regular expression style *search_patterns*, such as [ ], \, and { }, also have meaning for Tcl in general. To ensure that such characters are not interpreted by the Tcl processor, the whole pattern can be enclosed in curly braces. This inhibits variable, command, and backslash substitution within the pattern by the Tcl processor (see 5.3.4).

NOTE 2—Square brackets used within a *search_pattern* will be interpreted as indicating a set of characters, any of which will match a single character in a name. To use square brackets to refer to one or more bits of a bus, the square brackets must be escaped. For example, B\[3\] refers to B[3]. The two interpretations of square brackets can also be used in combination. For example, B\[[1-3]\] refers to B[1], B[2], B[3].

6.26.2 Wildcarding examples

Delete both NOTE 1 and NOTE 2.

Insert the following after Table 5:

In particular, to return individual bus bits, instead of a bus name, the *search_pattern* pattern shall explicitly contain escaped brackets \[and \]. For example, for a design with the following objects:

`xyz1` .... a single bit net
`xyz2[3:0]` .... a four-bit bus
`xyz[1:0]` ...... a two-bit bus

Table 5a shows the return value for each of the following examples of **find_objects**.

```
find_objects top -pattern xyz*
find_objects top -pattern xyz
find_objects top -pattern {xyz*\[*\]}
find_objects top -pattern {xyz\[*\]}
find_objects top -pattern {xyz*\[0\]}
```

Table 5a—Bus patterns and return values

`xyz*`	Returns bus/single-bit net names only: `xyz1 xyz2 xyz`
`xyz`	Returns the bus `xyz` only (no wild card)
`xyz*\[*\]`	Returns individual bus bits: `xyz2[3] xyz2[2] xyz2[1] xyz2[0] xyz[1] xyz[0]`
`xyz\[*\]`	Returns individual bus bits: `xyz[1] xyz[0]`
`xyz*\[0\]`	Returns individual bus bits: xyz2[0] xyz[0]

6.28 load_upf

Change the syntax table as follows:

Purpose	Set the scope to the specified instance and execute the specified UPF commands	
Syntax	**load_upf** *upf_file_name* [**-scope** *instance_name_list*] ~~[**-version** *upf_version*]~~	
Arguments	*upf_file_name*	The UPF file to execute.
	-scope *instance_name_list*	The list of scopes where the UPF commands contained in *upf_file_name* are executed.
	~~**-version** *upf_version*~~	~~The UPF version for which commands in this file are written. See also 6.54.~~
Deprecated arguments	[**-version** *upf_version*]	This argument is ignored and provided for syntactic backward compatibility only. This is a deprecated option; see also 6.1 and Annex D.
Return value	Return an empty string if successful or raise a `TCL_ERROR` if not.	

Delete the fifth paragraph as follows:

~~If **-version** *upf_version* is specified, the command~~

~~`upf_version` *upf_version*~~

~~is implicitly executed before executing the commands in the loaded file.~~

Change the syntax example as follows:

```
load_upf my.upf -scope {I1/I2 I3/I2} ~~-version 2.1~~
```

6.29 load_upf_protected

Change the syntax table as follows:

Purpose	Load a UPF file in a protected environment that prevents corruption of existing variables	
Syntax	**load_upf_protected** *upf_file_name* [**-hide_globals**] [**-scope** *instance_name_list*] ~~[**-version** *upf_version*]~~ [**-params** *param_list*]	
Arguments	*upf_file_name*	The UPF file be sourced.
	-hide_globals	Save all globals before sourcing *upf_file_name* and restore them afterwards. Globals named in the *param_list* retain any modified values resulting from sourcing the file. Any globals not in the *param_list* shall be unset before *upf_file_name* is loaded. Any globals created in the sourced file, other than the ones named in *param_list*, are unset at the end of loading.
	-scope *instance_name_list*	The list of scopes where the UPF commands contained in *upf_file_name* are executed.
	~~**-version** *upf_version*~~	~~The UPF version for which commands in this file are written. See also 6.54.~~
	-params *param_list*	A list of variables to be made available while sourcing the file. In *param_list*, each element has one of the following formats: a) *param_name* — declared as "`global $paramName`". Any changes made to this variable are visible at the calling level once this command completes. b) {*param_name param_value*} — a local variable *param_name* is created and its initial value is set to *param_value*. The Tcl variable `errorInfo` shall behave as if it has been specified in this list.
Deprecated arguments	[**-version** *upf_version*]	This argument is ignored and provided for syntactic backward compatibility only. This is a deprecated option; see also 6.1 and Annex D.
Return value	Return an empty string if successful or raise a `TCL_ERROR` if not.	

Delete the fourth paragraph as follows:

~~If **-version** *upf_version* is specified, the command~~

~~`upf_version upf_version`~~

~~is implicitly executed before executing the commands in the loaded file.~~

Change the syntax example as follows:

`load_upf_protected my.upf -hide_globals` ~~`-version 2.0`~~

6.40 set_equivalent

Change the path names in the syntax example to remove the leading "/" as follows:

`set_equivalent -function_only -sets {` ~~`/`~~`sys/aon_ss` ~~`/`~~`mem/PD1.core_ssh }`

6.41 set_isolation

Change the definition of clamp value in the syntax table to remove the default as follows:

The value(s) that the isolation cell can drive. ~~The default is **0**.~~

Change paragraph 21 as follows:

If ~~two or more isolation strategies apply to the same port on the interface of a power domain, such that multiple isolation cells need to be~~ isolation cell insertion is inferred for different paths from ~~that~~ a port ~~to a receiving domain~~, the **-location** specified explicitly or implicitly ~~for each of those strategies~~ by the strategy shall be such that the ~~various~~ isolation cell(s) can be inserted without splitting the port into multiple ports. It shall be an error if ~~multiple isolation strategies for the same~~ an isolation strategy for a port cannot be implemented without duplicating the port.

Change paragraph 24 as follows:

~~If **-clamp_value** is not specified, it defaults to `0`.~~
It is an error if **-clamp_value** is not specified.

Change the first sentence of paragraph 28 as follows:

-isolation_supply_set specifies the supply set(s) that shall be used to power the inferred isolation cell, including the logic receiving the isolation signal(s).

6.43 set_level_shifter

Change the first sentence of paragraph 23 as follows:

-input_supply_set specifies the supply set connected to input supply ports of the level-shifter ~~(see 7.4)~~.

Change the first sentence of paragraph 24 as follows:

-output_supply_set specifies the supply set connected to the output supply ports of the level-shifter ~~(see 7.4)~~.

6.44 set_partial_on_translation

Change the first paragraph as follows:

~~This command defines the translation of **PARTIAL_ON** to **FULL_ON** or **OFF** for purposes of evaluating the power state of supply sets and power domains. The state of a supply *set* is evaluated after **PARTIAL_ON** is translated to **FULL_ON** or **OFF** for each supply *net* in the set.~~

This command causes translation of **PARTIAL_ON** to **FULL_ON** or **OFF**, as specified by the command argument, for purposes of evaluating the power state of supply sets and power domains. If this command is executed in a given run, the state of a supply set is evaluated after **PARTIAL_ON** is translated to **FULL_ON** or **OFF** for each supply net in the set. If this command is not executed in a given run, no translation of **PARTIAL_ON** is performed.

6.46 set_port_attributes

Change the definition of "-ports" in the syntax table as follows:

A list of simple names (if used with -model) or rooted names (otherwise) of ports to be attributed.

Change the third paragraph as follows:

The set of ports attributed is determined as follows:

a) A set of candidate ports is first identified. This set includes the following:

   1) If **-elements** is specified, all ports of each instance named in the elements list are included in the candidate set, including any logic ports inferred from **create_logic_port** (see 6.16), but excluding any supply ports.

   2) If **-ports** is specified, each port named in the ports list is included in the candidate set.

   3) If **-model** is specified and **-ports** is also specified, each port of the named module or library cell named in the ports list is included in the candidate set.

   4) If none of the options **-ports**, **-elements**, **-model** are present, every port of the instance corresponding to the current scope is included in the candidate set, including any logic ports inferred from create_logic_port (see 6.16), but excluding any supply ports.

b) The candidate set is then restricted to those ports that satisfy any filters specified. A port is removed from the candidate set if:

   1) The port name appears in the **-exclude_ports** list.

   2) The port is a port on an instance named in the **-exclude_elements** list.

   3) The port direction is not consistent with the direction identified by the **-applies_to** option.

c) The resulting restricted set is the final candidate set of ports to be attributed.

If a given port is included in the final candidate set of ports of more than one **set_port_attribute** command, the precedence rules (see 5.8) determine which of those **set_port_attribute** commands actually apply to that port.

Change the fourth paragraph as follows:

If **-model** is specified, the port attributes are applied to the selected ports of ~~each instance of~~ the model. In this case, only simple names that are declared in the model may be referenced in arguments to this command and all names are interpreted relative to the topmost scope of the model. If **-model** is not specified, the port attributes are applied to the selected instance ports. In this case, only rooted names of instance ports may be referenced in this command, and all such names are interpreted relative to the current scope.

Change the first sentence of the fifth paragraph as follows:

-model and **-ports** ~~**-attribute**~~ can be used together to specify attributes for ports of a hard IP.

Insert the following additional errors to "The following also apply" list:

- It shall be an error if **-ports** is specified and **-elements** is also specified.
- It shall be an error if attribute **UPF_driver_supply** or **UPF_receiver_supply** is specifed for a hard macro port that also has the attribute **UPF_unconnected** associated with it.

6.48 set_repeater

Change the syntax table as follows:

[**-exclude_elements** *exclude_list*]

6.49 set_retention

Change the color of the -transitive option in the syntax table as follows:

~~[**-transitive** [<**TRUE** | **FALSE**>]]~~
[-transitive [<TRUE | FALSE>]]

Remove the -transitive option from the Arguments section of the syntax table.

Insert the following to the Legacy Arguments section of the syntax table:

-transitive <TRUE|FALSE> When -transitive is TRUE (the default), the command applies to the descendants of the elements.
This is a deprecated option; see also 6.1 and Annex D.

Change the color of the hyphen in the -update option in the syntax table as follows:

[~~-~~**-update**]

Change the fourth paragraph as follows:

~~**-retention_supply_set** powers the register holding the retained value.~~

-retention_supply_set specifies the supply set that shall be used to power the state element holding the retained value, as well as the control logic, if any, that evaluates the **-save_condition**, **-restore_condition**, and **-retention_condition**. The supply set specified by **-retention_supply_set** is implicitly connected to the retention logic inferred by this command. If **-retention_supply_set** is not specified, the -**default_retention** supply of the power domain for which the retention strategy is defined is used as the retention supply. For example, if **set_retention** is specified for a domain PD and **-retention_supply_set** is not specified, then **PD.default_retention** is used.

Change the ninth paragraph as follows:

-retention_condition defines the retention behavior of the retention element~~ while the primary supply is not `NORMAL`~~. If the retention condition evaluates to `FALSE` and the primary supply is not `NORMAL` the retained value of the register is corrupted. The receiving supply of any pin listed in the **-retention_condition** shall be ~~assumed to be~~ at least as on as the retention supply of the retention strategy.

Delete the eleventh paragraph and insert a new paragraph as follows:

~~**-use_retention_as_primary** powers the storage element and the output drivers of the register using the retention supply. The result of this is the simstate for the retention supply set is applied to the register's output. Inferred state elements shall be consistent with the **-use_retention_as_primary** constraint.~~

The normal mode storage element of the retention register is powered by the primary supply of the domain, therefore the receiver supply of the retention register's data input is the primary supply. By default, the output driver of the retention register is also powered by the primary supply of the domain, in which case the driver supply of the retention register output is the primary supply. However, if **-use_retention_as_primary** is specified, the retention supply powers the output driver of the register instead, and the driver supply of the data output of the retention register is therefore the retention supply. In the latter case, the simstate for the retention supply set is applied to the register's output. Inferred state elements shall be consistent with the **-use_retention_as_primary** constraint.

6.51 set_retention_elements

Change the first paragraph as follows:

The **set_retention_elements** command defines a~~n "atomic"~~ list of objects whose state shall be retained or not retained together by the **set_retention** and **map_retention_cell** commands (see 6.49 and 6.33).

Change the third paragraph as follows:

-applies_to filters the *effective_element_list,* removing any elements that ~~do not have a **UPF_retention** attribute value~~ are not consistent with the selected filter choice: **required**, **optional**, **not_required,** or **not_optional**, ~~**not_required**, or **optional**~~, as follows:

Filter choice **required** ~~matches~~ removes all elements that do not have the **UPF_retention** attribute value `required`.

Filter choice **optional** ~~matches~~ removes all elements that do not have the **UPF_retention** attribute value `optional`.

Filter choice **not_required** ~~matches~~ removes all elements that do ~~not~~ have the **UPF_retention** attribute value `required`.

Filter choice **not_optional** ~~matches~~ removes all elements that do ~~not~~ have the **UPF_retention** attribute value `optional`.

6.54 upf_version

Change the first paragraph as follows:

The **upf_version** command returns a string value representing the UPF version currently being used by the tool reading the UPF file. When the UPF version defined by this standard is being used, the returned value shall be the string "2.~~1~~2". **upf_version** may also include an argument that documents the UPF version for which the UPF commands that follow were written. For UPF commands intended to be interpreted according to the UPF version defined by this standard, the argument shall be the string "2.~~1~~2".

Change syntax example as follows:

upf_version 2.~~1~~2

6.55 use_interface_cell

Change the syntax example to insert the -lib_cells argument as follows:

```
use_interface_cell my_interface -strategy {ISO1 LS1} -domain PD1 \
  -lib_cells {combo1 combo2} \
  -elements {top/moduleA/port1 top/moduleA/port2 top/moduleA/port3}
```

7. Power management cell commands

7.2 define_always_on_cell

Change the definition of -power and -ground in the syntax table summary as follows:

-power *power_pin*
-ground *ground_pin*

Change the definition of -power and -ground in the syntax table arguments as follows:

-power *power_pin*
-ground *ground_pin*

7.4 define_isolation_cell

Change the definition of -power and -ground in the syntax table summary as follows:

-ground *~~power~~ground_pin*

Change the definition of -power and -ground in the syntax table arguments as follows:

-ground *~~power~~ground_pin*

7.5 define_level_shifter_cell

Change the first sentence of the -pin_groups definition in the syntax table as follows:

Specifies a list of input-output paths for multi-bit ~~isolation~~ level-shifter cells

9. Simulation semantics

9.1 Supply network creation

Change the first sentence in the fifth paragraph as follows:

If a supply net is connected to a HDL port of a single bit type, a default VCT that maps the **FULL_ON** state to logic 1 and the **~~FULL_~~OFF** state to logic 0 shall be inserted automatically.

9.2 Supply network simulation

9.2.2 Power-switch evaluation

Delete the first paragraph and insert a new paragraph as follows:

~~During simulation, a power switch created with **create_power_switch** corresponds to a process that is sensitive to changes in its input port (net state and voltage value), as well as its control ports. [A general introduction to power-switch behavior is described here (see 6.18 for the complete power-switch semantics).] Whenever the signals on the control ports change, the corresponding on-state Boolean functions are evaluated. If an on-state function evaluates *True*, the switch is closed, which causes the state of its input port to propagate to the output port (or for a multiplexed switch, the corresponding input is switched to the output), otherwise the switch is opened – the output supply port is assigned the state **OFF** and the voltage value is unspecified. If any of the control signals is `X` or `Z`, the input supply port is **UNDETERMINED**, the control signals match one of the error-state Boolean functions, or more than one on-state function evaluates *True*, then the behavior of the output supply port is assigned the state **UNDETERMINED**, the voltage level shall be unspecified, and the acknowledge ports shall be driven `X`; in this case, implementations may issue a warning or an error.~~

During simulation, a power switch created with **create_power_switch** corresponds to a process that is sensitive to changes in its input port (net state and voltage value), as well as the signals referenced in the Boolean expressions that define its control inputs. Whenever the input supply ports or control signals change, the corresponding on-state, on-partial-state, off-state, and error-state Boolean functions are evaluated and the value of the power switch output port is recomputed. See 6.18 **create_power_switch** for the algorithm used to determine the output value of the switch.

Delete the Example starting on page 156, up to but not including 9.2.3.

9.3 Power state simulation

9.3.2 Power state determination

Change the pseudo code for determining the power state of a supply set as follows:

```
else (PS has both a logic expression and a supply expression)
  ~~if the logic expression is True, then~~
    ~~CPS = CPS + {PS}~~
    ~~if the supply expression is False, then~~
      ~~Error: Supply status insufficient to support power state~~
    ~~end~~
  if both the logic expression and the supply expression are True, then
    CPS = CPS + {PS}
  end
end
```

9.6 Simulation of retention

9.6.2 Retention modeling for different retention styles

Change the font in the state definition in the second row of Table 9 as follows (from Times New Roman to Courier New):

~~RETAIN-ON/RETAIN-OFF~~
`RETAIN-ON/RETAIN-OFF`

9.7 Simulation of isolation

Replace the entirety of this subclause with the following:

The simulation semantics for isolation are defined by the following algorithm, unless a specific simulation model is specified for a given instance by a **use_interface_cell** command (see 6.55):

For a single-stage isolation cell with **isolation_sense** high and a **clamp_value** of **0**, **1**, **Z** from a predefined logic type (see 5.4), or any value of a user-defined datatype:

```
on any input change,
   if the current simstate of the isolation supply set is NORMAL, then
      if isolation_signal == 0 then
         data_output = data_input;
      else if isolation_signal == 1 then
          data_output = clamp_value;
      else /* isolation_signal has an unknown value */
          data_output = corrupted value;
      end
   else /* the isolation supply set is in a non-NORMAL state */
      data_output = corrupted value;
   end;
```

where the corrupted value is X from the predefined logic type for a 1-bit port of that type, an array of X values for a port that is an array with elements of that type, and the leftmost value of the relevant data type for any port that is of a user-defined datatype. For an isolation cell with **isolation_sense** low, the isolation signal values 0 and 1 would be interchanged.

For a single-stage isolation cell with isolation_sense high and a clamp_value of latch:

```
on any input change,
   if the current simstate of the isolation supply set is NORMAL, then
      if isolation_signal == 0 then
         data_output = data_input;
         latched_value = data_input;
      else if isolation_signal == 1 then
         data_output = latched_value;
      else /* isolation_signal has an unknown value */
         data_output = corrupted value;
         latched_value = corrupted value;
      end
   else /* the isolation supply set is in a non-NORMAL state */
      data_output = corrupted value;
      latched_value = corrupted value;
   end;
```

For a multi-stage isolation cell with N stages, each stage is simulated as given above, and the multiple stages are composed as follows:

```
  isolation_stage[1].input = data_input;
  isolation_stage[1].isolation_supply_set = isolation_supply_set[1];
  isolation_stage[1].isolation_signal = isolation_signal[1];
  for each stage K in 2 to N,
    isolation_stage[K].input = isolation_stage[K-1].output;
    isolation_stage[K].isolation_supply_set = isolation_supply_set[K];
    isolation_stage[K].isolation_signal = isolation_signal[K];
  end;
  data_output = isolation_stage[N].output;
```

9.8 Simulation of level-shifting

Replace the entirety of this subclause with the following:

The simulation semantics for level shifting are defined by the following algorithm, unless a specific simulation model is specified for a given instance by a **use_interface_cell** command (see 6.55):

```
on any input change,
   if the current simstate of any level shifter supply set \
         is not NORMAL, then
      data_output = corrupted value;
   else
      data_output = data_input;
   end;
```

where the corrupted value is as defined in 9.7.

Change the title of 9.9 to be plural as follows:

9.9 Simulation of repeaters

Replace the entirety of this subclause with the following:

The simulation semantics for repeaters are defined by the following algorithm:

```
on any input change,
   if the current simstate of the repeater_supply_set \
         is not NORMAL, then
      data_output = corrupted value;
   else
      data_output = data_input;
   end;
```

where the corrupted value is as defined in 9.7.

Annex C

(normative)

Queries

C.26 query_power_state

Change the second paragraph to remove the back tick and comma from the power state definitions as follows:

-detailed returns all the parameters of the specified power state *state_name* as a Tcl list consisting of {*key value*} pairs. For example, if a legal state called `LPS` on the supply set `PDA_SUPPLY` has the **-supply_expr** condition {power == ~~`~~{FULL_ON~~,~~ 0.8}} and the **-logic_expr** condition `{u1/PdA == GO_MODE}`, then the **-detailed** option returns the following:

```
{state_name LPS} {object_name PDA_SUPPLY} {supply_expr {power ==
~~`~~{FULL_ON~~,~~ 0.8}}} {logic_expr {u1/PdA == GO_MODE}} {legal 1}
{illegal 0} {simstate {}}
```

Change the third paragraph to remove the back tick and comma from the power state definition as follows:

Without the **-detailed** option, the format of the returned parameters shall be in the format of the corresponding **add_power_state** command, i.e.,

```
add_power_state PDA_RET
    -state {LPS
    -supply_expr {power == ~~`~~{FULL_ON~~,~~ 0.8}}
    -logic_expr {u1/PdA == GO_MODE}
    -legal}
```

Annex D

(normative)

Replacing deprecated and legacy commands and options

D.1 Deprecated and legacy constructs

Insert the following new subclauses:

D.1.1.3.a 6.28

load_upf *upf_file_name*
...
[**-version** *upf_version*] (This is a deprecated option.)

D.1.1.3.b 6.29

load_upf_protected *upf_file_name*
...
[**-version** *upf_version*] (This is a deprecated option.)

D.1.2.6 6.49

Insert the following line to set_retention

[-transitive [<TRUE|FALSE>]] (This is a deprecated option.)

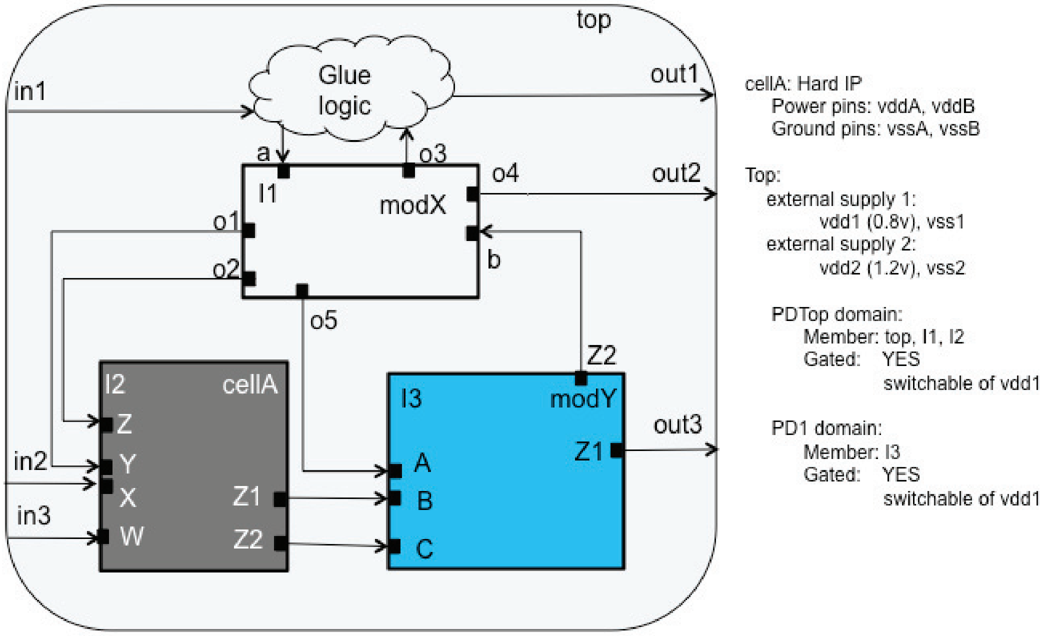
Annex E

(informative)

Low-power design methodology

E.1 Design, implementation, and verification flow for a soft IP

Replace Figure E.1 with the following figure:



cellA: Hard IP
Power pins: vddA, vddB
Ground pins: vssA, vssB

Top:
external supply 1:
vdd1 (0.8v), vss1
external supply 2:
vdd2 (1.2v), vss2

PDTop domain:
Member: top, I1, I2
Gated: YES
switchable of vdd1

PD1 domain:
Member: I3
Gated: YES
switchable of vdd1

Figure E.1—Simple soft IP design

E.2.1 UPF modeling for a hard IP

Replace the entire example code with the following:

```
# Start of hard IP UPF, assume file name is cellA.upf
begin_power_model upf_macroA \
 -for {cellA}

# Section 1: define the interfaces of hard IP power model
create_power_domain PD \
 -elements {.}  \
```

```
 -supply {SSAH} \
 -supply {SSBH} \
 -supply {SSBH_SW}

# Section 2: associate the interface supplies to boundary supply ports
#           or internally generated supplies
create_supply_set PD.SSAH \
 -update \
 -function {power vddA} \
 -function {ground vssA}

create_supply_set PD.SSBH \
 -update \
 -function {power vddB} \
 -function {ground vssB}

# special handle for internally generated supply
create_supply_net vddB_int

create_supply_set PD.SSBH_SW \
 -update \
 -function {power vddB_int} \
 -function {ground vssB}

create_power_switch internal_sw \
 -output_supply_port {out  SSBH_SW.power} \
 -input_supply_port  {in   vddB} \
 -control_port       {ctrl Y} \
 -on_state        {ON in   !ctrl}

# Section 3: define data port and interface supply set handle
associations
set_port_attributes \
 -ports {W X} \
 -receiver_supply PD.SSAH
set_port_attributes \
 -ports {Y Z} \
 -receiver_supply PD.SSBH
set_port_attributes \
 -ports {Z1 } \
 -driver_supply   PD.SSBH_SW
set_port_attributes \
 -ports {Z2 } \
 -driver_supply   PD.SSBH

# Section 4: define power states for interface supply set handles
add_power_state PD.SSAH \
 -supply \
 -state {ON  \
   -simstate NORMAL \
   -supply_expr {power=={FULL_ON 0.7 0.9} && ground=={FULL_ON 0}}} \
 -state {OFF \
   -simstate CORRUPT \
   -supply_expr {power == OFF || ground == OFF}}

add_power_state PD.SSBH \
 -supply\
```

```
 -state {ON  \
   -simstate NORMAL \
   -supply_expr {power=={FULL_ON 1.1 1.3} && ground=={FULL_ON 0}}} \
 -state {OFF \
   -simstate CORRUPT \
   -supply_expr {power == OFF || ground == OFF}}

add_power_state PD.SSBH_SW \
 -supply \
 -state {ON  \
   -simstate NORMAL  \
   -supply_expr {power=={FULL_ON 1.1 1.3} && ground=={FULL_ON 0}}} \
 -state {OFF \
   -simstate CORRUPT \
   -supply_expr {power == OFF || ground == OFF}}

# Section 5: define system power states of the hard IP
add_power_state PD \
 -domain\
 -state {S1 \
   -logic_expr {SSAH == ON  && SSBH == ON  && SSBH_SW == ON }} \
 -state {S2 \
   -logic_expr {SSAH == ON  && SSBH == ON  && SSBH_SW == OFF}} \
 -state {S3 \
   -logic_expr {SSAH == ON  && SSBH == OFF && SSBH_SW == OFF}} \
 -state {S4 \
   -logic_expr {SSAH == OFF && SSBH == OFF && SSBH_SW == OFF}}

# Section 6: Define internal low power logic
set_isolation internal_iso \
 -domain PD \
 -elements {X} \
 -isolation_signal {W} \
 -isolation_sense {low}

# Section 7: Define hard IP level isolation constraints
set_port_attributes \
 -ports {W Z} \
 -clamp_value 1
set_port_attributes \
 -ports {Y} \
 -clamp_value 0
end_power_model

# End of hard IP UPF cellA.upf
```

E.2.2 UPF modeling for a soft IP

Replace the entire example code with the following:

```
# Start of top level configuration UPF, assume the file name of
top_soft.upf

# Section 1: load hard IP power models
```

```
load_upf cellA.upf

# Section 2: define the control ports for special low power logic
create_logic_port sw_en1  \
 -direction in ; # power switch enable for PDTop
create_logic_port sw_en2  \
 -direction in ; # power switch enable for PD1
create_logic_port iso_en1 \
 -direction in
create_logic_port iso_en2 \
 -direction in
create_logic_port ret_en  \
 -direction in

# Section 3: define power domains and interface supply set handles
create_power_domain PDTop \
 -elements {.} \
 -supply {SSH1} \
 -supply {SSH2} ;# interface supply set handles
create_power_domain PD1 \
 -elements {I3}

# Section 4: integrate hard IP
apply_power_model upf_modelA \
 -elements I2 \
 -supply_map {{PD.SSAH PDTop.SSH1} \
              {PD.SSBH PDTop.SSH2}}

# Section 5: define power states for supply set handles
add_power_state PDTop.SSH1 \
 -supply \
 -state {ON  \
   -simstate NORMAL \
   -supply_expr {power == FULL_ON && ground == FULL_ON}}\
 -state {OFF \
   -simstate CORRUPT \
   -supply_expr {power == OFF     || ground == OFF}}

add_power_state PDTop.SSH2 \
 -state {ON  \
   -simstate NORMAL \
   -supply_expr {power == FULL_ON && ground == FULL_ON}}\
 -state {OFF \
   -simstate CORRUPT \
   -supply_expr {power == OFF     || ground == OFF}}

add_power_state PDTop.primary \
 -supply \
 -state {ON  \
   -simstate NORMAL \
   -supply_expr {power == FULL_ON && ground == FULL_ON}}\
 -state {OFF \
   -simstate CORRUPT \
   -logic_expr {sw_en1}}


add_power_state PD1.primary \
```

```
 -supply \
 -state {ON  \
   -simstate NORMAL \
   -supply_expr {power == FULL_ON && ground == FULL_ON}}\
 -state {OFF \
   -simstate CORRUPT \
   -logic_expr {sw_en2}}

# Section 6: define system power states of the soft IP
add_power_state PDTop \
 -domain\
 -state {S1 \
   -logic_expr {SSH1 == ON  && SSH2 == ON  \
              && primary == ON  && PD1.primary == ON  \
              && I2/PD   == S1}} \
 -state {S2 \
   -logic_expr {SSH1 == ON  && SSH2 == ON  \
              && primary == ON  && PD1.primary == OFF \
              && I2/PD == S1}} \
 -state {S3 \
   -logic_expr {SSH1 == ON  && SSH2 == ON  \
              && primary == OFF && PD1.primary == OFF \
              && I2/PD == S1}} \
 -state {S4 \
   -logic_expr {SSH1 == ON  && SSH2 == ON  \
              && primary == ON  && PD1.primary == ON  \
              && I2/PD == S2}} \
 -state {S5 \
   -logic_expr {SSH1 == ON  && SSH2 == ON  \
              && primary == ON  && PD1.primary == OFF \
              && I2/PD == S2}} \
 -state {S6 \
   -logic_expr {SSH1 == ON  && SSH2 == ON  \
              && primary == OFF && PD1.primary == OFF \
              && I2/PD == S2}} \
 -state {S7 \
   -logic_expr {SSH1 == ON  && SSH2 == OFF \
              && primary == OFF && PD1.primary == OFF \
              && I2/PD == S3}} \
 -state {S8 \
   -logic_expr {SSH1 == ON  && SSH2 == OFF \
              && primary == OFF && PD1.primary == OFF \
              && I2/PD == S4}} \
 -state {S9 \
   -logic_expr {SSH1 == OFF && SSH2 == OFF \
              && primary == OFF && PD1.primary == OFF \
              && I2/PD == S4}}

# Section 7: define isolation strategies
set_isolation iso1 \
 -domain PDTop \
 -applies_to outputs \
 -diff_supply_only TRUE \
 -isolation_signal iso_en1 \
 -isolation_sense high \
 -clamp_value 0 \
 -isolation_supply_set PDTop.SSH1 \
```

```
 -location self

set_isolation iso2 \
 -domain PD1 \
 -source PD1.primary \
 -diff_supply_only TRUE \
 -isolation_signal iso_en2 \
 -isolation_sense high \
 -clamp_value 0 \
 -isolation_supply_set PDTop.SSH1 \
 -location parent

set_isolation iso3 \
 -domain PDTop \
 -source PDTop.primary \
 -diff_supply_only TRUE \
 -sink PDTop.SSH1\
 -isolation_signal iso_en1 \
 -isolation_sense high \
 -clamp_value 1 \
 -isolation_supply_set PDTop.SSH1 \
 -location self

set_isolation iso4 \
 -domain PDTop \
 -source I2/SSBH_SW \
 -diff_supply_only TRUE \
 -isolation_signal in2 \
 -isolation_sense high \
 -clamp_value 1 \
 -isolation_supply_set PDTop.SSH2 \
 -location parent

# section 8: define retention strategies
set_retention ret1 \
 -domain PD1 \
 -save_signal {ret_en negedge} \
 -restore_signal {ret_en posedge} \
 -retention_supply_set PDTop.SSH1

# section 9: define soft IP level isolation constraint
set_port_attributes \
 -domain { . } \
 -applies_to inputs \
 -clamp_value 0 \
 -exclude_ports {in3}

set_port_attributes \
 -ports {in3} \
 -clamp_value 1

# end of Top level configuration UPF, top_soft.upf
```

E.2.3 How to use configuration UPF

Replace Figure E.4 with the following figure:

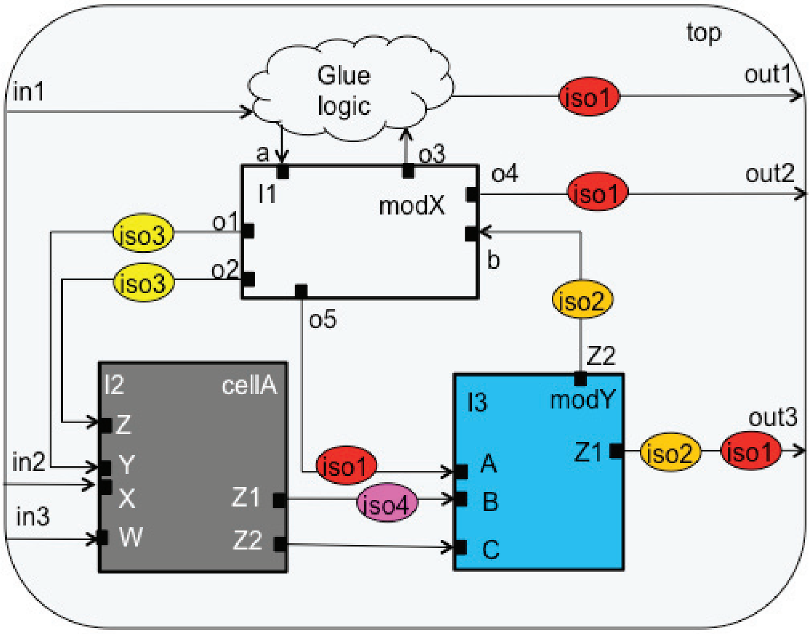


Figure E.4—Block example with isolation logic inserted

E.3.1 Logic implementation UPF for the soft IP

Replace the entire example code with the following:

```
# Start of incrementation implementation UPF
# assume the file name of top_impl.upf

# Section 1: Add voltage information for supply set states
add_power_state PDTop.SSH1 \
 -supply \
 -state {ON \
   -supply_expr {power== {FULL_ON 0.8}}} \
   -update

add_power_state PDTop.SSH2 \
 -supply \
 -state {ON \
   -supply_expr {power== {FULL_ON 0.8 1.2}}} \
   -update

add_power_state PDTop.primary \
 -supply \
 -state {ON \
   -supply_expr {power== {FULL_ON 0.8}}} \
```

```
   -update

add_power_state PD1.primary \
 -supply \
 -state {ON \
   -supply_expr {power== {FULL_ON 0.8}}} \
   -update

# Section 2: Add level-shifting strategy
set_level_shifter lvl1 \
 -domain PDTop \
 -source PDTop.primary \
 -sink PDTop.SSH2 \
 -input_supply_set PDTop.primary \
 -output_supply_set PDTop.SSH2

set_level_shifter lvl2 \
 -domain PD1 \
 -source PD1.primary \
 -sink PDTop.SSH2 \
 -input_supply_set PD1.primary \
 -output_supply_set PDTop.SSH2

# Section 3: Add library info for retention strategy
map_retention_cell ret1 \
 -domain PD1 \
 -lib_cells {my_ret_cell1 my_ret_cell2}

# Section 4: Add library info for isolation strategy
use_interface_cell iso1_cells \
 -domain PDTop \
 -strategy iso1 \
 -lib_cells {my_iso_cell1}

use_interface_cell iso1_cells \
 -domain PDTop \
 -strategy iso2 \
 -lib_cells {my_iso_cell1}

use_interface_cell iso1_cells \
 -domain PDTop \
 -strategy iso4 \
 -lib_cells {my_iso_cell12}

# Section 5: Add library info for level-shifting strategy
use_interface_cell lvl1_cells \
 -strategy lvl1 \
 -domain PDTop \
 -lib_cells {my_lvl_cell1 my_lvl_cell2}

use_interface_cell lvl2_cells \
 -strategy lvl2 \
 -domain PD1 \
 -lib_cells {my_lvl_cell1 my_lvl_cell2}

# Section 6: Add library info for combined level-shifting and isolation
cells
```

```
use_interface_cell enabled_lvl \
 -domain PDTop \
 -strategy {iso3 lvl1} \
 -lib_cells {en_lvl}

# end of incrementation implementation UPF top_impl.upf
```

E.3.2 How to use implementation UPF

Replace Figure E.5 with the following figure:

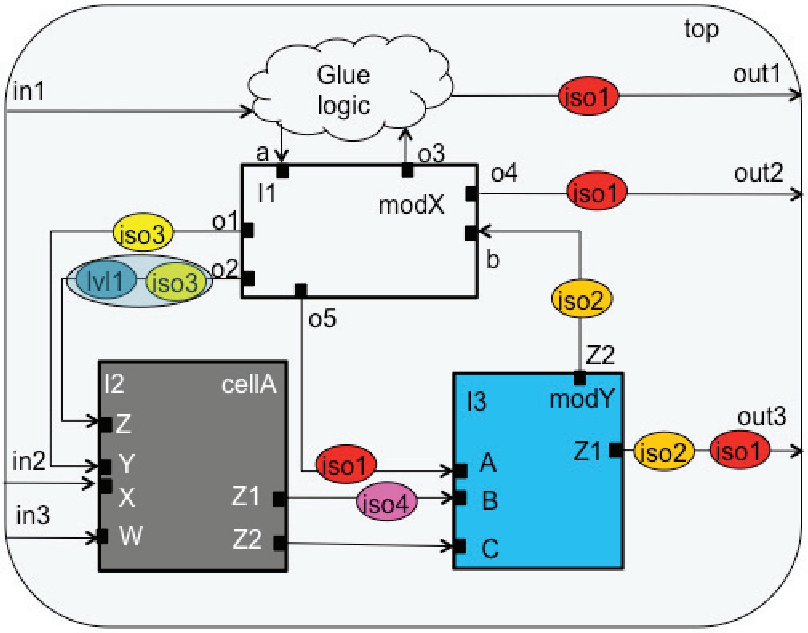


Figure E.5—Block example with isolation and level-shifting logic inserted after synthesis

E.4.1 Physical Implementation UPF for the Soft IP

Change the file name in last sentence of the second paragraph as follows:

Designers can either add the `source top_phy.upf` command at the end of the UPF file `top_impl.upf` or add the `source top_impl.upf` command at the start of the UPF file ~~`top_impl.upf`~~ `top_phy.upf` or create a new UPF file with the following commands:

Replace the entire example code with the following:

```
# Start of physical implementation UPF
# assume the file name of top_phy.upf

# Section 1: define supply ports
create_supply_port vdd1
```

```
create_supply_port vss1
create_supply_port vdd2
create_supply_port vss2

# Section 2: define supply nets
create_supply_net vdd_top \
 -domain PDTop
create_supply_net vdd1_sw \
 -domain PD1

# Section 3: associate supply nets to supply sets
create_supply_set ss1 \
 -function {power vdd1} \
 -function {ground vss1}
associate_supply_set ss1 \
 -handle PDTop.SSH1

create_supply_set ss2 \
 -function {power vdd2} \
 -function {ground vss2}
associate_supply_set ss2 \
 -handle PDTop.SSH2

create_supply_set PDTop_ss \
 -function {power vdd_top} \
 -function {ground vss1}
associate_supply_set PDTop_ss \
 -handle PDTop.primary

create_supply_set PD1_ss \
 -function {power vdd1_sw} \
 -function {ground vss1}
associate_supply_set PD1_ss \
 -handle PD1.primary

# Section 4: define power switch strategy
create_power_switch PDTop_switch \
 -output_supply_port  {sw_out vdd_top} \
 -input_supply_port   {sw_in vdd1  } \
 -control_port        {pso sw_en1  } \
 -on_state    {top_on  sw_in {!pso}} \
 -off_state   {top_off       {pso}}

create_power_switch PD1_switch \
 -output_supply_port {sw_out vdd1_sw} \
 -input_supply_port  {sw_in  vdd1 } \
 -control_port       {pso    sw_en2} \
 -on_state   {top_on  sw_in {!pso}} \
 -off_state  {top_off       {pso}}

# end of physical implementation UPF top_phy.upf
```

E.4.3 How to use physical implementation UPF

Replace Figure E.6 with the following figure:

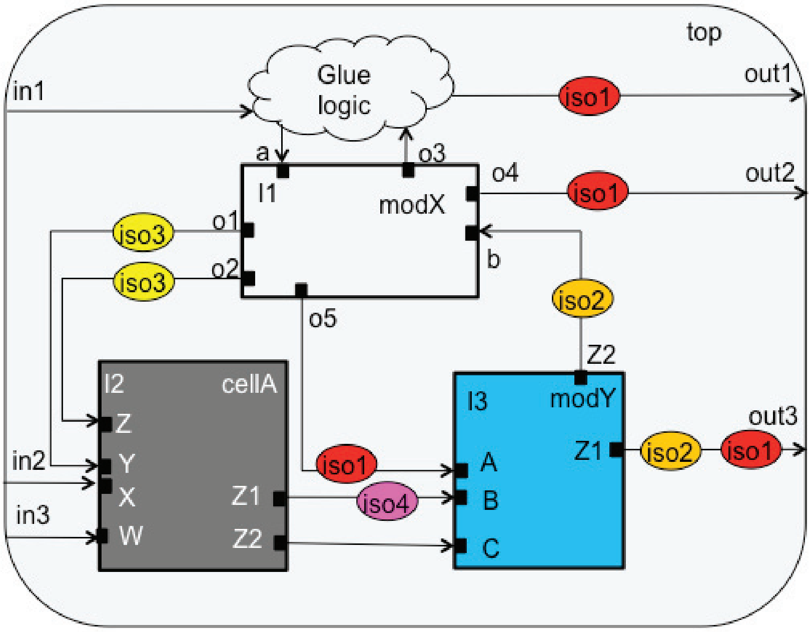


Figure E.6—Block example diagram after physical implementation

E.4.3.2 Supply net connection

Change the last sentence of bullet 1 in paragraph 2 as follows:

If the isolation strategy has no **-isolation_supply** option specified, the default isolation supply associated with the ~~reference~~ location domain is used.

E.6.1 UPF constraints

Replace the entire example code with the following:

```
# Start of top level Constraint UPF
# assume the file name of top_constr.upf

# Section 1: define power domains
create_power_domain PDTop -elements {.}
create_power_domain PD1 \
 -elements { I3 }

# Section 2: define supply set states
add_power_state PDTop.primary \
 -supply \
```

```
 -state {ON \
   -simstate  NORMAL \
   -supply_expr {power == FULL_ON && ground == FULL_ON}} \
 -state {OFF \
   -simstate CORRUPT \
   -supply_expr {power == OFF || ground == OFF}}
add_power_state PD1.primary \
 -supply \
 -state {ON \
   -simstate NORMAL\
   -supply_expr {power == FULL_ON && ground == FULL_ON}} \
 -state {OFF \
   -simstate CORRUPT\
   -supply_expr {power == OFF || ground == OFF}}

# Section 3: define system level power states
add_power_state PDTop \
 -domain \
 -state {CS1 \
   -logic_expr {primary == ON && PD1.primary == ON}} \
 -state {CS2 \
   -logic_expr {primary == ON && PD1.primary == OFF}} \
 -state {CS3 \
   -logic_expr {primary == OFF && PD1.primary == OFF}}

# Section 4: define isolation constraints
set_port_attributes \
 -elements { . } \
 -applies_to inputs \
 -clamp_value 0
set_port_attributes \
 -elements { I3 } \
 -applies_to inputs \
 -clamp_value 0
set_port_attributes \
 -ports {in3} \
 -clamp_value 1

# Section 5: define state retention constraints
set_retention_elements ret_list \
 -elements I3 \
 -transitive TRUE

# end of Top level Constraint UPF, top_constr.upf
```

E.6.2 Configure a constraint UPF into a configuration UPF

Replace the entire example code with the following:

```
# A configuration UPF configured from the constraint UPF,
# assume the file name of top_soft2.upf

source top_constr.upf
source cellA.upf
```

```
create_logic_port sw_en1 \
 -direction in
create_logic_port sw_en2 \
 -direction in
create_logic_port iso_en1 \
 -direction in
create_logic_port iso_en2 \
 -direction in
create_logic_port retention \
 -direction in

create_supply_set ss1
create_supply_set ss2

create_power_domain PDTop \
 -supply SSH1\
 -supply SSH2\
 -update

create_power_switch PDTop_sw \
 -output_supply_port {out  PDTop.primary.power} \
 -input_supply_port  {in   PDTop.SSH1.power}\
 -control_port       {ctrl sw_en1} \
 -on_state        {ON in   !ctrl}

create_power_switch PD1_sw\
 -output_supply_port {out  PD1.primary.power} \
 -input_supply_port  {in   PDTop.SSH1.power} \
 -control_port       {ctrl sw_en2} \
 -on_state        {ON in   !ctrl}

apply_power_model upf_modelA \
 -elements I2 \
 -supply_map {{PD.SSAH PDTop.SSH1} \
              {PD.SSBH PDTop.SSH2}}

add_power_state PDTop.SSH1 \
 -state {ON\
   -simstate NORMAL\
   -supply_expr {power == FULL_ON && ground == FULL_ON}} \
 -state {OFF\
   -simstate CORRUPT\
   -supply_expr {power == OFF || ground == OFF}}

add_power_state PDTop.SSH2 \
 -state {ON \
   -simstate NORMAL \
   -supply_expr {power == FULL_ON && ground == FULL_ON}} \
 -state {OFF \
   -simstate CORRUPT \
   -supply_expr {power == OFF || ground == OFF}}

add_power_state PDTop \
 -state {S1 \
   -logic_expr { SSH1 == ON && SSH2 == ON \
               && primary == ON && PD1.primary == ON \
               && I2/PD == S1}} \
```

```
 -state {S2 \
   -logic_expr { SSH1 == ON && SSH2 == ON \
               && primary == ON && PD1.primary == OFF \
               && I2/PD == S1}} \
 -state {S3 \
   -logic_expr { SSH1 == ON && SSH2 == ON \
               && primary == OFF && PD1.primary == OFF \
               && I2/PD == S1}} \
 -state {S4 \
   -logic_expr { SSH1 == ON && SSH2 == ON \
               && primary == ON && PD1.primary == ON \
               && I2/PD == S2}} \
 -state {S5 \
   -logic_expr { SSH1 == ON && SSH2 == ON \
               && primary == ON && PD1.primary == OFF \
               && I2/PD == S2}} \
 -state {S6 \
   -logic_expr { SSH1 == ON && SSH2 == ON \
               && primary == OFF && PD1.primary == OFF \
               && I2/PD == S2}} \
 -state {S7 \
   -logic_expr { SSH1 == ON && SSH2 == OFF \
               && primary == OFF && PD1.primary == OFF \
               && I2/PD == S3}} \
 -state {S8 \
   -logic_expr { SSH1 == ON && SSH2 == OFF \
               && primary == OFF && PD1.primary == OFF \
               && I2/PD == S4}} \
 -state {S9 \
   -logic_expr { SSH1 == OFF && SSH2 == OFF \
               && primary == OFF && PD1.primary == OFF \
               && I2/PD == S4}}

set_isolation iso1 \
 -domain PDTop \
 -applies_to outputs \
 -isolation_supply_set PDTop.SSH1 \
 -location self \
 -isolation_signal iso_en1 \
 -isolation_sense high \
 -clamp_value 0\
 -diff_supply_only TRUE

set_isolation iso2 \
 -domain PD1 \
 -source PD1.primary \
 -isolation_supply_set PDTop.SSH1 \
 -location parent \
 -isolation_signal iso_en2 \
 -isolation_sense high \
 -clamp_value 0 \
 -diff_supply_only TRUE

set_isolation iso3 \
 -domain PDTop \
 -source PDTop.primary \
 -sink PDTop.SSH1 \
```

```
 -isolation_supply_set PDTop.SSH1 \
 -location self \
 -isolation_signal iso_en1 \
 -isolation_sense high \
 -clamp_value 1 \
 -diff_supply_only TRUE

set_isolation iso4\
 -domain PDTop \
 -source I2/SSBH_SW \
 -isolation_supply_set PDTop.SSH2 \
 -location parent \
 -isolation_signal in2 \
 -isolation_sense high \
 -clamp_value 1 \
 -diff_supply_only TRUE

set_retention ret1 \
 -domain PD1 \
 -retention_supply_set PDTop.SSH1 \
 -elements ret_list \
 -save_signal {ret_en negedge} \
 -restore_signal {ret_en posedge}

# end of top_soft2.upf
```

Consensus

WE BUILD IT.

IEEE
standards.ieee.org
Phone: +1 732 981 0060   Fax: +1 732 562 1571