

# IEEE Standard for Design and Verification of Low-Power Integrated Circuits

IEEE Computer Society

Sponsored by the  
Design Automation Committee  
and the  
IEEE Standards Association Corporate Advisory Group



---

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

29 May 2013

**IEEE Std 1801™-2013**  
(Revision of IEEE Std 1801-2009)



**This is a Redline Document produced by Techstreet, a business of Thomson Reuters.**

This document is intended to provide users with an indication of changes from one edition to the next. It includes a full-text version of the new document, plus an indication of changes from the previous version.

Redlines are designed to save time and improve efficiencies by using the latest software technology to find and highlight document changes. More professionals are using valuable new technologies like redlines, to help improve outcomes in a fastpaced global business world.

Because it may not be technically possible to capture all changes accurately, it is recommended that users consult previous editions as appropriate. In all cases, only the current base version of this publication is to be considered the official document.

## Redline Processing Notes:

1. ~~Red Text~~ - Red strikethrough text denotes deletions.
2. Blue Text - Blue underlined text denotes modifications and additions.

**IEEE Std 1801™-2013**

(Revision of  
IEEE Std 1801-2009)

# IEEE Standard for Design and Verification of Low-Power Integrated Circuits

Sponsor

**Design Automation Committee**

of the

**IEEE Computer Society**

and the

**IEEE Standards Association Corporate Advisory Group**

Approved 6 March 2013

**IEEE-SA Standards Board**

Grateful acknowledgment is made to the following for permission to use source material:

Accellera Systems Initiative

Unified Power Format (UPF) Standard, Version 1.0

Cadence Design Systems, Inc.

Library Cell Modeling Guide Using CPF

Hierarchical Power Intent Modeling Guide Using CPF

Silicon Integration Initiative, Inc.

Si2 Common Power Format Specification, Version 2.0

**Abstract:** A method is provided for specifying power intent for an electronic design, for use in verification of the structure and behavior of the design in the context of a given power management architecture, and for driving implementation of that power management architecture. The method supports incremental refinement of power intent specifications required for IP-based design flows.

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

---

The Institute of Electrical and Electronics Engineers, Inc.  
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2013 by The Institute of Electrical and Electronics Engineers, Inc.  
All rights reserved. Published 29 May 2013. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

Verilog is a registered trademark of Cadence Design Systems, Inc.

Print: ISBN 978-0-7381-8282-7 STDPD98167  
PDF: ISBN 978-0-7381-8281-0 STDGT98167

*IEEE prohibits discrimination, harassment and bullying.*

For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.

*No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.*

**Notice and Disclaimer of Liability Concerning the Use of IEEE Documents:** IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. 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.

Use of an IEEE Standard is wholly voluntary. IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon any IEEE Standard document.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied “AS IS.”

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. 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 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.

**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 the official position of IEEE or any of its committees and shall not be considered to be, nor 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 to ensure 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. Any person who would like to participate in evaluating comments or revisions to an IEEE standard is welcome to join the relevant IEEE working group at <http://standards.ieee.org/develop/wg/>.

Comments on standards should be submitted to the following address:

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

**Photocopies:** Authorization to photocopy portions of any individual standard for internal or personal use is granted by The Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, 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.

## Notice to users

## 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

This document is copyrighted by the IEEE. It is made available 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 this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

## Updating of IEEE 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. 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://standards.ieee.org/index.html> or contact the IEEE at the address listed previously. For more information about the IEEE Standards Association or the IEEE standards development process, visit IEEE-SA Website at <http://standards.ieee.org/index.html>.

## Errata

Errata, if any, for this and all other standards can be accessed 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

The Unified Power Format Working Group is entity based. At the time this standard was completed, the Unified Power Format Working Group had the following membership:

**John Biggs**, *Chair*  
**Erich Marschner**, *Vice Chair*  
**Jeffrey Lee**, *Secretary*  
**Joe Daniels**, *Technical Editor*

Dave Allen  
Ido Bourstein  
Shir-Shen Chang  
David Cheng  
Cary Chin  
Sumit DasGupta  
Sorin Dobre  
Shaun Durnan  
Colin Holehouse

Sushma Honnavara-Prasad  
Fred Jen  
Tim Jordan  
Knut Just  
James Kehoe  
Rick Koster  
Rolf Lagerquist  
Lisa McIlwain  
Don Mills  
Barry Pangrle

Albert Rich  
Judith Richardson  
Jim Sproch  
Amit Srivastava  
Prasad Subbarao  
Venki Venkatesh  
Qi Wang  
Jon Worthington  
Louis Yu

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

Accellera Systems Initiative  
Advanced Micro Devices (AMD)  
ARM, Ltd.  
Atrenta Inc.  
Broadcom Corporation  
Cadence Design Systems, Inc.  
Cambridge Silicon Radio  
Cortina Systems  
Intel Corporation  
Japan Electronics and Information Technology  
Industries Association (JEITA)

LSI Corporation  
Marvell Technology Group Ltd.  
MediaTek Inc.  
Mentor Graphics Corporation  
Qualcomm Incorporated  
Silicon Integration Initiative, Inc.  
STMicroelectronics  
Synopsys, Inc.  
Texas Instruments Incorporated  
Xilinx

When the IEEE-SA Standards Board approved this standard on 6 March 2013, it had the following membership:

**John Kulick, *Chair***  
**David J. Law, *Vice Chair***  
**Richard H. Hulet, *Past Chair***  
**Konstantinos Karachalios, *Secretary***

Masayuki Ariyoshi  
Peter Balma  
Farooq Bari  
Ted Burse  
Wael William Diab  
Stephen Dukes  
Jean-Philippe Faure  
Alexander Gelman

Mark Halpin  
Gary Hoffman  
Paul Houzé  
Jim Hughes  
Michael Janezic  
Joseph L. Koepfinger\*  
Oleg Logvinov

Ron Petersen  
Gary Robinson  
Jon Walter Rosdahl  
Adrian Stephens  
Peter Sutherland  
Yatin Trivedi  
Phil Winston  
Yu Yuan

\*Member Emeritus

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

Richard DeBlasio, *DOE Representative*  
Michael Janezic, *NIST Representative*

Julie Alessi  
*IEEE Standards Program Manager, Document Development*

Krista Gluchoski  
*IEEE Standards Program Manager, Technical Program Development*

# Introduction

This introduction is not part of IEEE Std 1801-2013, IEEE Standard for Design and Verification of Low-Power Integrated Circuits.

The purpose of this standard is to provide portable low-power design specifications that can be used with a variety of commercial products throughout an electronic system design, analysis, verification, and implementation flow.

When the electronic design automation (EDA) industry began creating standards for use in specifying, simulating, and implementing functional specifications of digital electronic circuits in the 1980s, the primary design constraint was the transistor area necessary to implement the required functionality in the prevailing process technology at that time. Power considerations were simple and easily assumed for the design as power consumption was not a major consideration and most chips operated on a single voltage for all functionality. Therefore, hardware description languages (HDLs) such as VHDL (IEC 61691-1-1/IEEE Std 1076™)<sup>a</sup> and SystemVerilog (IEEE Std 1800™) provided a rich set of capabilities necessary for capturing the functional specification of electronic systems, but no capabilities for capturing the power architecture (how each element of the system is to be powered).

As the process technology for manufacturing electronic circuits continued to advance, power (as a design constraint) continually increased in importance. Even above the 90 nm process node size, dynamic power consumption became an important design constraint as the functional size of designs increased power consumption at the same time battery-operated mobile systems, such as cell phones and laptop computers, became a significant driver of the electronics industry. Techniques for reducing dynamic power consumption—the amount of power consumed to transition a node from a 0 to 1 state or vice versa—became commonplace. Although these techniques affected the design methodology, the changes were relatively easy to accommodate within the existing HDL-based design flow, as these techniques were primarily focused on managing the clocking for the design (more clock domains operating at different frequencies and gating of clocks when logic in a clock domain is not needed for the active operational mode). Multi-voltage power-management methods were also developed. These methods did not directly impact the functionality of the design, requiring only level-shifters between different voltage domains. Multi-voltage power domains could be verified in existing design flows with additional, straight-forward extensions to the methodology.

With process technologies below 100 nm, static power consumption has become a prominent and, in many cases, dominant design constraint. Due to the physics of the smaller process nodes, power is leaked from transistors even when the circuitry is quiescent (no toggling of nodes from 0 to 1 or vice versa). New design techniques were developed to manage static power consumption. Power gating or power shut-off turns off power for a set of logic elements. Back-bias techniques are used to raise the voltage threshold at which a transistor can change its state. While back bias slows the performance of the transistor, it greatly reduces leakage. These techniques are often combined with multi-voltages and require additional functionality: power-management controllers, isolation cells that logically and/or electrically isolate a shutdown power domain from “powered-up” domains, level-shifters that translate signal voltages from one domain to another, and retention registers to facilitate fast transition from a power-off state to a power-on state for a domain.

The EDA industry responded with multiple vendors developing proprietary low-power specification capabilities for different tools in the design and implementation flow. Although this solved the problem locally for a given tool, it was not a global solution in that the same information was often required to be specified multiple times for different tools without portability of the power specification. At the Design

<sup>a</sup>Information on references can be found in Clause 2.

Automation Conference (DAC) in June 2006, several semiconductor/electronics companies challenged the EDA industry to define an open, portable power specification standard. The EDA industry standards incubation consortium, Accellera Systems Initiative, answered the call by creating a Technical SubCommittee (TSC) to develop a standard. The effort was named Unified Power Format (UPF) to recognize the need of unifying the capabilities of multiple proprietary formats into a single industry standard. Accellera approved *UPF 1.0* as an Accellera standard in February 2007. In May 2007, Accellera donated UPF to the IEEE for the purposes of creating an IEEE standard, and in March 2009, the first version of the IEEE Std 1801 was released. So this standard, although the second version of the IEEE Std 1801, represents the third version of what is more colloquially referred to as UPF.



# Contents

|      |                                                      |    |
|------|------------------------------------------------------|----|
| 1.   | Overview.....                                        | 1  |
| 1.1  | Scope.....                                           | 1  |
| 1.2  | Purpose.....                                         | 1  |
| 1.3  | Key characteristics of the Unified Power Format..... | 1  |
| 1.4  | Use of color in this standard .....                  | 3  |
| 1.5  | Contents of this standard.....                       | 3  |
| 2.   | Normative references .....                           | 4  |
| 3.   | Definitions, acronyms, and abbreviations.....        | 4  |
| 3.1  | Definitions .....                                    | 4  |
| 3.2  | Acronyms and abbreviations .....                     | 9  |
| 4.   | UPF concepts .....                                   | 11 |
| 4.1  | Design structure .....                               | 11 |
| 4.2  | Design representation .....                          | 11 |
| 4.3  | Power architecture .....                             | 14 |
| 4.4  | Power distribution.....                              | 17 |
| 4.5  | Power management.....                                | 23 |
| 4.6  | Power states .....                                   | 26 |
| 4.7  | Simstates .....                                      | 29 |
| 4.8  | Successive refinement.....                           | 30 |
| 4.9  | Tool flow.....                                       | 31 |
| 4.10 | File structure .....                                 | 32 |
| 5.   | Language basics .....                                | 33 |
| 5.1  | UPF is Tcl.....                                      | 33 |
| 5.2  | Conventions used.....                                | 33 |
| 5.3  | Lexical elements .....                               | 34 |
| 5.4  | Boolean expressions .....                            | 37 |
| 5.5  | Object declaration.....                              | 39 |
| 5.6  | Attributes of objects.....                           | 40 |
| 5.7  | Power state name spaces.....                         | 43 |
| 5.8  | Precedence .....                                     | 44 |
| 5.9  | Generic UPF command semantics.....                   | 45 |
| 5.10 | effective_element_list semantics .....               | 45 |
| 5.11 | Command refinement .....                             | 48 |
| 5.12 | Error handling .....                                 | 49 |
| 5.13 | Units.....                                           | 50 |
| 6.   | Power intent commands.....                           | 51 |
| 6.1  | Categories .....                                     | 51 |
| 6.2  | add_domain_elements [deprecated] .....               | 51 |
| 6.3  | add_port_state [legacy] .....                        | 52 |
| 6.4  | add_power_state .....                                | 52 |
| 6.5  | add_pst_state [legacy] .....                         | 57 |
| 6.6  | apply_power_model.....                               | 58 |
| 6.7  | associate_supply_set .....                           | 59 |

|      |                                           |     |
|------|-------------------------------------------|-----|
| 6.8  | begin_power_model .....                   | 60  |
| 6.9  | bind_checker .....                        | 61  |
| 6.10 | connect_logic_net .....                   | 63  |
| 6.11 | connect_supply_net .....                  | 64  |
| 6.12 | connect_supply_set .....                  | 65  |
| 6.13 | create_composite_domain .....             | 67  |
| 6.14 | create_hdl2upf_vct .....                  | 68  |
| 6.15 | create_logic_net .....                    | 69  |
| 6.16 | create_logic_port .....                   | 70  |
| 6.17 | create_power_domain .....                 | 71  |
| 6.18 | create_power_switch .....                 | 74  |
| 6.19 | create_pst [legacy] .....                 | 80  |
| 6.20 | create_supply_net .....                   | 80  |
| 6.21 | create_supply_port .....                  | 83  |
| 6.22 | create_supply_set .....                   | 84  |
| 6.23 | create_upf2hdl_vct .....                  | 85  |
| 6.24 | describe_state_transition .....           | 86  |
| 6.25 | end_power_model.....                      | 87  |
| 6.26 | find_objects .....                        | 88  |
| 6.27 | load_simstate_behavior .....              | 90  |
| 6.28 | load_upf .....                            | 91  |
| 6.29 | load_upf_protected .....                  | 92  |
| 6.30 | map_isolation_cell [deprecated] .....     | 93  |
| 6.31 | map_level_shifter_cell [deprecated] ..... | 93  |
| 6.32 | map_power_switch .....                    | 93  |
| 6.33 | map_retention_cell .....                  | 94  |
| 6.34 | merge_power_domains [deprecated] .....    | 97  |
| 6.35 | name_format .....                         | 98  |
| 6.36 | save_upf .....                            | 99  |
| 6.37 | set_design_attributes .....               | 100 |
| 6.38 | set_design_top .....                      | 101 |
| 6.39 | set_domain_supply_net [legacy] .....      | 101 |
| 6.40 | set_equivalent .....                      | 102 |
| 6.41 | set_isolation .....                       | 104 |
| 6.42 | set_isolation_control [deprecated] .....  | 110 |
| 6.43 | set_level_shifter .....                   | 111 |
| 6.44 | set_partial_on_translation .....          | 116 |
| 6.45 | set_pin_related_supply [deprecated] ..... | 116 |
| 6.46 | set_port_attributes .....                 | 117 |
| 6.47 | set_power_switch [deprecated] .....       | 121 |
| 6.48 | set_repeater .....                        | 121 |
| 6.49 | set_retention .....                       | 124 |
| 6.50 | set_retention_control [deprecated] .....  | 128 |
| 6.51 | set_retention_elements .....              | 128 |
| 6.52 | set_scope .....                           | 129 |
| 6.53 | set_simstate_behavior .....               | 130 |
| 6.54 | upf_version .....                         | 131 |
| 6.55 | use_interface_cell .....                  | 132 |
| 7.   | Power management cell commands.....       | 135 |
| 7.1  | Introduction.....                         | 135 |
| 7.2  | define_always_on_cell.....                | 136 |
| 7.3  | define_diode_clamp.....                   | 137 |

|     |                                                                                        |     |
|-----|----------------------------------------------------------------------------------------|-----|
| 7.4 | define_isolation_cell .....                                                            | 138 |
| 7.5 | define_level_shifter_cell .....                                                        | 141 |
| 7.6 | define_power_switch_cell .....                                                         | 145 |
| 7.7 | define_retention_cell .....                                                            | 147 |
| 8.  | UPF processing .....                                                                   | 150 |
| 8.1 | Overview .....                                                                         | 150 |
| 8.2 | Data requirements .....                                                                | 150 |
| 8.3 | Processing phases .....                                                                | 150 |
| 8.4 | Error checking .....                                                                   | 153 |
| 9.  | Simulation semantics .....                                                             | 154 |
| 9.1 | Supply network creation .....                                                          | 154 |
| 9.2 | Supply network simulation .....                                                        | 155 |
| 9.3 | Power state simulation .....                                                           | 157 |
| 9.4 | Simstate simulation .....                                                              | 159 |
| 9.5 | Transitioning from one simstate state to another .....                                 | 161 |
| 9.6 | Simulation of retention .....                                                          | 162 |
| 9.7 | Simulation of isolation .....                                                          | 168 |
| 9.8 | Simulation of level-shifting .....                                                     | 168 |
| 9.9 | Simulation of repeater .....                                                           | 168 |
|     | Annex A (informative) Bibliography .....                                               | 169 |
|     | Annex B (normative) HDL package UPF .....                                              | 170 |
|     | Annex C (normative) Queries .....                                                      | 182 |
|     | Annex D (informative) Replacing deprecated and legacy commands and options .....       | 219 |
|     | Annex E (informative) Low-power design methodology .....                               | 227 |
|     | Annex F (normative) Value conversion tables .....                                      | 252 |
|     | Annex G (normative) Supporting hard IP .....                                           | 255 |
|     | Annex H (normative) UPF power-management commands semantics and Liberty mappings ..... | 258 |
|     | Annex I (informative) Power-management cell modeling examples .....                    | 273 |
|     | Annex J (informative) Switching Activity Interchange Format .....                      | 303 |



# IEEE Standard for Design and Verification of Low-Power Integrated Circuits

*IMPORTANT NOTICE: This standard is not intended to ensure safety, security, health, or environmental protection in all circumstances. Implementers of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.*

*This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Documents." They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>*

## 1. Overview

### 1.1 Scope

This standard establishes a format used to define the low-power design intent for electronic systems and electronic intellectual property (IP). The format provides the ability to specify the supply network, switches, isolation, retention, and other aspects relevant to power management of an electronic system. The standard defines the relationship between the low-power design specification and the logic design specification captured via other formats [e.g., standard hardware description languages (HTML)].

## 1.2 Purpose

The standard provides portability of low-power design specifications that can be used with a variety of commercial products throughout an electronic system design, analysis, verification, and implementation flow.

## 1.3 Key characteristics of the Unified Power Format (UPF)

The Unified Power Format (UPF) provides the ability for electronic systems to be designed with power as a key consideration early in the process. UPF accomplishes this by allowing the specification of what was traditionally physical implementation-based power information early in the design process—at the register transfer level (RTL) or earlier. Figure 1 shows UPF supporting the entire design flow. UPF provides a consistent format to specify power design information that may not be easily specifiable in an HDL or when it is undesirable to directly specify the power semantics in an HDL, as doing so would tie the logic specification directly to a constrained power implementation. UPF specifies a set of HDL attributes and HDL packages to facilitate the expression of power intent in HDL when appropriate (see Table 4 and Annex BX). UPF also defines consistent semantics across verification and implementation, i.e., what is implemented is the same as what has been verified.

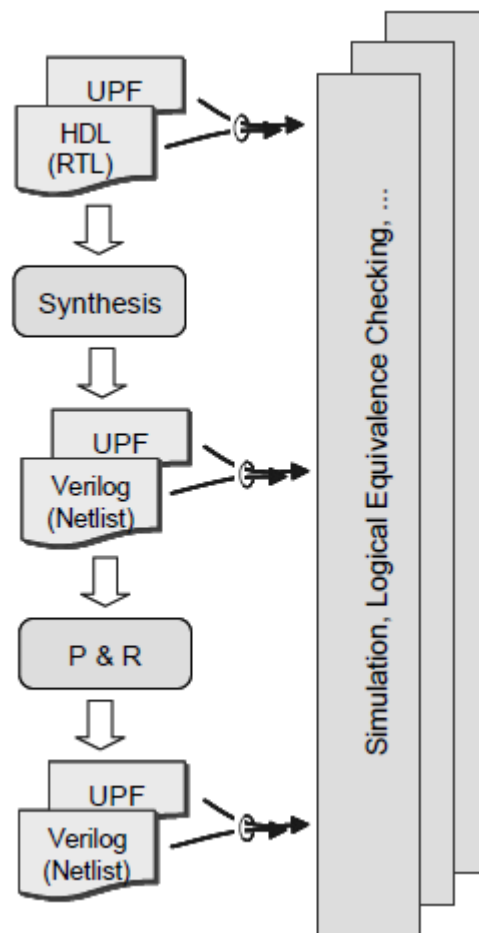


**Figure 1—UPF tool flow**

As indicated in Figure 1. UPF files are part of the design source [and, when combined with the HDL, represent a complete design description: the HDL describing the logical intent and the UPF describing the power intent](#). Combined with the HDL, the UPF files are used to describe the intent of the designer. This collection of source files is the input to several tools, e.g., simulation tools, synthesis tools, and formal verification tools. [UPF supports the successive refinement methodology \(see 4.81 where power intent information will grow along the design flow to provide needed information for each design stage.](#)

- [Simulation tools can read the HDL/UPF design input files and perform RTL power-aware simulation. At this stage, the UPF may only contain abstract models such as power domains and supply sets without the need to create the power and ground network and implementation details.](#)
- Synthesis tools can read the HDL/UPF design input files and produce a netlist. The tool or user may produce a new UPF fileset that, combined with the netlist, represents a further refined version of same design.
- In those cases where [design object](#) names change, a UPF file with the new names is needed. A UPF-aware logical equivalence checker can read the full design [and UPF](#) filesets and perform the checks ~~including the results of the UPF commands~~ to ensure [power-aware](#) equivalence.
- Place and route tools read both the netlist and the UPF files and produce ~~outputs a~~ [physical netlist](#), potentially including an output UPF file.

UPF is a concise power ~~design~~ intent specification capability. Power ~~design~~ intent can be easily specified over many elements in the design. A UPF specification can be included with the other deliverables of IP blocks and reused along with the other delivered IP. UPF supports various methodologies through carefully defined semantics, flexibility in specification, and, when needed, defined rational limitations that facilitate automation in verification and implementation (see Annex E).

A *UPF specification* defines how to create a supply network to supply power to each ~~design element~~ [instance](#), how the individual supply nets behave with respect to one another, and how the logic functionality is extended to support dynamic power switching to these logic ~~design elements~~ [instances](#). By controlling the ~~operating states and~~ voltages of ~~each supply net and whether the supply nets (and their connected design elements) are turned on or off,~~ [the supplies provided to](#) the supply network ~~only provides power at~~ [and by controlling the level the states of power switches that are part of the supply network, the power management logic of a system can cause each](#) functional ~~areas of~~ [region to receive](#) the ~~chip need~~ [power required](#) to complete ~~the its~~ computational tasks in a timely manner.

## 1.4 Use of color in this standard

This standard uses a minimal amount of color to enhance readability. The coloring is not essential and does not affect the accuracy of this standard when viewed in pure black and white. The places where color is used are the following:

- Cross references that are hyperlinked to other portions of this standard are shown in underlined-blue text (hyperlinking works when this standard is viewed interactively as a PDF file).
- Syntactic keywords and tokens in the formal language definitions are shown in **boldface-red text**.
- Command arguments that can be [provided incrementally](#) (*layered*) are shown in **boldface-green text**. See also 5.11.
- [Syntactic keywords and tokens that have been explicitly identified as legacy or deprecated constructs \(see 6.1\) may be shown in brown text.](#)

## ~~1.4 Power supply network design intent~~

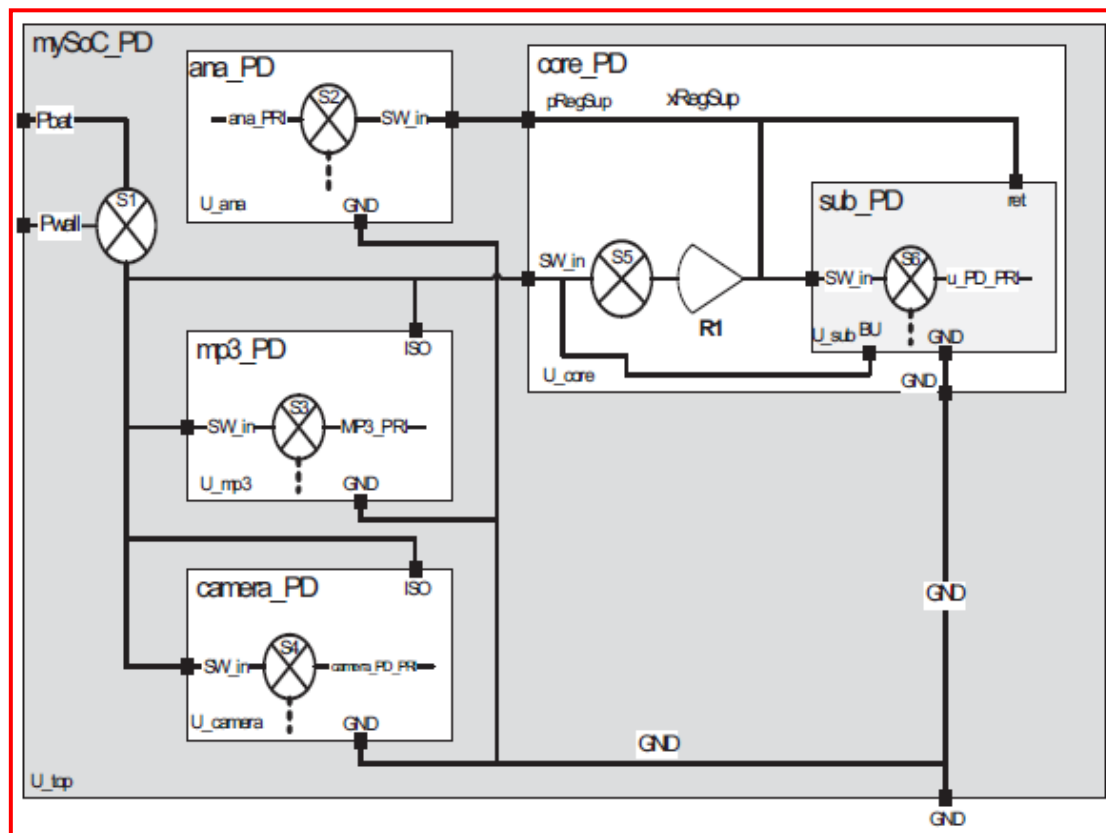
~~Designing electronics to meet low power design constraints requires the specification of a power supply network that can control the distribution of that supply to minimize energy consumption. UPF supports the specification of the power supply distribution network so the supply network can be automatically implemented at a relatively abstract level.~~

~~To help manage the complexity of the supply network specification, power domains are defined to group elements from the logic hierarchy that share common supply needs. By default, all logic elements in a power domain use the same primary supply. Additional supplies may be defined to serve different uses in a power domain. In addition to the primary supply, UPF provides well-defined semantics for other supplies for both verification and implementation contexts.~~

The *supply network* consists of supply ports, switches, and supply nets. Supply network objects are defined within the logical hierarchy relative to the context of a power domain. *Supply ports* provide the supply interface to the logical hierarchy and the power domain's elements. Supply ports also provide the supply interface to switches. *Switches* control the supply distribution. *Supply nets* connect supply ports.

Although there is an obvious inference to the actual wires and ports in the implemented hardware, a UPF supply network is an abstraction of the electrical network on the chip. UPF defines no routing or layout information. As the supply network is specified apart from the logic design, the logic design specification remains independent of a specific power supply network specification.

Figure 2 shows an example supply distribution network for a hypothetical chip, mySoC. A top-level power domain, mySoC\_PD defined at the U\_top instance level in the logical hierarchy, is defined with three supply ports, Pbat, Pwall, and GND. These ports represent the off-chip power sources. A single switch (SI) controls whether the chip receives its supply from Pbat or Pwall. Supply nets connect the output from SI to each power domain defined within the top-level power domain. mySoC\_PD does not contain any logic elements other than the root of the design U\_top; its purpose is to define the interface to the off-chip power sources and provide the top-level supply network.



(Deleted)

**Figure 2 A supply distribution network**

Four power domains are defined at the next level, ana\_PD, camera\_PD, core\_PD, and mp3\_pd. camera\_PD, mp3\_PD, and ana\_PD each have a similar supply network internal to them. The primary supply is switched (S4, S3, and S2), which indicates power to the logic elements in the domains can be turned off. Unswitched isolation supplies are provided for camera\_PD and mp3\_PD.

core\_PD contains a nested domain sub\_PD. core\_PD's primary supply xRegSup is switched and regulated within the domain, as it is the output of a regulator that has a switch (S5) driving the regulator's input (RI). Besides being the primary supply for core\_PD, xRegSup also provides a retention supply for sub\_PD. The regulated supply is also connected to the output supply port (pRegSup) and, ultimately, the switch input (S2) whose output drives the primary supply for ana\_PD. This indicates ana\_PD is powered on only when core\_PD is powered on. Within sub\_PD, the primary supply is switched (S6), indicating sub\_PD can be powered down separately from core\_PD, but holding the relationship that if core\_PD is powered down, so is sub\_PD. sub\_PD also contains two additional supplies (ret and BU). The ret supply (which is used as a retention supply) has the same source as core\_PD's primary supply (the output from switch S5) and the BU supply has the same source as the input to switch S5; therefore, the retention supply (ret) is valid only when core\_PD is on (switch S5 in the core\_PD power domain). The BU supply is provided for any retention elements whose state needs to be saved when both sub\_PD and core\_PD are powered down.

This example demonstrates a methodology of locating the switches for a power domain's supplies within the power domain. UPF also supports placing the switches that can be switched off outside the power domain. If this methodology were being used, switches S2, S3, and S4 would be located in mySoC\_PD and additional supply nets to connect the outputs of the switches to the corresponding ports of the nested power domains' logical hierarchies would be created. Similarly, the switch S5 and regulator RI would be placed in mySoC\_PD and the switch S6 moved up to core\_PD. Either methodology is valid and the supply networks are equivalent.

UPF provides the flexibility to specify the network in whatever way makes sense for a given design or methodology, including what can be switched. In this example, only the power supplies are switched; however, switches can also be placed on grounds, bias nets, and other named supply functions. This provides complete flexibility in how the supply distribution is controlled.

~~A UPF supply network defines a directed acyclic graph (DAG) when inout supply ports are not used. Limiting the supply network to a DAG representation simplifies the evaluation of the supply network state. This simplification enables efficient verification of the logic design and supply network within a digital simulation environment.~~

~~The state of the supply network is simply the state of each control element (switch) in the network and the values of any root supply driver at any given point in time. Absolute relationships are easy to infer from the supply network specification (e.g., core\_PD is always on when sub\_PD and ana\_PD are on). To capture situations that are never intended to occur, UPF can also define legal and illegal power states (see Clause 4). The named power states and the legality of a named or unnamed power state can be used to optimize the design implementation and to facilitate design verification.~~

## 1.5 Contents of this standard

The organization of the remainder of this standard is as follows:

- Clause 2 provides references to other applicable standards that are presumed or required for this standard.
- Clause 3 defines terms and acronyms used throughout the different specifications contained in this standard.
- Clause 4 describes ~~power domains and specifies the conventions for naming objects, their name space resolution, and conflict resolution~~ [basic concepts underlying UPF](#).
- Clause 5 ~~defines simulation semantics~~ [describes the language basics](#) for ~~various~~ UPF [and its](#) commands.
- Clause 6 details the syntax and semantics for each UPF [power intent](#) command.

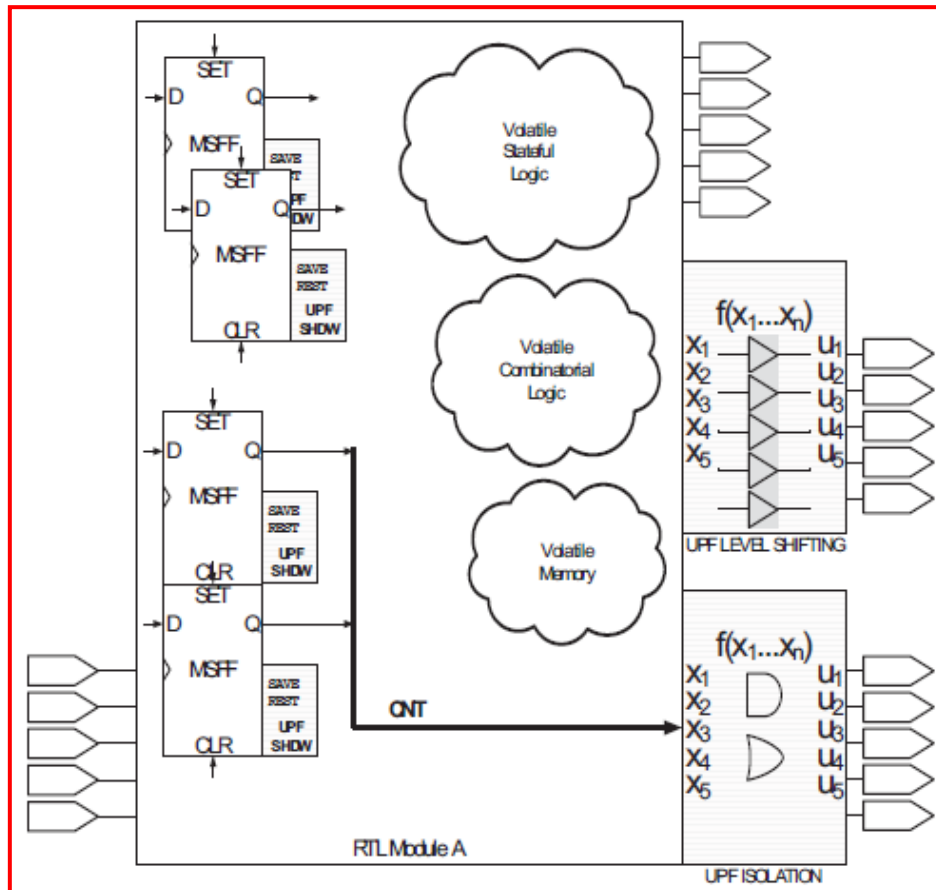
- Clause 7 details the syntax and semantics for each UPF ~~query and the find\_objects~~ [power-management cell](#) command.
- Clause 8 ~~describes the syntax and semantics of the Switching Activity Interchange Format (SAIF)~~ [defines a reference model for UPF command processing.](#)
- [Clause 9 defines simulation semantics for various UPF commands.](#)
- Annexes. Following Clause 9 are a series of annexes.

~~NOTE The normative portion of this document is organized in two major sections. The first, Clause 1 through Clause 7, defines the UPF capabilities for capturing low power design intent. The second, Clause 8, defines a format for capturing switching activity information, which can be used for analysis and estimating power consumption. While designing electronics to minimize power would benefit from tools supporting both sets of capabilities, the two sections hold no dependencies between one another.<sup>1</sup>~~

## 1.5 Extending logic specification

~~UPF defines extensions of the logic design with power specific capabilities and constraints without modifying the original logic specification. UPF provides designers with the guarantee they intuitively expect, i.e., by adding a UPF power specification to a logic design or by changing or replacing an existing UPF specification for a logic design, designers do not need to touch the original logic specification or re-verify the logic functionality independent of the power specification. The UPF standard also facilitates the reuse of the logic specification in contexts where an explicit power design intent is not required or where the power design intent can change from one implementation of the design to another. Figure 3 illustrates how UPF extends the logic design by adding power functionality while leaving the original functionality unmodified.~~

~~Figure 3 also demonstrates the addition of retention and isolation functionality to the logic design. Isolation is required to ensure undefined outputs from powered-down design elements do not drain power from those design elements that are not powered down. Isolation also ensures a specific logic value is driven from the power domain's outputs. Assuming the output CNT in Figure 3 can serve as an interrupt or reset indicator to another logic element, if the isolation of CNT did not take into account the active level of the interrupt or reset interpretation of its value, then powering down the domain containing this counter could result in undesired side effects, e.g., logic in another power domain entering and remaining in a reset or interrupt handling mode. UPF supports the specification of isolation strategies that provide information on the clamp values and location of the isolation logic.~~



(Deleted)

**Figure 3—UPF extends the logic design**

*Retention* is the ability to save the value of a sequential element in a power domain prior to switching off the power to that element and then later restoring its value after power has been enabled for the element. Behavioral and implementation retention semantics can be specified for sequential elements that require save and restore capabilities.

~~Isolation and retention both demonstrate UPF's concise specification characteristics and its flexibility in specification. A general strategy can be defined for both isolation and retention. This strategy is then applied to all ports or sequential elements governed by the scope of the strategy. Specific overriding of a general strategy allows for the management of exceptional situations. Flexibility is realized through the recognition that isolation and retention behavior, for any given design or implementation, may be more complex or require a different connectivity than the predefined general behaviors supported by UPF. In these situations, mapping of the isolation or retention to specific verification (functional behavior) and implementation models is supported.~~

## 2. Normative references

The following referenced documents are indispensable for the application of this ~~document~~ [standard](#) (i.e., they must be understood and used, so each referenced document is cited in [the](#) text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEC 61691-1-1/IEEE [Std 1076™](#), Behavioural languages—Part 1: VHDL Language Reference Manual.<sup>1 2</sup>

<sup>1</sup> IEC publications are available from the ~~Sales Department of the~~ International Electrotechnical Commission, ~~Case Postale 131, 3, rue de Varembe, CH 1211, Geneve 20, Switzerland/Suisse~~ (<http://www.iec.ch/>). IEC publications are also available in the United States from the ~~Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA~~ (<http://www.ansi.org/>).

<sup>2</sup> IEEE publications are available from The Institute of Electrical and Electronics Engineers ~~Inc., 445 Hoes Lane, Piscataway, NJ 08854, USA~~ (<http://standards.ieee.org/>)

IEEE Std 1800™, IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language.<sup>3</sup>

### 3. Definitions, acronyms, and abbreviations

For the purposes of this document, the following terms and definitions apply. The ~~Authoritative Dictionary of IEEE Standards Terms~~ [Dictionary Online](#) [B1]<sup>5</sup> should be ~~referenced~~ [consulted](#) for terms not defined in this clause.<sup>4, 5</sup> [Certain terms in this standard reflect their corresponding definitions in IEEE Std 1800 or IEC 61691-1-1/IEEE Std 1076, or they are listed in Annex A.](#)<sup>6</sup>

#### 3.1 Definitions

**active component:** A component that contains one or more input receivers and one or more output drivers whose values are functions of the inputs, but whose inputs and outputs are not directly connected; or any HDL construct(s) that synthesize(s) to an active component.

<sup>3</sup> The IEEE standards or products referred to in this clause are trademarks of The Institute of Electrical and Electronics Engineers, Inc.

<sup>4</sup> [IEEE Standards Dictionary Online](#) subscription is available at:

[http://www.ieee.org/portal/innovate/products/standard/standards\\_dictionary.html](http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html).

<sup>5</sup> The numbers in brackets correspond to those of the bibliography in Annex A.

<sup>6</sup> [Information on references can be found in Clause 2.](#)

**active control signal:** A control signal that is currently presenting the value (level) or transition (edge) that enables or triggers an active component to operate in a particular manner.

**active power state:** A power state whose logic expression and, if present, supply expression evaluate to *True* at a given time.

**activity:** Any change in the value of a net, regardless of whether that change is propagated to an output.

**ancestor:** Any instance between the current **scope** in the **logic hierarchy** and its root scope. When the current **scope** is a top-level module, it does not have any ancestors. *See also:* **descendant**.

**anonymous object:** An object that is not named in the context of UPF. Implementations may assign a legal name, but such names are not visible in the UPF context.

~~3.1.2 **active scope:** The instance specified by the `set_scope` command (see 6.50).~~

~~3.1.3 **activity:** Any change of the inputs of an element, regardless of whether there are any changes on the output.~~

~~3.1.4 **ancestor:** Any instance between the active scope in the logic hierarchy and its root, including the root. However, when the active scope is the root, it does not have any ancestors.~~

~~3.1.5 **automatic supply net connection:** A supply net connection created as a result of the association of a supply net of a supply set identified by the function the net performs for the set and the `pg_type` attribute on a supply port. *See also:* **explicit supply net connection** and **implicit supply set connection**.~~

~~3.1.6 **command:** A Tel procedure [B5] defined to specify UPF power design intent with one or more design objects as its target.~~

~~3.1.7 composite domain: A container comprising a set of power domains called subdomains. All subdomains in the composite domain share the same primary supply set. Any operation performed on the composite domain has the same effect as performing the operation on each of the subdomains.~~

~~3.1.8 connection: The attachment of a port to a net. See also HighConn and LowConn.~~

balloon latch: A retention element style in which a register's value is saved to a dedicated latch at powerdown and the latch value is restored to the register at power-up.

boundary instance: An instance that has no parent or whose parent is in a different power domain.

component: A physical and logical construction that relates inputs to outputs.

composite domain: A power domain consisting of subordinate power domains called subdomains. All subdomains in a composite domain share the same primary supply set. Any operation performed on a composite domain has the same effect as performing the operation on each of its subdomains.

configuration UPF: The UPF specification for an intellectual property (IP) block that defines a particular configuration of the block for use in a given system. The configuration UPF typically includes the constraint UPF and extends it with configuration-specific details. Sometimes referred to as golden UPF.

connected: Attached together via a direct connection.

constraint UPF: The UPF specification for an intellectual property (IP) block that defines constraints that must be met by any configuration of the IP block used in a larger system. Sometimes referred to as platinum UPF

**corruption semantics:** The rules defining the behavior of ~~in~~ logic response to reduction or disconnection of power to that logic. ~~context of the supply network specification, which might remove support of normal logical operation from time to time, as distinguished from the default always-on logical behavior~~

**current scope:** The design hierarchy location that serves as the immediate context interpretation and execution of UPF commands. Also, the **instance** specified by the **set scope** command.

NOTE—See 6.52.<sup>7</sup>

**declared:** Specified in the HDL explicitly or implicitly via a UPF command.

**descendant:** Any instance between the current scope in the logic hierarchy and its leaf-level instances. When the **current scope** is a leaf-level instance, it does not have any descendants. *See also:* **ancestor.**

**descendant subtree:** A portion of a **logic hierarchy**, rooted at one **instance** in the hierarchy, and containing that **instance** and all of its **descendants.**

**design hierarchy:** A hierarchical structure of nested definitions described in an HDL.

direct

**direct connection:** A physical wire; or any HDL construct(s) that synthesize(s) to a direct connection.

**domain port:** A port that is on the interface of a **power domain.**

<sup>7</sup>Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

**driver supply:** For a **driver** that is a transistor, the supply connected to its source or drain; for a **driver** that is an inverter, the pair of supplies connected to the source/drain of the transistor pair comprising the inverter; or for an output of an **active component**, the related **supply set** of that output.

**electrically equivalent:** For **supply ports/nets**, connected (whether the connections are evident or not in the design) without any intervening switches, and therefore guaranteed to have the same value at all times from the perspective of any load; for **supply sets/set handles**, consisting of a set of electrically equivalent **supply nets** for each required function.

**equivalent:** A pair of **supply nets** or a pair of **supply sets** that are considered to be interchangeable for certain purposes. *See also:* **electrically equivalent; functionally equivalent.**

**erroneous:** A usage that is likely to lead to an error in the design, but that tools may not be able to detect and report.

~~3.1.10 **design element:** An **instance** of a SystemVerilog module (see IEEE Std 1800), VHDL entity (see IEC/IEEE 61691-1-1), or library cell. The term design element is often abbreviated to **element**.~~

~~3.1.11 **design object:** A design object is any object that can be declared or created in the **logic hierarchy**. Design objects include (but are not limited to) wires, registers, switches, **ports, supply nets, and design elements**. The term design object is often abbreviated to **object**~~

~~3.1.12 **driver:** Any **design element** that supplies a value to a net, either directly, or indirectly via a **port**. Ports are not drivers.~~

~~3.1.13 **explicit supply net connection:** A supply net connection that is explicitly specified in a **connect\_supply\_net** command (see 6.13). See also **automatic supply net connection** and **implicit supply net connection**.~~

**extent (of a domain):** The set of ~~design elements~~instances that comprise a **power domain**.

**feedthrough:** A direct connection between two **ports** on the **interface of a power domain**, where the connection involves two **ports** on the upper boundary, or two **ports** on the lower boundary, or one of each; also, a direction connection between two **ports** of the same **leaf-level instance**.

**feedthrough port:** A **port** on the **interface of a power domain** that is part of a **feedthrough** through that domain, or a **port** on the interface of a **leaf-level instance** that is part of a **feedthrough** through that **instance**.

**functionally equivalent:** Functioning identically from the perspective of any load, either as a result of being **electrically equivalent**, or due to independent but parallel circuitry.

**generate block:** In the HDL code, this represents a level of design hierarchy, although a generate block is not itself an **instance**. After synthesis, generate blocks do not exist as an independent level of hierarchy. It is illegal to create any UPF objects in a scope that corresponds to a generate block.

**golden source:** The design together with the constraint UPF and the configuration UPF.

**golden UPF:** *See:* **configuration UPF**.

**hard macro:** A block that has been completely implemented and can be used as it is in other blocks. This may be modeled by an HDL module for verification or as a library cell for implementation.

**hierarchical name:** A series of names separated by the **hierarchical separator character**, the final name of which may be any legal HDL name or UPF name, and each preceding name is the name of an **instance** or **generate block** in which the following name is declared. *See also:* **hierarchical separator character**.

~~3.1.15 HighConn: This indicates the hierarchically higher (closer to the root) port connection.~~

**hierarchical separator character:** A special character used in composing hierarchical ~~port, pin, net, or instance names from simple identifiers~~ names. The hierarchical separator character is a slash (/).

**HighConn:** The side of a **port** connection that is higher in the **design hierarchy**; the actual signal associated with a formal **port** definition.

**implementation UPF:** The UPF specification of how power distribution and control is to be implemented for a system. The implementation UPF typically includes the configuration UPF for each of the intellectual property (IP) blocks instantiated in the system. Sometimes referred to as **silicon UPF**.

**inactive:** A normally active component in a state in which it does not respond to **activity** on its inputs. Also, a control signal that is not currently presenting the value (level) or transition (edge) that enables or triggers an **active component** to operate in a particular manner.

~~3.1.17 implicit supply set connection: The supply nets of a supply set are implicitly connected to those elements in the extent of a power domain, or those elements targeted by an isolation, retention, or level shifter strategy, when no supply port exists on the element. See also automatic supply net connection and explicit supply net connection.~~

**instance:** A particular occurrence of a ~~design object~~ SystemVerilog module (see IEEE Std 1800), VHDL entity (see IEC 61691-1-1/IEEE Std 1076), or library cell at a specific location within the design hierarchy.

**interface of a power domain:** The union of the **upper boundary** and the **lower boundary** of the **power domain**.

**isolation:** A technique used to provide defined behavior of logic signals ~~associated with independently switched supplies~~ when its driving logic is not active.

**isolation cell:** ~~A design element~~ 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.

**leaf-level cell:** An instance that has no descendants, or an instance that has the attribute UPF is leaf cell associated with it.

NOTE—See Table 4.

leaf-level instance: See: leaf-level cell.

**level-shifter:** ~~A design element~~ An instance that translates signal values from an input voltage swing to a different output voltage swing.

**live slave:** A retention element style in which the slave latch of a master-slave flip-flop (MSFF) is always on and therefore maintains the value of the MSFF during power-down.

**logic hierarchy:** An abstract view of a **design hierarchy** in which only those definitions representing **instances** are included.

~~3.1.21 **logic design:** Design functionality defined using an HDL, such as SystemVerilog or VHDL.~~

~~3.1.22 **logic hierarchy:** The tree structure of **design elements** specified within a **logic design**.~~

**LowConn:** ~~This indicates the lower (further from the root) port connection~~ The side of a **port** connection that is lower in the **design hierarchy**; the formal **port** definition.

**lower boundary (of a power domain):** The HighConn side of each **port** of each **boundary instance** in the **extent** of another **power domain** whose **parent** is in the **extent** of this domain, together with the HighConn side of each **port** of any macro cell instance in this **power domain**, for which the related supply set is neither identical to nor equivalent to the **primary supply set** of this domain.

**map:** ~~Substitute~~ Identifies a specific **model** ~~for~~ corresponding to an abstract behavior. An **instance** of the **model** can then be used to implement the specific behavior.

~~3.1.26 **merged domain:** A single **power domain** that is the union of the elements that were members of the power domains being merged. The domains that are merged into the merged domain are no longer visible.~~

**model:** A SystemVerilog module, VHDL entity/architecture, or Liberty cell.

**named power state:** A power state defined using `add_power_state` [add port state, or add pst state](#) for a supply set or power domain, or the `DEFAULT_NORMAL` and `DEFAULT_CORRUPT` power states predefined for supply sets. ~~See also unnamed power state.~~

**net:** The [individual net segments that make up a](#) collection of interconnections between a collection of **ports**. A **net** may be named or anonymous ~~At a module boundary, a named net may be considered as having two parts, the **HighConn** and **LowConn** portion of the relative to a particular module or cell boundary~~

**net segment:** [A direct connection within a single instance.](#)

**parent:** The immediate **ancestor** of a given **instance** [within the logic hierarchy](#).

**passive component:** [A direct connection; a component that has neither a receiver nor a driver, whose output is connected to its input, and therefore its output is always the same as its input, e.g., a pass transistor; or any HDL construct\(s\) that synthesize\(s\) to a feedthrough component.](#)

**pg\_type:** ~~Supply ports on components can be labeled with a "Power/Ground Type" name; e.g., the Liberty definition [B6] of pg\_type.~~ [An attribute of a port that indicates its use in providing power to a cell.](#)

**platinum UPF:** [See: constraint UPF.](#)

**port:** A **connection** on the interface of a SystemVerilog module or VHDL entity [Also, a port on the interface of a power domain.](#)

**power domain:** A collection of ~~design elements that~~ [instances 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.** A power domain may also have additional supplies, including **retention and isolation** supplies.

**power rail:** The physical implementation of a power supply net.

**power state:** The state of a **supply net**, **supply port**, **supply set**, or **power domain**.

NOTE—See Clause 4.

~~3.1.34 **power state:** The state of a **supply net**, **supply port**, **supply set**, or **power domain**. The power state of a **supply net** or **supply port** is the state and voltage values of that **supply net** or **supply port**. The power state of a **supply set** is determined by the state of the **supply nets** in the set or by a logic expression if the supply network has not yet been created. The power state of a **power domain** is determined by the state of **supply sets** associated with the domain and **supply sets** or **supply nets** referenced in a named power state's **supply\_expr** defined for the domain. The term power state is often abbreviated to **state**.~~

**power state table (PST):** A table that captures the legal combinations of **power states** for a set of **supply ports** and/or **supply nets**.

**primary supply set:** The **supply net** connections inferred for all **elements instances** in the **power domain**, unless overridden **elsewhere**.

**receiver:** The gate of a transistor; the input to an inverter; any **component** whose behavior is determined by an input signal; any combinational logic including a buffer of any kind; any sequential logic; or any HDL construct(s) that synthesize(s) to such combinational or sequential logic.

**receiver supply:** For a **receiver** that is the gate of a transistor, the supply connected to that transistor's source or drain; for a **receiver** that is the input to an inverter, the pair of supplies connected to the source/ drain of the transistor pair comprising the inverter; or for a **receiver** that is part of an **active component**, the primary supply of the **power domain** to which that **receiver** belongs or, in some cases, the secondary supply of the **component** if it has a secondary supply.

**3.1.37 rail:** ~~See the Liberty definition [B6] of power\_rail.~~

**3.1.38 receiver:** ~~Any design element that uses the value of a net as an input, either directly, or indirectly via a port. Ports are not receivers.~~

**3.1.39 regular expressions:** ~~See the Tel definition [B4] of regular expressions.~~

**regulator:** ~~A design element~~ An instance that takes a set of input **supply nets** and provides the source for a set of output **supply nets**. The output voltage is a function of ~~combining~~ the input voltages and the logical state of any control signals.

**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.

**retention register:** A register that extends the functionality of a **sequential element** with the ability to retain its memory value during the power-down state ~~following the application of a suitable save and restore protocol sequence.~~

**rooted name:** ~~The name of a design object in the active scope or one of its descendants. It may include a hierarchical separator character, but may not start with the hierarchical separator character~~ The hierarchical name, relative to the current scope, of an object in the logic hierarchy or a UPF object defined for a scope in the hierarchy.

**root scope:** The topmost scope in the logic hierarchy, which contains an implicit instance of each top-level module.

**root supply driver:** The origin of a supply, e.g., an on-system voltage regulator, bias generator modeled in HDL, or an off-chip supply source. *See also:* supply source.

root supply source: An input or inout **supply port** that is not connected to an "upstream" **supply net**; an input or inout **supply port** that is not connected to a root supply source defined in an ancestor scope; an output or inout **supply port** that is not connected to a **supply source** defined in a child scope; a **supply set** or supply set handle function that is neither associated with a **supply port or supply net** (via **associate supply set**) nor connected to another root supply source (via **connect supply net**).

NOTE—See 6.7 and 6.11.

scope: An **instance** in the **logic hierarchy**.

silicon UPF: See: **implementation UPF**.

~~3.1.45 scope~~: A particular ~~design element in the logic hierarchy~~.

~~3.1.46 sequential element~~: An ~~object~~ containing latch or flip-flop functionality.

**simple name**: ~~A name in the active scope. The name does not contain any hierarchical separator~~  
~~characters~~. An identifier that denotes an object declared in a given scope and is not a hierarchical name

**simstate**: The level of operational capability supported by ~~the active~~ a given power state of a supply set.

sink: ~~A receiver~~; the HighConn of an input port or inout port of an **instance**; or the LowConn of an output port or inout port of an **instance**.

source: ~~A driver~~; the LowConn of an input port or inout port of an **instance**; or the HighConn of an output port or inout port of an **instance**.

~~3.1.49 source~~: ~~The active device or port connection that propagates a value on a net.~~

**state element:** A ~~storage device~~ [sequential element](#) such as a flip-flop, latch, or memory element. [Also, a conditionally stored value that would infer in register transfer level \(RTL\) code from which a sequential element from HDL code would be inferred.](#)

**strategy:** [A rule that specifies where and how to apply isolation, level-shifting, state retention, and buffering in the implementation of power intent.](#)

**subdomain:** A member of the set of ~~subdomains~~ comprising a composite power domain ~~each subdomain retains its original identity as well as the collective identity of the composite power domain (see 6.15)~~

**supply function:** ~~The purpose of an individual net in a supply set (e.g., power, ground, etc.)~~ [An abstraction of a supply net in a supply set, the name of which identifies the purpose of the corresponding net in the supply set.](#)

**supply net:** ~~A net with power state semantics~~ [An HDL representation of a power rail.](#)

**supply port:** A ~~port with power state semantics~~ [connection point for supply nets.](#)

**supply set:** A collection of ~~supply nets~~ [functions](#) that [in aggregate](#) provide a [complete](#) power source.

**supply source:** A **supply port** that propagates but does not originate a supply ~~state and voltage~~ value.

**switch:** ~~A design element~~ [An instance](#) that conditionally connects one or more input **supply nets** to a single output **supply net** according to the logical state of one or more control inputs.

**top-level instance:** [An implicit instance corresponding to a top-level module.](#)

**upper boundary (of a power domain):** [The LowConn side of each port of each boundary instance in the extent of this power domain.](#)

~~3.1.58 **unnamed power state:** A power state that is not predefined nor defined by the `add_power_state` command (see 6.8). See also **named power state**.~~

~~3.1.59 **UPF simulation:** A simulation run that depends on semantics described in this standard.~~

## 3.2 Acronyms and abbreviations

DAG            ~~directed acyclic graph~~

CMOS            [complementary metal oxide semiconductor](#)

DFT            [Design for Test](#)

EDA            electronic design automation

HDL            hardware description language

IP              intellectual property

MSFF           [master-slave flip-flop](#)

NMOS           [N-channel metal oxide semiconductor](#)

PG              [power/ground](#)

PMOS           [P-channel metal oxide semiconductor](#)

PST            power state table

|                       |                                                    |
|-----------------------|----------------------------------------------------|
| <a href="#">ROM</a>   | <a href="#">Read-only Memory</a>                   |
| RTL                   | register transfer level                            |
| SAIF                  | Switching Activity Interchange Format              |
| <a href="#">SoC</a>   | <a href="#">System On Chip</a>                     |
| <a href="#">Tel</a>   | <a href="#">Tool Command Language</a>              |
| UPF                   | Unified Power Format                               |
| VCT                   | value conversion table                             |
| VHDL                  | VHSIC hardware description language                |
| <a href="#">VHSIC</a> | <a href="#">Very High Speed Integrated Circuit</a> |

## 4. UPF concepts

This clause provides an overview of concepts involved in defining power intent using UPF. These concepts include those related to the representation of the design structure and functionality in one or more hardware description languages (HDLs), as well as those related to power-management structures and functionality defined for and/or added to the design to model intended power-management capabilities.

The structure and functionality of a design is specified using HDLs such as Verilog [B2], SystemVerilog, or VHDL. Each HDL may have specific terminology and concepts that are unique to that language, but all HDLs share some common concepts and capabilities. A typical design may be expressed in one or more HDLs.

UPF is defined in terms of a generalized abstraction of an HDL-based design hierarchy. This abstraction enables the UPF definition to apply to a design expressed in any of the three HDLs previously mentioned, or in any combination thereof, while at the same time minimizing the complexity of the UPF definition. This clause presents the abstract model and maps it to specific HDL concepts.

UPF is intended to apply to a design as its representation changes from an abstract functional model to a concrete physical model, during which process the power intent expressed in UPF becomes realized as part of the implementation. Because of this, the abstract logic hierarchy that is the basis of the UPF definition shall be understood in terms of both functional specification and physical implementation.

## 4.1 Design structure

### 4.1.1 Transistors

At the lowest level, UPF focuses on controlling power (or more precisely, voltage and current) delivered to transistors. These are usually assumed to be digital complementary metal oxide semiconductor (CMOS) transistors, but they could be analog devices as well or implemented in other technologies. The gate connection of a transistor is a receiver; the source of the signal provided to a gate (in CMOS, typically the output of a P/N transistor pair) is a driver.

### 4.1.2 Standard cells

Transistors are seldom modeled individually in an HDL description; typically, collections of transistors are represented by standard cells that have been developed as part of a particular technology library, which is usually expressed in the Liberty library format (see [B7]). Such cells typically have a primary supply (power and ground) and may also have a secondary supply for related behavior (e.g., state retention).

### 4.1.3 Hard macros

A library may also contain hard macros, which provide predefined physical implementations for much larger and more complex functions. A hard macro may have multiple supplies.

## 4.2 Design representation

### 4.2.1 Models

Library elements have corresponding behavioral models for use in simulation. These models may or may not include power and ground pins for their supplies. Standard cell models are usually written as Verilog modules and use constructs such as Verilog built-in primitives or user-defined primitives (UDPs) to express the relatively simple behavior of a standard cell. They may also be written as VHDL design entities (entity/architecture pairs) using package VITAL, which provides Verilog-like primitive modeling capabilities. Hard macro models may be written in either language, using more complex behavioral constructs such as Verilog initial blocks and always blocks or VHDL processes and concurrent statements.

### 4.2.2 Netlist

A netlist is a collection of unique instances of standard cells and hard macros, interconnected by nets (Verilog) or signals (VHDL). Such instances are considered to be leaf-level instances, because their models are not constructed from an interconnection of subordinate instances, but instead are built using behavioral or functional HDL statements. A netlist may also include hierarchical instances, i.e., instances of a model that is itself defined as a netlist.

A power/ground (PG) netlist is a netlist containing cell and/or hard macro instances that include power and ground pins and a representation of the power and ground supply routing for those instances. A non-PG netlist is one that does not include any representation of the power supply network.

### 4.2.3 Behavioral models

Behavioral models that are written using the RTL synthesis subset of Verilog or VHDL are synthesizable models, or *soft macros*, which can be read by an RTL synthesis tool and mapped to a functionally equivalent netlist. Synthesis involves identifying or inferring the state elements needed to implement the specified behavior and implementing the combinational logic interconnecting those elements and the model's ports.

For many synthesizable HDL constructs, synthesis creates combinational or sequential logic elements that are ultimately defined in terms of transistors, which in turn define drivers and receivers. In particular, any synthesizable statement that involves conditional computation or conditional updating of an output will most likely create logic. In contrast, unconditional assignment statements and port associations typically result in interconnect, not logic; for such HDL constructs, no drivers or receivers are created. In particular, ports do not create drivers; it is the logic driving a port that creates a driver for the port and for the net associated with the port.

### 4.2.4 HDL scopes

An HDL model defines one or more scopes. A scope is a region of HDL text within which names can be defined. Such names are typically visible (i.e., can be referenced) within the scope in which they are defined and, in certain cases, in other scopes (e.g., nested scopes). A Verilog model usually defines a single scope for the whole model. A VHDL model often defines multiple scopes; one for the whole model, plus other nested scopes for process statements and block statements, generate statements in either HDL are also considered to be nested scopes within the model's top-level scope.