IEC 63501-2416

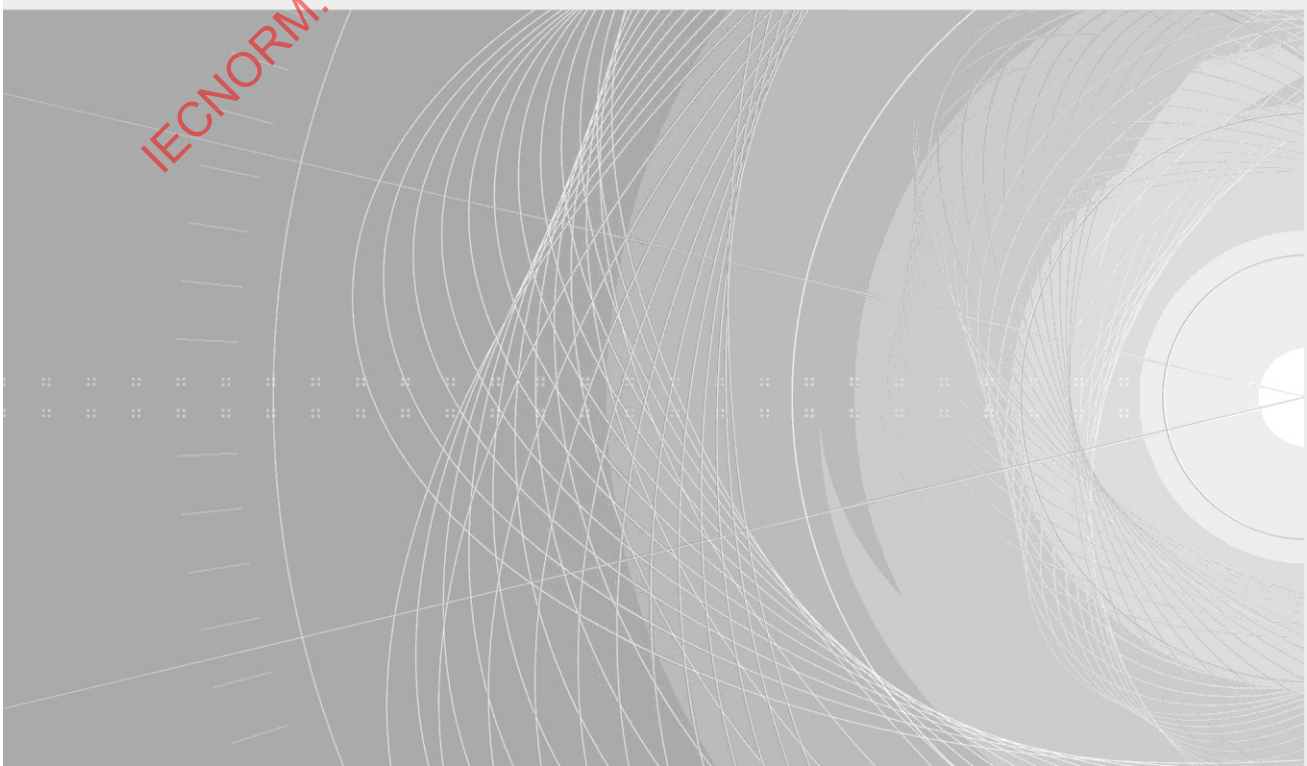
Edition 1.0 2023-10

INTERNATIONAL STANDARD

IEEE Std 2416™

colour inside

Power Modeling to Enable System Level Analysis



IEC 63501-2416:2023-10(en) IEEE Std 2416-2019

IEC Secretariat
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue
New York, NY 10016-5997
United States of America
stds.info@ieee.org
www.ieee.org

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee, …). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

IEC Products & Services Portal - products.iec.ch
Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 300 terminological entries in English and French, with equivalent terms in 19 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC 63501-2416

Edition 1.0 2023-10

INTERNATIONAL STANDARD

IEEE Std 2416™

colour inside

Power Modeling to Enable System Level Analysis

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.01, 35.060

ISBN 978-2-8322-7505-4

Warning! Make sure that you obtained this publication from an authorized distributor.

Contents

POWER MODELING TO ENABLE SYSTEM LEVEL ANALYSIS

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC document(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation.

   IEEE Standards documents are developed within IEEE Societies and 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 IEEE 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 contained in its standards. Use of IEEE Standards documents is wholly voluntary. *IEEE documents are made available for use subject to important notices and legal disclaimers (see* https://standards.ieee.org/ipr/disclaimers.html *for more information).*

   IEC collaborates closely with IEEE in accordance with conditions determined by agreement between the two organizations. This Dual Logo International Standard was jointly developed by the IEC and IEEE under the terms of that agreement.

2) The formal decisions of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees. The formal decisions of IEEE on technical matters, once consensus within IEEE Societies and Standards Coordinating Committees has been reached, is determined by a balanced ballot of materially interested parties who indicate interest in reviewing the proposed standard. Final approval of the IEEE standards document is given by the IEEE Standards Association (IEEE SA) Standards Board.

3) IEC/IEEE Publications have the form of recommendations for international use and are accepted by IEC National Committees/IEEE Societies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC/IEEE Publications is accurate, IEC or IEEE cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications (including IEC/IEEE Publications) transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC/IEEE Publication and the corresponding national or regional publication shall be clearly indicated in the latter.

5) IEC and IEEE do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC and IEEE are not responsible for any services carried out by independent certification bodies.

6) All users should ensure that they have the latest edition of this publication.

7) No liability shall attach to IEC or IEEE or their directors, employees, servants or agents including individual experts and members of technical committees and IEC National Committees, or volunteers of IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board, for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC/IEEE Publication or any other IEC or IEEE Publications.

8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

9) Attention is drawn to the possibility that implementation of this IEC/IEEE Publication may require use of material covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. IEC or IEEE shall not be held responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent 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.

IEC 63501-2416/IEEE Std 2416 was processed through IEC technical committee 91: Electronics assembly technology, under the IEC/IEEE Dual Logo Agreement. It is an International Standard.

The text of this International Standard is based on the following documents:

IEEE Std	FDIS	Report on voting
2416 (2019)	91/1867A/FDIS	91/1888/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

The IEC Technical Committee and IEEE Technical Committee have decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn, or
- revised.

IMPORTANT – The "colour inside" logo on the cover page of this document indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

IEEE Standard for Power Modeling to Enable System-Level Analysis

Developed by the

Design Automation Committee
of the
IEEE Computer Society

Approved 21 May 2019

IEEE-SA Standards Board

Abstract: In this standard, a parameterized and abstracted power model enabling system, software, and hardware intellectual property (IP)–centric power analysis and optimization are described. Concepts and constructs are defined for the development of parameterized, accurate, efficient, and complete power models for systems and hardware IP blocks usable for system power analysis and optimization. Process, voltage, and temperature (PVT) independence; power and thermal management interface; and workload and architecture parameterization are some of the concepts included.

Keywords: IEEE 2416™, intellectual property (IP) blocks, modeling standards, models, optimization, parameterization, power contributors, power data, process, voltage, and temperature (PVT)

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 Notices and Disclaimers Concerning IEEE Standards Documents." They can also be obtained on request from IEEE or viewed at http://standards.ieee.org/ipr/disclaimers.html.

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. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups 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 Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

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. A current 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 IEEE Xplore at http://ieeexplore.ieee.org/ 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.

IEEE Introduction

This introduction is not part of IEEE Std 2416-2019, IEEE Standard for Power Modeling to Enable System-Level Analysis.

All System-on-Chip (SoC) designs today face power issues of one sort or another—maximizing battery life, sizing power grids, controlling leakage power, verifying power sequencing, estimating and modeling power at various abstractions, analyzing electro-thermal effects, and so on.

Despite growing industry and societal concerns with power consumption and significant industry attention, there has not been a standard way of representing power data for use at the system level, especially across a range of process, voltage, and temperature (PVT) points in a single model.

The lack of a complete and robust modeling standard has led to a paucity of power models. Intellectual property (IP) vendors, if they produce any power models at all, produce models that are limited in use and application. As a result, designers and other model users struggle with reliably estimating power during system-level design, ultimately producing SoCs that are less power efficient than they could be. To address this situation, a modeling standard addressing data representation is needed to enable modeling accurate and efficient power and thermal analyses early in the design cycle.

Finally, this standard has incorporated several prior technologies and specifications developed and released by Si2, including *Leakage Power Contributor Modeling*, *Liberty Mode Extensions for Atomic Power Modeling*, *Standards for Efficient System Level Power Analysis*, and *Multi-level Power Modeling*.

Acknowledgments

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

Silicon Integration Initiative, Inc. (Si2™)—*System Level Power Model Interface and Data Representation™, V3.0.*

IEEE Standard for Power Modeling to Enable System-Level Analysis

1. Overview

Even though IEEE Std 1801™-2015[1] added power-state modeling capabilities to the most recently released specification, data representation was not included as that capability was explicitly left out of scope during the IEEE 1801 standardization process. Instead, IEEE Std 1801-2015 assumes another standard will supply the power data to the state-based model defined in IEEE Std 1801-2015. Figure 1 illustrates the power data flow between IEEE Std 1801-2015 and this standard.

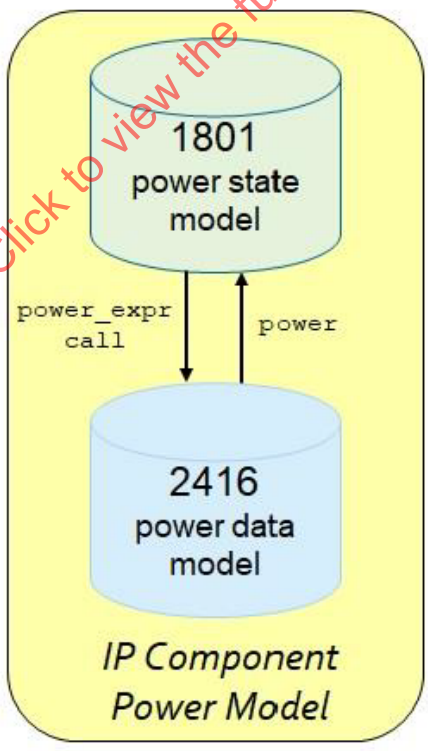


Figure 1—Relationship between this standard and IEEE Std 1801-2015

[1] Information on references can be found in Clause 2.

1.1 Scope

This standard describes a parameterized and abstracted power model enabling system, software, and hardware intellectual property (IP)–centric power analysis and optimization. It defines concepts for the development of parameterized, accurate, efficient, and complete power models for systems and hardware IP blocks usable for system power analysis and optimization. These concepts include, but are not limited to, process, voltage, and temperature (PVT) independence; power and thermal management interface; and workload and architecture parameterization. Such models are suitable for use in software development flows and hardware design flows, as well as for representing both pre–silicon-estimated and post–silicon-measured data. This standard also defines the necessary requirements for the information content of parameterized, accurate, efficient, and complete power models to help guide development and usage of other related power, workload, and functional modeling standards, such as UPF IEEE Std 1801™-2015, SystemC IEEE Std 1666™-2011, and SystemVerilog IEEE Std 1800™-2012. Beyond defining the concepts and related standard requirements, this standard also recommends the use of other relevant design flow standards (e.g., IP-XACT IEEE Std 1685™-2014 [B2][2]), with the objective of enabling more complete and usable power-aware design flows.

1.2 Purpose

This standard supports the ability to develop accurate, efficient, and interoperable power models for complex designs, to be used with a variety of commercial products throughout an electronic system design, analysis, and verification flows.

1.3 Key aspects and considerations

This standard describes a PVT-independent modeling capability that can serve up power data throughout the entire System-on-Chip (SoC) development flow from IEEE 1801 system-level power estimation all the way down to gate-level power calculation. Contributor-based modeling techniques are employed to achieve PVT independence, and multilevel modeling concepts enable a single model to serve various abstraction levels. Contributors are separable, summable components of power consumption that depend on parameters such as transistor widths and stacks, as well as on charging capacitances. Contributors offer the capability to do a "late binding" of specific PVT conditions to an IP model, avoiding the need to characterize the target IP up front under particular PVT conditions.

This standard addresses three separate, but related, modeling concerns: 1) interfaces to other modeling standards, such as IEEE Std 1801-2015, as well as to existing electronic design automation (EDA) tools; 2) persistent data representation and storage; and 3) model evaluation—the conversion of contributor data into energy and power data.

In addition, the following requirements have been identified as being essential to improving both model generation and model usage:

- Direct support for IEEE 1801 IP component power-state model data and interface definitions
- PVT independence: Power data need not be generated at specific PVT points prior to model evaluation
- Ability to coexist with legacy power data formats, such as Liberty [B3]
- Ability to model IP blocks of arbitrary complexity as well as logic primitives, such as `NAND`s, `NOR`s, and Flip-Flops

[2] The numbers in brackets correspond to those of the bibliography in Annex A.

— Ability for a single model to be used in the earliest phases of system design, the latest implementation phases, and all phases in between

The development and standardization of modeling capabilities addressing these issues can provide benefits to both model generators, such as foundries and IP providers, and model consumers, such as IP and SoC architects, SoC verification teams, and energy-aware software developers. For model generators, this development enables efficiency in the model generation and support process (in terms of time, compute resources, and resulting costs). For model consumers, it enables more efficient, reliable, and consistent power analysis and optimization flows from the beginning of the design process to the end. Additionally, the "late binding" of specific PVT conditions provides efficiency benefits for model generators and model consumers alike, including very early stage electrothermal analyses.

1.4 Typographic conventions

The following typographic conventions are used in this standard:

— The `courier` font indicates examples or key term usage from other languages, for example, `<Unit name="capacitanceUnit" value="pF" />`

— The **bold** font indicates key terms or symbols, text that shall be typed exactly as it appears, for example, the **Library** definition is an XML element.

— The *italic* font represents definitions (e.g., *Contributors* may be thought of as a foundational representation) or variables (e.g., `min="`*`min_value`*`"` [where *`min_value`* is a variable]).

— Square brackets ([]) indicate optional parameters.

— Pipes (|) indicate (and separate) a set of pick-one options. For example, in the following line, `EnergyContributor`, `Expression`, `Table`, and `Scalar` are a (optional) set of pick-one choices to consider:

```
EnergyContributor| Expression | Table | Scalar
```

The conventions are for ease of reading only. Any editorial inconsistencies in the use of this typography are unintentional and have no normative meaning in this standard.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in 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.

Accellera Systems Initiative, "Verilog-AMS Language Reference Manual, Version 2.4.0," May 30, 2014.[3]

IEEE Std 1666™-2011, IEEE Standard for Standard SystemC Language Reference Manual.[4, 5]

IEEE Std 1800™-2012. IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language.

[3] Verilog-AMS publications are available from the Accellera Systems Inititave (https://accellera.org/).
[4] IEEE publications are available from The Institute of Electrical and Electronics Engineers (http://standards.ieee.org).
[5] The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

IEEE Std 1801™-2015, IEEE Standard for Design and Verification of Low-Power Integrated Circuits.

World Wide Web Consortium, "eXtensible Markup Language (XML) 1.0" specification (http://www.w3.org/TR/REC-xml/).[6]

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.[7]

contributor: A separable component of power consumption represented by proxies, such as leaking transistors and switched capacitances.

contributor evaluator: An expression, or set of expressions, that evaluates a *contributor* using specified parameter values to return power or energy.

data name: A string that starts with an upper- or lowercase alphabetic character and has 0 or more additional alphanumeric characters or underscores (_).

expression: A combination of one or more constants, variables, operators, and functions used to return a value.

event: Defines the various energy-consuming activities that may occur for a modeled cell.

mode: Used in conventional gate-level power modeling to describe the logical condition of a cell in which all signals, including any clocks, are quiescent.

scaled SI units: A string that represents a unit notation that may include a scaling factor prefix and shall contain a unit suffix.

state: Used in system-level power modeling to describe the logical condition of a cell in which certain signals, particularly clocks, toggle at some particular frequency while all other signals are quiescent.

3.2 Acronyms and abbreviations

IP	intellectual property
PVT	process, voltage, and temperature
SOC	System-on-Chip

4. Model architecture

The general model architecture is a multilevel architecture, which provides backward compatibility with existing gate-level models. This architecture also enables a single model, with multiple levels and multiple representations, to be used at various phases in a design flow. See Figure 2.

[6] XML publications are available from the World Wide Web Consortium (http://www.w3.org).
[7] *IEEE Standards Dictionary Online* is available at: http://dictionary.ieee.org.

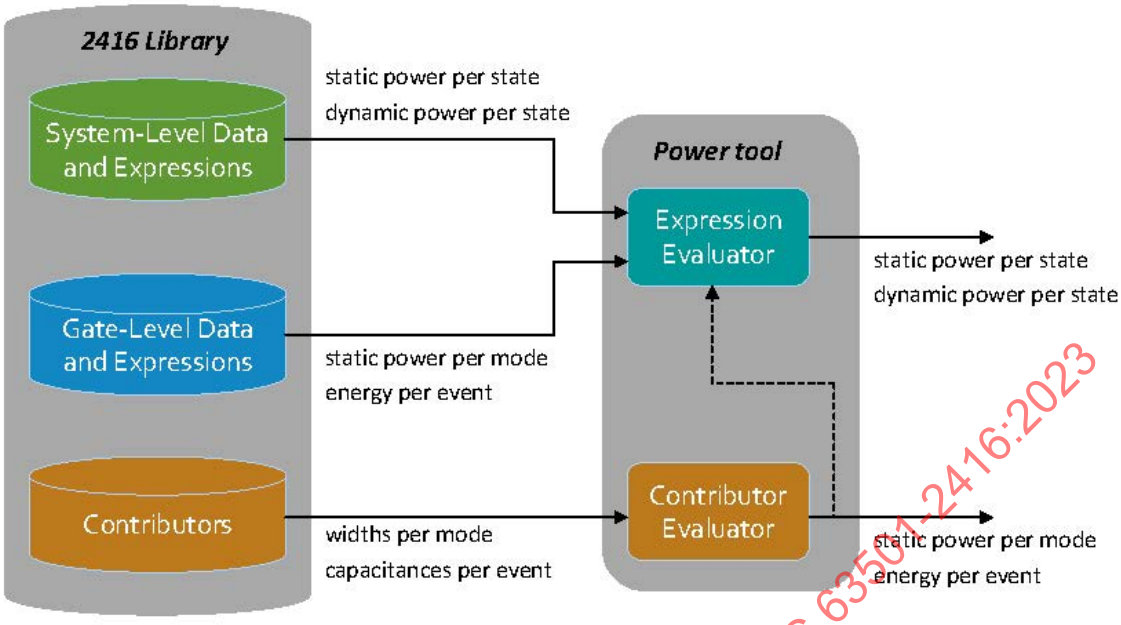


Figure 2—General model architecture

4.1 Two modeling levels

Two different modeling levels are provided, gate level and system level. Data are represented separately for each level, although data at a lower level may be referenced by an upper level.

4.1.1 Gate level

The *gate level* provides the storage of data for backward compatibility with legacy modeling approaches, such as Liberty [B3], in which static power and energy data are stored. A cell modeled in this manner may store hard-coded PVT-specific data or reference contributors. This level uses the **Pins** (see 6.2), **Modes** (see 6.3), and **Events** (see 6.4) elements.

The power and energy data returned from this level are identical to that provided by Liberty [B3], namely, static power per mode per supply and energy per event per supply as shown in Figure 2, along with associated data such as pin names and related information.

4.1.2 System level

The *system level* provides for interoperability with IEEE 1801 component power-state models.

The output from the system level shall be consistent with the IEEE 1801 `-power_expr` attribute of the `add_power_state` command, which expects to receive static power and dynamic power for a defined state. Several inputs (in addition to those required by the gate level) need to be added to provide this power data, including the operating frequency (and potentially others depending on how the IEEE 1801 *power-state model* is defined and parameterized). This level uses the **ModelParameters** (see 5.4), **CellParameters** (see 6.5), and **States** (see 6.6) elements.

For a voltage-independent, system-level power model, data may be represented by expressions (see Clause 9) or tables (see Clause 8). Contributor data are accessed through power expressions and the

Contributor Evaluator (see Figure 2) in a cascaded fashion. The power expressions shall convert PVT-specific energy and static power into dynamic and static power compatible with IEEE 1801 requirements.

4.2 Four modeling representations

Energy and power data may be represented by contributors, expressions, tables, or scalars.

4.2.1 Contributors

Contributors may be thought of as a foundational representation. With this representation, power data are not stored as absolute numbers of energy or power. Instead, data are stored as different power contributors, such as subthreshold leakage contributors, gate leakage contributors, load capacitance contributors, and/or crowbar current contributors. In effect, contributors are proxies for explicit power data.

A contributor's output may be used with external tools as well as with more abstract views. A separate executable, the Contributor Evaluator, shall convert the contributors for switching energy and leakage power into PVT-specific energy and power values as needed. Several inputs are required to define the operating conditions under which the power and energy data are to be returned. These include process corner information, such as t_{ox}, V_t, and μ_0; environmental information, such as temperature and voltage; and instance information, such as input transition time and output load.

Modes, events, and states may each use contributors. Modes represent static power using **PowerContributor**s (see 5.7.1) for subthreshold leakage and gate leakage using contributors, such as equivalent leaking transistor widths. The contributor data stored for events represent energy using **EnergyContributor**s (see 5.7.2) for effective switching capacitances. See Clause 7.

NB: *Static power contributors* (equivalent leaking transistor widths) are PVT independent, while *energy contributors* (effective switching capacitances) are only voltage and temperature (VT) independent.

4.2.2 Multidimensional tables

Multidimensional tables may also be used to represent static power or dynamic power. Tables are especially useful for representing characterized data (either measured or simulated). See Clause 8.

4.2.3 Expressions

Expressions may be used to represent static power or dynamic power. The expressions may be parameterized with parameters global to the library, local to the cell, or both. Expressions are especially useful during early system and architecture explorations. See Clause 9.

4.2.4 Scalars

For modeling simplicity, *scalars* represent power with a single, nonvarying value.

5. Library-level semantics

This clause details the semantics for storing and interpreting power model data. The semantics are based on XML elements and attributes (see World Wide Web Consortium, “eXtensible Markup Language (XML) 1.0” specification) with extensions for representing expressions using Verilog-A (see Accellera Systems Initiative, “Verilog-AMS Language Reference Manual, Version 2.4.0”).

The **Library** is defined as the top level, or root, element and typically contains multiple cells. The **Library** definition is an XML element. Each **Cell** (see 6.1) is defined as a subelement of the **Library**. The **Library** element has several subelements associated with it that apply to all cells. Each **Cell** has several subelements and attributes that are associated only with the cell in which they are defined.

5.1 Library

The **Library** element defines a power model library (as the top-level element). There shall be only one **Library** element per library. The components of the library are contained in its subelements.

Library attributes

name (required data name)—Name of the library.

version (optional data name)—User-defined version string.

condition (required data name)—Name of one **ModelConditions** element (see 5.6) that defines the conditions under which any PVT-specific data are valid. This attribute is valid for all cells in the library unless specified at the **Cell** level (see 6.1).

datagen (optional data name)—Description of the data generation method for all data in this library: one of **measured**, **simulated**, or **estimated**. Applies to all **Cell**s (see 6.1) in the library.

```
<Library name=”library_name” version=”library_version”
    condition=”condition_name”
    datagen=”measured”|”simulated”|”estimated”>
```

5.2 Technology

The optional **Technology** subelement defines the technology for which this library is built. The valid processes are defined as **Process** subelements. The process names and versions can be used to validate the use of the data for a particular design.

Technology attributes

name (required data name)—Technology name.

version (optional data name)—User-defined version string.

Process attributes

name (required data name)—Name of the process corner supported.

version (optional data name)—User-defined version string.

```
<Technology name="technology_name" version="technology_version">
    <Process name="process_corner" version="process_version"/>
    </Technology>
```

5.3 Units

The required library-level **Units** element contains the subelement definitions for the set of SI units for technology, electrical, and environmental parameters, such as the capacitance, voltage, and frequency used in the models. Each **Unit** subelement sets the default name and scaled unit value. The values are composed of an SI unit symbol optionally preceded by a scaling symbol. The name of the unit reflects its use in the models or as a return value of an expression. For example, a power contributor expression could return a value expressed in the **powerUnit** format, such as mW for milliwatts.

A required unnamed **Units** element defines the default units for the library. Additional named **Units** elements can be specified to redefine the set of individual units using the **name** attribute. The set of override values in a named **Units** element shall be accessed by the set name from the **unitsSet** attribute of a **Cell** element. Values not found in the named **Units** element shall come from the default **Units** element. This override capability supports the inclusion of models using different units in a single library.

Units attributes

name (optional data name)—**Units** group identifier. The first **Units** element in the **Library** is required and shall not have a name. Additional **Units** elements are optional and, if present, shall be named. They shall immediately follow the un-named **Units** element.

The **Unit** element is a **subelement** of the **Units** element. Each **Unit** element sets the default name and scaled unit value for a particular parameter. The units are scaled SI units.

Unit attributes

name (required data name)—Unit name for the scaled unit value: one of **capacitanceUnit**, **resistanceUnit**, **dimensionUnit**, **frequencyUnit**, **powerUnit**, **energyUnit**, **voltageUnit**, **temperatureTkUnit**, **temperatureTcUnit**, or **timeUnit**.

value (required scaled SI unit)—Scale and units to scale a numeric value for use by the tools.

The default **Units** values can be overridden en masse using the **unitsSet** attribute on the **Cell** element (see 6.1). The values in the specified named **Units** element override all the values in the default set. In this case, an application or design tool shall use the overridden values.

The overriding set of units shall be specified with the **unitsSet** attribute on **Cell** elements, which will reset the unit values everywhere in the cell. The **Cell** element is the only element that accepts the **unitsSet** attribute.

Certain **Cell**-level subelements, such as **CellParameter**, **Index**, and **Expression**, have a **units** attribute to set the value for individual parameters. The value of this attribute is a scaled SI unit. For example, the **CellParameter units** attribute defines the expected units of the parameter value. The **Expression units** attribute defines the expected units of the returned value. The **Index units** attribute defines the units of the Index element values.

Each subelement inherits the unit parameter values of its parent. However, an inherited value shall be overridden by a subelement's **units** attribute. That is, a subelement's **units** attribute value takes precedence over a parent's value, whether the parent's value was explicitly specified or inherited. For example, any unit specification defined within a **Cell** (see 6.1) takes precedence over that unit specified at the **Library** level (see 5.1).

5.4 ModelParameters

The **ModelParameters** element contains a set of global parameters for use in power models. Each **ModelParameter** subelement defines a parameter by name, assigns an alias to be used when specifying the parameters in expressions and other subelements, and specifies a value and, optionally, a range. Certain parameters are required: **processCorner**, **voltage**, and **temperature** (common to all cells).

The **Constant** subelement can also be used to define constants used in **ModelParameters** expressions. When used, a **Constant** shall be defined before any **Parameter** definitions.

Constant attributes

name (required data name)—Constant name.

value (required numeric value)—Value for the parameter.

Parameter attributes

name (required data name)—Model parameter.

alias (optional data name)—Alias for parameter name; this is useful when specifying the parameters in other elements.

units (optional name)—Units defined for the parameter. The **units** value shall be a scaled SI unit or refer to the **Library Units** (see 5.3). The **units** value can also be the string **"Enumerated"**, indicating a string (which could be further restricted).

value (optional numeric value)—Default value; it is presumed to be a `float`.

min (optional numeric value)—Floor of a valid range of values.

max (optional numeric value)—Ceiling of a valid range of values.

```
<ModelParameters>
    <Constant name="constant_name" value="constant_value"/>
     ...
    <Parameter name="param_name" alias="param_alias"
        units="param_unit" min="units_min_value" max="units_max_value"/>
</ModelParameters>
```

5.5 ModelExpressions

The **ModelExpressions** element defines reusable expressions for any model calculation. These calculations may be used in parameterized model expressions and/or subexpressions. Expressions are defined using the **Expression** subelement. Any expression defined in the **ModelExpressions** element is global to all cells in that library and is referenced by name. The value returned by the expression shall be consistent with the units defined in the **Units** element (see 5.3).

Expression strings are described using a subset of the Verilog-A syntax (see Accellera Systems Initiative, "Verilog-AMS Language Reference Manual, Version 2.4.0"). They are described in depth in Clause 9.

Expression attributes

name (required data name)—Name of the expression, used when referencing the expression.

```
<ModelExpressions>
    <Expression name="expname"> expression_string <Expression/>
    <File name="ExternalFileName" path="PathNameToFile />
</ModelExpressions>
```

Complex expressions may be contained in a separate file. To include one or more files in the **ModelExpressions** element, use the **File** subelement. Each file may contain a single expression or multiple expressions, each defined by its **name**. More than one file element may be defined within the **ModelExpressions** element. The **File** elements shall be defined after the **Expression** elements. The **path** attribute to the **File** element is useful during library development and testing as well as when specifying design-specific expressions. By default, the **path** is controlled by the design tool.

File attributes

name (required data name)—Name of the file containing the expression, without any path description.

path (required path name)—Relative (to the library installation directory) file path to the directory containing the model file.

5.6 ModelConditions

The **ModelConditions** element defines the characterization conditions for any "hard" data (PVT-specific measured or characterized data). Each particular set of characterization conditions shall be **named**.

ModelConditions attributes

name (required data name)—Name of the set of model conditions.

Condition attributes

name (required data name)—Name or alias of one of the **ModelParameters** (see 5.4).

value (required data name)—Value for the parameter.

```
<ModelConditions name="condition_name" >
    <Condition name="P" value="condition_value"/>
    <Condition name="V" value="condition_value"/>
    <Condition name="T" value="condition_value"/>
    <Condition name="F" value="condition_value"/>
</ModelConditions>
```

5.7 ModelContributors

The **ModelContributors** element contains a list of the various contributors referenced by **Cell** models (see 6.1). **ModelContributors** subelements include the **PowerContributor** and the **EnergyContributor**.

Subelements for each contributor define the name, type, and expression used for contributor evaluation along with assigning voltage parameters to contributor pins, as needed.

```
<ModelContributors>
    <PowerContributor name="contributor_name">
        ...
    </PowerContributor>
```

```
    <EnergyContributor name="contributor_name">
        ...
    </EnergyContributor>
</ModelContributors>
```

5.7.1 Power contributor

The static power contributor is a single four-terminal equivalent transistor from which leaking currents are calculated according to leakage current equations appropriate for the target technology. The **Expression** subelement (see 9.2) defines the expression to be used to convert the power contributor into a power consumption value. The equivalent transistor shall be described by the **Device** subelement and its attributes and subelements in terms of its type and terminal voltages.

PowerContributor attributes

name (required data name)—Name of the contributor.

type (required data name)—Name of the type of the **PowerContributor**, either **gate_leakage** or **subthreshold_leakage**, one of which must be specified.

units (required data name)—SI units of the value returned when this **PowerContributor** is evaluated.

Device attributes

type (required data name)—Type of transistor used for the equivalent transistor, usually the name of the associated simulation program with integrated circuit emphasis (better known as "SPICE") transistor model.

The **Values** subelement contains a set of required **Voltage** elements, each describing a voltage (and an optional **Temperature** element) to be applied to the device during the evaluation of the contributor. There shall be at least three **Voltage** elements, one each for **Vgs**, **Vds**, and **Vbs**, corresponding to the Gate to Source voltage, the Drain to Source voltage, and the Body to Source voltage, respectively. These **Voltage** and (optional) **Temperature** elements are used to describe the conditions under which a particular **PowerContributor** is evaluated. The **Temperature** element is specified only for those device models that include an internal self-heating mechanism, such as found in a `FDSOI`.

Voltage attributes

name (required data name)—Name of the voltage to be applied to the power contributor: **Vgs**, **Vds**, or **Vbs**, which corresponds to the voltage on the four-terminal equivalent transistor.

One or more **Temperature** elements may also be defined as appropriate for the contributor and target process technologies. These elements use **Expression** subelements (see 9.2) to define their values.

Temperature attributes

name (required data name)—Name of the temperature to apply to the power contributor.

Only expressions shall be assigned to a **Voltage** or a **Temperature** element, although the expression may be as simple as a single operand. The voltage `Vds`, for example, may be represented by a regression expression and a set of coefficients, while the voltage `Vbs` might be assigned a value of `0.0` for a library in which the n-channel source is wired to the body. See also B.8.

5.7.2 Energy contributor

The energy contributor is a capacitor from which energy is calculated according to $E = CV^2$. Different energy-consuming structures, such as wiring capacitances or equivalent capacitance for switching currents, may be represented with separate and different capacitor contributor types, as defined in the **type** attribute. The output load capacitance shall be omitted from equivalent capacitance values as this modeling paradigm presumes any power tool using this model also accounts for all intercell loading capacitances.

EnergyContributor attributes

name (required data name)—Name of the contributor. Referenced in **EnergyContributor** subelements for **Mode** (see 6.3) or **Event** (see 6.4).

type (required data name)—Name of the type of the **EnergyContributor**, either **wire_cap** or **switched_cap**, one of which must be specified**.**

```
<EnergyContributor name="contributor_name" type="contributor_type">
    <Expression> EnergyExpression_name()</Expression>
</EnergyContributor>
```

6. Cell declaration semantics

The cell declaration **Cell** is the single-cell data model for a given IP block. A cell may have two different modeling levels, gate level and system level. Each level may use different data representations, such as **Contributor** (see Clause 7), **Expression** (see 9.2), **Table** (see Clause 8), or scalar (see 4.2.4).

The *gate level* is defined by **Pins** (see 6.2) and **Events** (see 6.4) and **Mode**s (see 6.3) involving the **Pins**. Conventional gate-level tools require power per mode and energy per event descriptions. At the gate level, static power is described for each mode and dynamic energy is described for each event. Static power and dynamic energy may each be represented by **Contributors**, **Expressions**, **Tables**, or scalars.

The *system level* is defined by **ModelParameters** (see 5.4), **CellParameters** (see 6.5), and **States** (see 6.6). Both static power and dynamic power are defined for each state. IEEE 1801-compatible system simulation tools require static power per state and dynamic power per state. Static power and dynamic power may each be represented by **Contributors**, **Expressions**, **Tables**, or scalars.

6.1 Cell

Each **Cell** element shall have a **name** defined as an attribute to the cell declaration. Each **Cell** element may have any or all of the following subelements: **CellParameters** (see 6.5), **Pins** (see 6.2), **Modes** (see 6.3), **Events** (see 6.4), or **States** (see 6.6). It shall also have a number of subelements and other attributes.

The **Pins** element (see 6.2) along with **Modes** (see 6.3) and **Events** (see 6.4) are required for gate-level modeling, while the **CellParameters** (see 6.5) and **States** (see 6.6) elements are required for system-level modeling.

Cell attributes

name (required data name)—Name of the cell.

condition (optional data name)—Name of one of the **Conditions** elements (see 5.6) that defines the process and temperature conditions under which any PVT-specific data are valid. If the **condition** attribute is not specified, the **Library**-level **Technology** default (see 5.2) shall be used.

unitsSet (optional data name)—Name of one of the named **Units** elements (see 5.3) that contains subelement definitions for the set of SI units for various technological, electrical, and environmental parameters. Use of this attribute overrides the defaults set in the unnamed **Units** element.

datagen (optional data name)—Description of data generation method for all data in this **Cell**: one of **measured**, **simulated**, or **estimated**. Overrides the **Library**-level **Technology** attribute (see 5.2) of the same name.

```
<Cell name="cellname" condition="condition_name"
  unitsSet="units_element_name"
  datagen="measured"|"simulated"|"estimated">
    <Pins ... />
    <Modes ... />
    <Events />
    <CellParameters ... />
    <States ... />
</Cell>
```

6.2 Pins

The **Pins** subelement contains the list of **Pin** elements for the pins on the cell. The **Pin** element describes the pin, and there shall be one **Pin** element for each pin on the cell. A **Pin** has no subelements.

Pin attributes

name (required data name)—Name of the pin on a cell.

width (optional numeric value)—Width of a multibit pin (specified as a string) using the format of *highest_bit***:***lowest_bit*. A single bit of the bussed pin is referenced using *name***[***#***]**.

direction (required data name)—Pin direction: one of **input**, **output**, **inout**, or **internal**. (Internal represents a signal internal to the modeled block that is not brought out as an input, output, or inout pin.)

type (required data name)—Pin usage: one of **clock** (an input that functions as a clock signal), **tri-state** (an output that may be put into a high impedance state), **signal**, **inverted** (an output that is the logical inversion of another output as specified by the **pinReference** attribute), **primary_power** (the primary supply pin for the cell), **primary_ground** (the primary ground pin for the cell), **secondary_power** (a secondary supply pin for the cell, as might be used in a level conversion cell or in an analog mixed-signal cell), or **secondary_ground** (a secondary ground pin for the cell, as might be used in a level conversion cell or in an analog mixed-signal cell).

pinReference (optional data name)—Pin name used with the inverted value of the **type** attribute to indicate which pin is the reference, or source, pin for the inversion.

condition (optional data name)—Name of the **ModelConditions** (see 5.6) under which the pin capacitances are valid.

capacitance (optional numeric value)—Numeric value of the pin capacitance.

```
<Pins>

    <Pin name="pin_name" width="highest_bit:lowest_bit"
        direction="pin_direction" type="pin_type"
        pinReference="pin_name" condition="condition_name"
        capacitance="value"/>
    ...
</Pins>
```

6.3 Modes

The **Modes** element defines the cell's various functional and power modes, which are described as a list of **Mode** subelements.

Modes attributes

mutuallyExclusive (optional data name)—Boolean with the value of **true** or **false**.

- When the value of `mutuallyExclusive` is `true` (the default), all of the modes defined within that **Modes** element are mutually exclusive with each other. When the value of `mutuallyExclusive` is `false`, all of the modes defined within that **Modes** element are non–mutually exclusive with each other.
- Multiple **Modes** elements may be defined. If so, each set of modes is non–mutually exclusive with each other.

defaultMode (optional data name)—Default mode referenced by name. The default mode is used when only a subset of the modes within a **Modes** element is explicitly defined.

initialMode (optional data name)—Initial mode referenced by name. The initial mode is used for initialization in probabilistic power calculations when the initial mode cannot be automatically determined.

The **Mode** subelement describes the power contributors when the specified logical conditions are met.

Mode attributes

when (required data name)—Logical condition of the **Mode**, using a logical expression of the cell pins and/or signal names. An asterisk "*" shall be used to represent all **Modes**. Parentheses shall be used, where needed, to group logical operations.

name (optional data name)—Assigns a name to the **Mode**. This name is used when referencing the **Mode**.

value (optional numeric value)—Scalar representation of the power consumed in that mode; it is presumed to be a `float`.

units (optional data name)—SI units of power data values represented in **Mode** elements.

Modes can be referenced in **Event** elements (see 6.4) using their **name** or the **value** of their **when** statement. If the **Mode** is named, it should be referenced by the **name**. If the Mode is not named, the only way to reference the **Mode** is by its **when** statement.

Each **Mode** element shall contain a **when** condition attribute and, optionally, a **name** attribute. It shall also contain either a **value** attribute (if the data is a scalar; see 4.2.4) one or more **PowerContributor** subelements (see 5.7.1), an **Expression** (see 9.2) representing power data, or a **Table** containing power data (see Clause 8).

```
<Mode name="mode_name" when="logical_expression" value="numeric_value"
    units="power_units">
    [<PowerContributor … /> | <Table … /> | <Expression … />]
```

6.4 Events

The **Events** element defines the various energy-consuming events that may occur for the modeled cell. An *event* may be a simple transition on an input or an output pin, on an internal signal, or on a logical expression of various pins or signals. An event may also be an *arc*—a transition on an input pin leading to a transition on an output pin. An event may also be mode specific (when the cell is in a specific mode).

The **Events** element consists of one or more **Event** subelements. The **Event** subelement describes when the **Event** is evaluated and contains the contributors.

Either the **when** attribute or the **mode** attribute may be specified or neither may be specified, but they shall not be specified together. The **when** attribute uses a logical expression to describe the active mode during which an event occurs, while the **mode** attribute describes the active mode by name. If neither is specified, then the **Event** is not conditional.

Event attributes

name (optional data name)—Name used to reference the **Event**.

when (optional data name)—Logical condition for the **Event** using a logical expression of the cell pins and/or signal names. An asterisk "*" shall be used to represent a logical wildcard. Parentheses shall be used, where needed, to group logical operations.

mode (optional data name)—Name of a mode that describes the logical condition for the **Event**.

style (optional data name)—Defines the modeling style that, in turn, defines the rest of the attributes on the **Event**. One of **pinTransition** (Transition to Transition) or of **modeTransition** (Mode to Mode).

value (optional numeric value)—Scalar representation of the energy consumed by that event; it is presumed to be a `float`.

units (optional data name)—SI units of energy data values represented in **Event** elements.

Each **Event** element shall optionally contain a **name** attribute, a **when** condition attribute, a **mode** attribute, a **style** attribute, and either a **pinTransition** or a **modeTransition** attribute.

- If the **pinTransition** attribute is used (see 6.4.1), the **inputPin** and **inputTransition** attributes shall be present for any transition involving an input pin and the **outputPin** and **outputTransition** attributes shall be present for any transition involving an output pin.
- If the **modeTransition** attribute is used (see 6.4.2), the **nextMode**, **trigger**, and **triggerTransition** attributes shall be present. The **Event** element shall also contain either a **value** attribute (if the data is scalar; see 4.2.4) or one or more **EnergyContributor** subelements (see 5.7.2), an **Expression** (see 9.2) representing energy data, or a **Table** containing energy data (see Clause 8).

Each **EnergyContributor** (see 5.7.2) shall reference contributors defined in the **Library**-level **Contributors** element (see Clause 7). Table-driven energy modeling can be used to model those circuits that are not conveniently represented by the equivalent switched capacitance method.

6.4.1 pinTransition

A **pinTransition** is an event defined by transitions on cell pins or signals. The specified transition may be on an input pin, an output pin, an internal signal, or a logical combination of pins or signals. The event may also be described by an *arc*—a transition on an input leading to a transition on an output pin. A **pinTransition** shall contain the transition attribute **inputTransition** or **outputTransition**, or both attributes. Each transition attribute shall take the value of **rising** or **falling**. Each **Event** element shall also contain one or more **EnergyContributor** elements (see 5.7.2) that assign an energy contributor (defined in the **Library**-level **Contributors** element; see Clause 7) to the **Event** parent.

pinTransition attributes

inputPin (optional data name)—Name of the input pin or signal whose transition initiates the event.

inputTransition (optional data name)—Transition type on the initiating pin: one of **rising** or **falling**.

outputPin (optional data name)—Name of the output pin or signal whose transition completes the event.

outputTransition (optional data name)—Transition type on the concluding pin: one of **rising** or **falling**.

```
<Events>
    <Event name="event_name" when="logical_expression"
        style="pinTransition"
        inputPin="pin_name" inputTransition="trans_type"
        outputPin="pin_name" outputTransition="trans_type"
        value="numeric_value" units="energy_units">
           [<EnergyContributor … /> | <Table … /> | <Expression … />]
</Events>
```

6.4.2 modeTransition

A **modeTransition** is an event defined by a start mode and a next mode, in essence, a transition between modes. The *start mode* is defined by the **mode** attribute (see 6.3), and the *next mode* is described by the **nextMode** attribute.

modeTransition attributes

nextMode (optional data name)—Name of the mode into which the cell transitions.

trigger (optional data name)—Name of the input pin or signal whose transition initiates the mode change.

triggerTransition (optional data name)—Transition type on the trigger pin: one of **rising** or **falling**.

```
<Events>
    <Event name="event_name" style="modeTransition"
        trigger="pin_name"          triggerTransition="transition_type"
        mode="mode_name" nextMode="mode_name"
        value="numeric_value">
           [<EnergyContributor … /> | <Table … /> | <Expression … />]
</Events>
```

6.5 CellParameters

The **CellParameters** element contains the set of parameters (local to the **Cell**) that defines the system-level power model. This set shall match, 1-for-1, the parameters specified by `-power_expr` within an IEEE 1801 power-state model. The scope of each **Parameter** is only that of the enclosing **Cell** (see 6.1). Values for each **Parameter** subelement are to be passed in for use in **Expressions** (see 9.2) and **Tables** (see Clause 8). However, global defaults shall be set in **ModelParameters** (see 5.4) by defining a **Parameter** with the same **name** as a **Parameter** in **CellParameters**.

CellParameter attributes

name (required data name)—Model parameter.

value (optional numeric value)—Value for the parameter; it is presumed to be a `float`. If the value is not specified, the **Parameter** is assigned a value as defined in **ModelParameters** (see 5.4).

units (optional data name)—SI units defined for the parameter.

```
<CellParameters>
    <Parameter name="param_name" value="param_value"
        units="param_units"/>
</CellParameters>
```

6.6 States

Similar to the **Modes** element (see 6.3), the **States** element defines the various functional and power states for system-level modeling. An asterisk "*" shall be used to represent all **States**.

The motivation to distinguish between **Modes** and **States** relates to the environments in which the models are used. The term *mode* is used in conventional gate-level power modeling to describe the logical condition of a cell in which all signals, including any clocks, are quiescent. By contrast, the term *state* is used in system-level power modeling to describe the logical conditional of a cell in which certain signals, especially clocks, toggle at some particular frequency. Thus, mode, at the gate level, indicates static power consumption, while state, at the system level, indicates both dynamic and static power consumption.

States attributes

units (optional data name)—SI units of power data values represented in **States** subelements.

mutuallyExclusive (optional data name)—Boolean with the value of **true** or **false**.

- When the value of `mutuallyExclusive` is `true` (the default), all of the states defined within that **States** element are mutually exclusive with each other. When the value of `mutuallyExclusive` is `false`, all of the states defined within that **States** element are non–mutually exclusive with each other.
- Multiple **States** elements may be defined. If so, each set of states is non–mutually exclusive with each other.

The **States** element contains a list of **State** subelements. Each **State** element contains the dynamic power and static power representations. The power values for each **State** are contained in the subelements **DynamicPower** and **StaticPower** (see 6.7).

State attributes

name (required data name)—Name used to reference the **State**.

when (optional data name)—Logical condition of the cell pins and/or internal signals. This when condition is used for documentation purposes only.

units (optional data name)—SI units of power data values represented in **State** subelements.

6.7 DynamicPower and StaticPower

The **State** element (see 6.6) contains two power-state subelements, **DynamicPower** and **StaticPower**. The **DynamicPower** subelement defines a value for dynamic power consumed by the **State**, and the **StaticPower** subelement fills a similar role for static power. The values for **DynamicPower** and **StaticPower** may be represented by an **Expression** representing power (see 9.2), a **Table** containing power data (see Clause 8), or a scalar (see 4.2.4).

StaticPower and DynamicPower attributes

value (optional numeric value)—Scalar value for power (only used with scalars).

units (optional data name)—SI units of power data values represented in **DynamicPower** and **StaticPower** elements.

6.7.1 Scalars

Use the following type of meta-syntax when using a scalar value:

```
<States mutuallyExclusive="value">
    <State name="state_name" when="logical_expression">
        <DynamicPower value="scalar"/>
        <StaticPower value="scalar"/>
    </State>
</States>
```

6.7.2 Expressions

Expressions representing the cell's static and dynamic power consumption are defined in an **Expression** subelement (see 9.2) within each **DynamicPower** and **StaticPower** element. Each expression may be defined only as a function of external parameters (such as voltage and frequency) or as a function of both external parameters and contributors. The content of the **Expression** subelement is the algebraic equation for calculating the power. The units of the values returned by the expressions need to be consistent with the units set for the data type.

```
<States mutuallyExclusive="value">
    <State name="state_name" when="logical_expression">
        <DynamicPower>
          <Expression ... />
        </DynamicPower">
        <StaticPower>
          <Expression ... />
        </StaticPower">
    </State>
    ...
 </States>
```

6.7.3 Tables

Use the following type of meta-syntax to define a **Table** (see Clause 8 and B.9):

```
<States mutuallyExclusive=”value”>
    <State name=”state_name” when=”logical_expression”>
        <DynamicPower>
          <Table ... />
        </DynamicPower>
        <StaticPower>
          <Table ... />
          </StaticPower>
        </State>
</States>
```

7. Contributors

Contributors shall be specified for PVT-independent modeling of static power per mode and energy per event of IP primitives, using the **PowerContributor** and **EnergyContributor** elements.

7.1 PowerContributor

Each **PowerContributor** shall reference contributors defined in the **Library**-level **Contributors** element (see 5.7). Table-driven power modeling (see Clause 8) can be used to model those circuits that are not conveniently represented by the reduced transistor contributor method.

PowerContributor attributes

name (required data name)—References a **PowerContributor** from the **Library**-level **ModelContributors** element (see 5.7).

quantity (required value)—Number of power contributors.

width (required value)—Gate width.

length (required value)—Gate length.

```
<Modes mutuallyExclusive =”value”>
<Modes>
    <Mode name=”mode_name” when=”logical_expression”>
        <PowerContributor name=”contributor_name”
          quantity=”number” width=”value” length=”value”/>
        ...
    </Mode >
</Modes >
```

When using a **Table** to represent power contributor data (see Clause 8), the table data shall be specified over the range of Voltage, Temperature, and Process values or only over Voltage and Temperature. In the former case, a `3D` table with Indices of `P`, `V`, and `T` is used, while the latter case only requires `V` and `T` as indices.

For an example of a `2D` VT power **Table**, see B.9.2.

7.2 EnergyContributor

Each **EnergyContributor** subelement shall have a name (referencing an **EnergyContributor** from the **Library**-level **Contributors** element; see 5.7), a quantity, and a capacitance. The units for the capacitance attribute are specified in the **Library**-level **Units** element with the **capacitanceUnit** (see 5.3).

EnergyContributor attributes

name (required data name)—Name of contributor defined in the **Library**-level **Contributors** element (see 5.7).

quantity (optional value)—Number of contributors.

capacitance (required value)—Effective switched capacitance for specified contributor.

Several different types of units may be specified for the table data, such as current, power, switched capacitance, or energy. The **units** attribute to the **Table** specifies the data units.

When using a **Table** to represent energy contributor data (see Clause 8), the table data shall be specified over the range of Voltage and Temperature, as illustrated in B.9.3.

8. Tables

The value of certain elements can be a scalar (see 4.2.4), an expression (see 9.2), or a multidimensional table. The **Table** element (see 8.2) represents a data array of indexed values. At the lowest (data) level, the values are compound strings of multiple values contained in the content of **RowArray** or **Row** elements (see 8.4.1), with each element in a row separated by a **separator** character and the rows separated by a **rowSeparator** character (see 8.1).

8.1 separator, RowSeparator, and ArraySeparator

The default **separator** is the comma (**,**), and the default **rowSeparator** is the newline character. These characters are mapped directly from the contents of a comma separated value (CSV) file. Leading and trailing whitespace is ignored. The separator character for one-dimensional arrays is independent of the separator character for multidimensional arrays.

The separator value for one-dimensional arrays can be set in the **Library** (see 5.1), **Cell** (see 6.1), or **Table** (see 8.2) element via a **RowSeparator** element. This element also defines the **valueType** of the individual elements in the array.

RowSeparator attributes

separator (optional single character)—Separator character for single-dimensional compound strings. The default value is a comma (**,**).

valueType (optional data type)—Datatype individual elements of the compound string: one of **string**, **float** (the default), **int**, and **mixed**.

A *multidimensional array* is built from two-dimensional compound strings. Each string contains the individual values accessed through the **rowIndex** and **columnIndex**. The string is broken into rows and columns using individual separator characters defined by the **ArraySeparator** element. This element can

be used in the **Library** (see 5.1), **Cell** (see 6.1), or **Table** (see 8.2) elements. This element also defines the **valueType** of the individual elements of the array.

ArraySeparator attributes

separator (optional single character)—Separator character for single-dimensional compound strings. The default value is a comma (**,**).

rowSeparator (optional single character)—Separator character for a compound string that separates the individual rows. The default value is the new line, or carriage return, character.

valueType (optional data type)—Datatype individual elements of the compound string: one of **string**, **float** (the default), **int**, and **mixed**.

Independent **separator** and **rowSeparator** characters shall be specified as attributes on individual elements that define compound data such as **Table** (see 8.2), **Index** (see 8.3), **Arrays** (see 8.4.2), and **RowArray** (see 8.4.1). When the **separator** character is set for an individual element, it is applied to all compound elements in the scope of that element.

8.2 valueType

The individual values in the compound string are of a specific **valueType**; the default is `"float"`. The possible values for **valueType** are **"float"**, **"int"**, **"string"**, and **"mixed"**. The separator, **rowSeparator**, and **valueType** can be set as attributes on the individual elements to override the default or inherited setting for the contents of that element. The **valueType** can be specified on the individual **Table**, **Index** (see 8.3), **Arrays** (see 8.4.2), and **RowArray** (see 8.4.1) elements.

Table attributes

name (required data name)—Name of the table.

units (required data name)—SI units of the values of the data array.

separator (optional single character)—Separator character for single-dimensional compound strings. The default value is a comma (**,**).

rowSeparator (optional single character)—Separator character for a compound string that separates the individual rows. The default value is the new line, or carriage return, character.

valueType (optional data type)—Datatype individual elements of the compound string: one of **string**, **float** (the default), **int**, and **mixed**.

```
<Table name="data_name" units="SiUnits" valueType="data_type">
    <Indices numDimensions="integer">
        <Index name="data_name" units="SiUnits"
          value_0, value_1, ... value_n
        </Index>
        ...
    </Indices>
        "data array"
</Table>
```

The *data array* may be specified using either a single `2D` **RowArray** element or a series of `1D` **Row** elements (see 8.4).

8.3 Indices

The indices of the table are defined in the **Indices** element. The number of dimensions of the table is defined in the **Indices** integer attribute **numDimensions**. The **Indices** element contains one **Index** element for each dimension in the array. Each **Index** shall be declared in **ModelParameters** (see 5.4).

The **Index** defines an individual index for the array. The index is referenced by its name. The contents are a compound string of individual index values. The default is a comma-separated list of `float` values.

Indices attributes

numDimensions (required numeric value)—Number of dimensions in the data array; this shall be an integer value. There shall be one **Index** element for each dimension.

Index attributes

name (required data name)—Unique name used to define the index.

matchingFunction (optional data name)—Name of the function to use when searching for a match in this **Index** element.

separator (optional single character)—Separator character for single-dimensional compound strings. The default value is a comma (**,**).

valueType (optional data type)—Datatype individual elements of the compound string: one of **string**, **float** (the default), **int**, and **mixed**.

units (required data name)—SI units of the values of the data array.

```
<Indices numDimensions="numeric_value">
    <Index name="data_name" units="data_name" valueType="data_name">
        "values"
    </Index>
    <Index name="data_name" units="data_name" valueType="data_name">
        "values"
    </Index>
</Indices>
```

8.4 Data array

Depending on the number of dimensions of the array, the data are contained in a **RowArray**. For three or more dimensions, an **Arrays** element is required for each dimension beyond `2`.

8.4.1 Two-dimensional tables

A two-dimensional table consists of one **RowArray** (as illustrated in B.9.4) containing a data array of rows and columns. The two-dimensional table has two **Index** elements (see 8.3) used to index the rows and columns of the data array.

RowArray attributes

rowIndex (required data name)—Name of the index used to find an individual row in the array.

columnIndex (required data name)—Name of the index used to find individual elements in the array.

units (required data name)—SI units of the values of the data array.

separator (optional single character)—Separator character for single-dimensional compound strings. The default value is a comma (**,**).

rowSeparator (optional single character)—Separator character for a compound string that separates the individual rows. The default value is the new line, or carriage return, character.

valueType (optional data type)—Datatype individual elements of the compound string: one of **string**, **float** (the default), **int**, and **mixed**.

8.4.2 Multidimensional tables

Multidimensional tables can be represented in a **Table** (see 8.2). In the **Indices** element (see 8.3), the **numDimensions** attribute specifies the number of dimensions and includes an **Index** subelement for each dimension. A three-dimensional table has `numDimensions` set to `3` and contains three **Index** elements.

For each dimension greater than `2`, an **Arrays** element is specified (as illustrated in B.9.5). Each **Arrays** subelement has an index attribute that defines the **Index** (see 8.3) used to parse the array. The **Arrays** element can use a **RowArray** element (see 8.4.1), containing the actual data in two-dimensional arrays, or a nested **Arrays** element to form an additional dimension.

Arrays attributes

index (required data name)—Name of the index used to parse the set of arrays.

units (required data name)—SI units of the values of the data array.

separator (optional single character)—Separator character for single-dimensional compound strings. The default value is a comma (**,**).

rowSeparator (optional single character)—Separator character for a compound string that separates the individual rows. The default value is the new line, or carriage return, character.

valueType (optional data type)—Datatype individual elements of the compound string: one of **string**, **float** (the default), **int**, and **mixed**.

arrayIndex (required data name)—Zero-based index pointer of the containing array.

Arrays and **RowArrays** shall be nested using the index pointer attribute **arrayIndex** on those elements.

9. Expressions

Expressions are used when specifying a value to be calculated, such as specifying power consumption algebraically. Such a representation is especially useful during early system and architecture explorations. The expression is evaluated using the current values for the variables, and the result is returned.

9.1 Expression types

This standard supports two different types of expressions: logical expressions and algebraic expressions. These expressions may have a global scope for an entire library or may be local to a cell definition. They may be defined within the library file or in an external file. Expressions may be used by various elements.

9.1.1 Logical expressions

Logical expressions are assigned to the **when** attribute of elements such as **Mode** (see 6.3), **Event** (see 6.4), and **State** (see 6.6) to describe the logical condition of the cell pins or internal signals. The expression string consists of pin names; the logical operations `AND`, `OR`, `NOT`, and `XOR`; and the values `1` and `0`. Parentheses are used as separators. For example, the following logical expression describes a particular mode of a 2-input logic primitive:

```
<Mode when="(And (Not A) (Not B))">
```

Note that the logical operators are not binary; they shall operate on any number of pins.

Logical expressions are calculated by a power tool to determine whether a particular **Mode** or **Event** has occurred.

9.1.2 Algebraic expressions

Algebraic expressions are used to compute numerical values for library attributes. The expression string may be a real number; reference a global expression, variable, or parameter; or algebraically represent a value.

Algebraic expressions are Verilog-A expressions (see Accellera Systems Initiative, “Verilog-AMS Language Reference Manual, Version 2.4.0”) with real, integer, and string variables using built-in mathematical functions and references to global **Expression** elements (see 9.2). The variables are local (to the cell) variables set in the **CellParameters** element (see 6.5) or global **ModelParameters** elements (see 5.4), or they are supplied by the user. If different assignments of the variables have the same name, the user set variables override the **CellParameters**, which override the **ModelParameters**. A variable without a value shall cause the evaluation to fail.

The **ModelExpressions** (see 5.5), **PowerContributor** (see 7.1), and **EnergyContributor** (see 7.2) elements contain **Expression** elements that may be defined using the algebraic expression format if the expression can easily be represented in the XML library. The content of the **Expression** element contains the expression string, for example,

```
<Expression name="Energy2Power" > Energy*Frequency </Expression>
```


9.2 Expression element

Expressions representing a cell’s static and dynamic power consumption are defined in an **Expression** subelement to the **DynamicPower** and **StaticPower** cell-level elements (see 6.7). The content of the **Expression** subelement is the algebraic equation for calculating power or a reference call to a global expression from **ModelExpressions** (see 5.5).

Expressions are described using a subset of Verilog-A syntax and support real, integer, and string variables, plus global expression calls. Array, vector, net, and branch data types, access operators, and functions are not supported. All operators that are valid for real, integer, and string variables are supported and behave identically to their Verilog-A equivalents, including logical/conditional operators for real variables and string operations (including logical equality and concatenation). Expressions may use built-in mathematical functions (e.g., exp, sqrt, and abs) but not analog filter functions (e.g., ddt or limexp) or system tasks. See Accellera Systems Initiative, "Verilog-AMS Language Reference Manual, Version 2.4.0" (Chapter 4, "Expressions"), for more explanation of these constructs.

Parameters specified in IEEE Std 1801-2015 using `-power_expr` within IEEE 1801 power-state models may be used as expression variables, provided the parameters are defined at the cell level in the **CellParameters** element (see 6.5) or in the **Library-**level **ModelParameters** element (see 5.4).

Expression attributes

name (required data name)—Name of the expression, used when referencing the expression.

- Required for global expressions; defined in the **ModelExpressions** element (see 5.5).
- Optional in local expressions; useful for user documentation.

units (optional data name)—Sets the units for the value returned by the expression. This value may reference a **Units** element (see 5.3) or be a scaled SI unit.

```
<Expression name="exprname" units="powerUnit">
    expressionString
</Expression >
```

For an example, see 9.4.3.

9.3 ExpressionParameters

Expressions may reference **ExpressionParameters**, which in effect serve as local variables. The **ExpressionParameters** element contains one or more Parameters used in the parent element's **Expression** (see 9.2).

Parameter attributes

name (required data name)—Parameter name.

value (optional data name)—Parameter value.

For an example, see B.6.

9.4 Global and local expressions

Expressions may have a global or a local scope, depending on where they are defined. *Global expressions* are defined in the **ModelExpressions** element (see 5.5) or contained in a file referenced in the **ModelExpressions** element. Global expressions require a **name** attribute (see 9.2) that is used when referencing the expression in another global or local expression. The expression name value shall be unique in the **ModelExpressions** element and in any referenced files.

Local expressions are defined within a **Cell** (see 6.1) and shall not be referenced for reuse. Local expressions shall be named but only for identification.

9.4.1 Global expressions

Expressions defined in the **Library**-level **ModelExpressions** element (see 5.5) are globally reusable expressions. These expressions shall be defined in the content of the **Expression** subelement (see 9.2) or in an external file defined by the **File** subelement (see 5.5). The **Expression** subelements shall be named using the **name** attribute and can be referenced by their name in any other **Expression** element in the **Library**.

NOTE—Defining global expressions in a file separate from the model library file can be useful when an expression is composed of multiple complex statements or when there is a need to separate the expressions from the power models.[8]

For an example, see B.10.2.

9.4.2 Referencing a global expression

A global expression is called by **name** (see 9.2). The expression name is followed by parentheses (**()**) that shall contain arguments to the expression. The argument values are passed by **name** and supply values to variables in the expressions, overriding any default value for the variable. Other variables get their values extracted from the instance, as supplied by the user, or defined at the cell level in the **CellParameters** element (see 6.5) or in the **Library**-level **ModelParameters** element (see 5.4). All variables used in the expression need to have a value if the expression is to evaluate correctly.

For an example, see B.10.2.

9.4.3 Local expressions

Local expressions are expressions defined where they are used. They reference global expressions or global variables by **name** (see 9.2). Full parameter names or aliases shall be used. The units for the expression are optionally specified as an attribute of the containing element. If the units are not specified, then the units resulting from the evaluation of the expression are set by the **powerUnit** attribute in the **Library**-level **Units** element (see 5.3).

For an example, see B.10.3.

9.4.4 Using expressions to reference contributors

Full PVT-independent expressions may be specified by referencing contributors (see Clause 7). Contributors are referenced by calling the event or mode of interest by using the **EvalEnergyContributor()** or **EvalPowerContributor()** functions, respectively, with the named event or mode. Use of either function implies evaluation by the Contributor Evaluator (see 4.1.2 and 4.2.1).

For examples, see B.10.4.

[8] Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

9.4.5 Expression precedence

When a named expression is present in more than one location (locally within a **Cell** [see 6.1], globally within the **ModelExpressions** element [see 5.5], or globally within the **File** element [see 5.5]), the following precedence shall be observed:

— Expressions in the **ModelExpressions** element shall take precedence over similarly named expressions in the **File** element.
— Expressions in a **Cell** element shall take precedence over similarly named expressions in the **ModelExpressions** and **File** elements.

Annex A

(informative)

Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only.

[B1] Dhanwada, N., D. Hathaway, J. Frenkil, W. R. Davis, and H. Demircioglu, "Leakage power contributor modeling," *IEEE Design and Test of Computers*, vol. 29, no. 2, pp. 71–78, Mar./Apr. 2012.

[B2] IEEE Std 1685™-2014, IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows.

[B3] Synopsys, *Liberty User Guides and Reference Manual Suite*, Version 2013.03 and later.[9]

[B4] World Wide Web Consortium, "eXtensible Markup Language (XML) schema part 0" specification (http://www.w3.org/TR/xmlschema-0).[10]

[9] Liberty publications are available from Synopsys (https://www.synopsys.com/).
[10] XML publications are available from the World Wide Web Consortium (http://www.w3.org).

Annex B

(informative)

Example models

Multiple examples are shown in this annex, along with brief descriptions of pertinent features of each.

B.1 Full library example

An example **Library** is shown as follows. The example contains all required elements. For brevity's sake, only a single **Cell** element has been included as several different **Cells** are shown as separate examples:

```
<!--
#####################################################################
Example Library
#####################################################################
-->
<Library name="ExampleLib" version="V3.0b0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="OpenLowPower"
    xsi:schemaLocation="OpenLowPower ../xsd/SystemLowPower3.0_b1.xsd"
    condition="TT_1.0V_0C_100M"
    datagen="measured"
>
    <annotation>
        <documentation>
            This library file contains example UPM models.
        </documentation>
    </annotation>

    <Technology name="22FDSOI" version="v1.0">
        <Process name="22nmXC01G" version="SS"/>
    </Technology>

    <!—
    ###################################################################
        The following 'Units' are the global default Units.
    ###################################################################
    -->
    <Units>
        <Unit name="capacitanceUnit" value="pF"/>
        <Unit name="resistanceUnit" value="ohm"/>
        <Unit name="dimensionUnit" value="nM"/>
        <Unit name="frequencyUnit" value="Hz"/>
        <Unit name="powerUnit" value="mW"/>
        <Unit name="energyUnit" value="pJ"/>
        <Unit name="voltageUnit" value="V"/>
        <Unit name="temperatureTkUnit" value="K"/>
        <Unit name="temperatureTcUnit" value="C"/>
        <Unit name=”timeUnit” value=”S”/>
    </Units>
```

```
<!–
#####################################################################
Define the Power Model parameters.
Any parameter needed by an 1801 parameterized model must be
    defined here before usage within an expression.
#####################################################################
-->
<ModelParameters>
    <Constant name="k" value="1.381e-23"/>
    <Constant name="q" value="1.602e-19"/>
    <Parameter name="processCorner" alias="P" units="Enumerated"/>
    <Parameter name="temperatureC" alias="Ti"
      units="temperatureTcUnit" min="-50" max="150" value="100"/>
    <Parameter name="temperatureK" alias="Tk"
      units="temperatureTkUnit" />
    <Parameter name="voltage" alias="Vi" units="voltageUnit"/>
    <Parameter name="internalState" alias="S" units="Enumerated"/>
    <Parameter name="frequency" alias="F" units="frequencyUnit"/>
     <Parameter name="width" alias="W" units=”dimensionUnit”/>
</ModelParameters>

<!–
#####################################################################
The ModelExpressions element defines expressions for reuse.
In the Leakage_Scaling_Expression below, Ti is the instance
    temperature, Vi is the instance voltage, Tnom is the nominal,
    or characterization, temperature and Vnom is the nominal
    voltage.
#####################################################################
-->
<ModelExpressions>
        <Constant name="k" value="1.381e-23"/>
        <Constant name="q" value="1.602e-19"/>
        <Expression name="Leakage_Scaling_Expression"
          powerUnit="W">
            ((Ti/Tnom)^0.5)*
            ((1/(0.4+(k*Ti/q))/(1/(0.4+(k*Tnom/q))))^0.5*
            (1-exp(-(Vi/(2*k*Ti/q))))*
            exp(-(0.7-(0.4+(k*Ti/q)^0.5)+0.08)/(3*k*Ti/q))/
            (1-exp(-(Vnom/(2*k*Ti/q))))*
            exp(-(0.7-(0.4+(k*Tnom/q)^0.5)+0.08)/(3*k*Tnom/q))
        </Expression>
</ModelExpressions>

<!–
#####################################################################
Define the characterization conditions for cells that have PVT
    specific or hard coded data.
#####################################################################
-->
<ModelConditions name="FF_1.25V_0C_100M">
    <Condition name="P" value="FF"/>
    <Condition name="Vnom" value="1.25"/>
    <Condition name="Tnom" value="273.15"/>
    <Condition name="F" value="1e8"/>
</ModelConditions>
```

```
<ModelConditions name="TT_1.0V_0C_100M">
    <Condition name="P" value="TT"/>
    <Condition name="Vnom" value="1.0"/>
    <Condition name="Tnom" value="273.15"/>
    <Condition name="F" value="1e8"/>
</ModelConditions>

<ModelConditions name="TT_1.0V_100C_100M">
    <Condition name="P" value="TT"/>
    <Condition name="Vnom" value="1.0"/>
    <Condition name="Tnom" value="373.15"/>
    <Condition name="F" value="1e8"/>
</ModelConditions>

<!-
#################################################################
Define a set of Power Contributors to be used cell definitions.
#################################################################
NOTE: For purposes of brevity, all but 2 of the PowerContributor
  definitions have been removed.
#################################################################
-->
<ModelContributors>
    <PowerContributor name="CONTRIB_1_CHANNEL_2_N_1_NETS"
    type="subthreshold_leakage">
        <Device type="N_planar">
            <Values>
                <Temperature name="Temp_t">
                    <ExpressionParameters>
                        <Parameter name="coef01" value="0.0020694663"/>
                        <Parameter name="coef02" value="0.0009647095"/>
                        <Parameter name="coef03" value="0.0004784309"/>
                        <Parameter name="coef04" value="0.0110506050"/>
                        <Parameter name="coef05" value="-20.82448499"/>
                        <Parameter name="coef06" value="-0.700959e-5"/>
                        <Parameter name="coef07" value="0.3062919e-5"/>
                    </ExpressionParameters>
                    <Expression>
                        (coef01 - coef02/(W*1e-3)) + (coef03 +
                         coef04*exp(coef05*(W*1e-3)))*Vi + (coef06 +
                        (coef07)/(W*1e-3))*Tk
                    </Expression>
                </Temperature>
                <Voltage name="Vds">
                    <ExpressionParameters>
                        <Parameter name="coef08" value="0.0219981429"/>
                        <Parameter name="coef09" value="0.0005980254"/>
                        <Parameter name="coef10" value="0.1052698895"/>
                        <Parameter name="coef11" value="0.0038615749"/>
                        <Parameter name="coef12" value="-0.243442e-4"/>
                        <Parameter name="coef13" value="0.2557680e-4"/>
                    </ExpressionParameters>
                    <Expression>
                        (coef08 + coef09/(W*1e-3)) + (coef10 -
                         coef11/(W*1e-3))*Vi + (coef12 -
                        (coef13)*(W*1e-3))*Tk
                    </Expression>
                </Voltage>
```

```
                <Voltage name="Vgs">
                    <ExpressionParameters>
                        <Parameter name="coef14" value="0.8028277e-6"/>
                        <Parameter name="coef15" value="0.1631028e-6"/>
                        <Parameter name="coef16" value="-0.658395e-7"/>
                        <Parameter name="coef17" value="0.1277801e-7"/>
                        <Parameter name="coef18" value="-0.239969e-8"/>
                        <Parameter name="coef19" value="0.0628823e-8"/>
                    </ExpressionParameters>
                    <Expression>
                        (coef14 + coef15/(W*1e-3)) + (coef16 -
                         coef17/(W*1e-3))*Vi +
                        (coef18 - coef19/(W*1e-3))*Tk
                    </Expression>
                </Voltage>
                <Voltage name="Vbs">
                    <ExpressionParameters>
                        <Parameter name="coef20" value="0.0334021503"/>
                        <Parameter name="coef21" value="0.0006022167"/>
                        <Parameter name="coef22" value="0.0382726521"/>
                        <Parameter name="coef23" value="0.0012660644"/>
                        <Parameter name="coef24" value="-0.749893e-4"/>
                        <Parameter name="coef25" value="0.0237484e-4"/>
                    </ExpressionParameters>
                    <Expression>
                        (coef20 - coef21/(W*1e-3)) +
                        (coef22 - coef23/(W*1e-3))*Vi +
                        (coef24 + coef25/(W*1e-3))*Tk
                    </Expression>
                </Voltage>
            </Values>
        </Device>
    </PowerContributor>
    <PowerContributor name="CONTRIB_2_GATE_1_N" type="gate_leakage">
        <Device type="N_planar">
            <Values>
                <Voltage name="Vbs">
                    <ExpressionParameters>
                        <Parameter name="coef1" value="-0.0427013606"/>
                        <Parameter name="coef2" value="-0.0635706757"/>
                        <Parameter name="coef3" value="-3.9393867430"/>
                        <Parameter name="coef4" value="-7.8243626190"/>
                        <Parameter name="coef5" value="8.21468563182"/>
                        <Parameter name="coef6" value="0.00294662222"/>
                        <Parameter name="coef7" value="0.00385683148"/>
                        <Parameter name="coef8" value="-0.0044298706"/>
                        <Parameter name="coef9" value="0.00648406735"/>
                    </ExpressionParameters>
                    <Expression>
                        (coef1 + coef2*exp(coef3*(W*1e-3))) + (coef4 +
                         coef5*exp(coef6/(W*1e-3)))*Vi +
                        (coef7 + coef8*exp(coef9/(W*1e-3)))*Tk
                    </Expression>
                </Voltage>
                <Temperature name="Temp_t">
                    <Expression>
                        0
                    </Expression>
                </Temperature>
                <Voltage name="Vds">
```

```
                    <Expression>
                        0
                    </Expression>
                </Voltage>
                <Voltage name="Vgs">
                    <Expression>
                        Vi
                    </Expression>
                </Voltage>
            </Values>
        </Device>
    </PowerContributor>
    <PowerContributor name="CONTRIB_3_GATE_1_P" type="gate_leakage">
        <Device type="P_planar">
            <Values>
                <Voltage name="Vbs">
                    <ExpressionParameters>
                        <Parameter name="coef1" value="0.20781190044"/>
                        <Parameter name="coef2" value="0.05133095381"/>
                        <Parameter name="coef3" value="-7.6060277496"/>
                        <Parameter name="coef4" value="0.57411234012"/>
                        <Parameter name="coef5" value="0.04539081277"/>
                        <Parameter name="coef6" value="-6.4205843261"/>
                        <Parameter name="coef7" value="-0.0015552468"/>
                        <Parameter name="coef8" value="-0.0001714597"/>
                        <Parameter name="coef9" value="-6.8380528934"/>
                    </ExpressionParameters>
                    <Expression>
                        (coef1 + coef2*exp(coef3*(W*1e-3))) + (coef4 +
                         coef5*exp(coef6*(W*1e-3)))*Vi + (coef7 +
                         coef8*exp(coef9*(W*1e-3)))*Tk
                    </Expression>
                </Voltage>
                <Temperature name="Temp_t">
                    <Expression>
                        0
                    </Expression>
                </Temperature>
                <Voltage name="Vds">
                    <Expression>
                        0
                    </Expression>
                </Voltage>
                <Voltage name="Vgs">
                    <Expression>
                        Vi
                    </Expression>
                </Voltage>
            </Values>
        </Device>
    </PowerContributor>
    <PowerContributor name="CONTRIB_4_CHANNEL_1_P"
    type="subthreshold_leakage">
        <Device type="P_planar">
            <Values>
                <Temperature name="Temp_t">
                    <ExpressionParameters>
                        <Parameter name="coef01" value="-0.085666138"/>
                        <Parameter name="coef02" value="0.0086905359"/>
                        <Parameter name="coef03" value="0.1260781576"/>
```

```
                <Parameter name="coef04" value="0.0461679989"/>
                <Parameter name="coef05" value="-0.008749651"/>
                <Parameter name="coef06" value="0.0969180178"/>
                <Parameter name="coef07" value="0.0001940220"/>
                <Parameter name="coef08" value="-0.000010950"/>
                <Parameter name="coef09" value="0.1582359733"/>
            </ExpressionParameters>
            <Expression>
                (coef01 + coef02*exp(coef03/(W*1e-3))) +
                (coef04 + coef05*exp(coef06/(W*1e-3)))*Vi +
                (coef07 + coef08*exp(coef09/(W*1e-3)))*Tk
            </Expression>
        </Temperature>
        <Voltage name="Vds">
            <ExpressionParameters>
                <Parameter name="coef10" value="-6.947116-12"/>
                <Parameter name="coef11" value="0.9999999999"/>
                <Parameter name="coef12" value="2.905476e-14"/>
            </ExpressionParameters>
            <Expression>
                coef10 + coef11*Vi + coef12*Tk
            </Expression>
        </Voltage>
        <Voltage name="Vgs">
            <ExpressionParameters>
                <Parameter name="coef13" value="0.0001859923"/>
                <Parameter name="coef14" value="0.0021400175"/>
                <Parameter name="coef15" value="-0.263883438"/>
                <Parameter name="coef16" value="0.0000150604"/>
                <Parameter name="coef17" value="-0.000428237"/>
                <Parameter name="coef18" value="-0.283619759"/>
                <Parameter name="coef19" value="-0.000000717"/>
                <Parameter name="coef20" value="-0.000006675"/>
                <Parameter name="coef21" value="-0.258341190"/>
            </ExpressionParameters>
            <Expression>
                (coef13 + coef14*exp(coef15/(W*1e-3))) +
                (coef16 + coef17*exp(coef18/(W*1e-3)))*Vi +
                (coef19 + coef20*exp(coef21/(W*1e-3)))*Tk
            </Expression>
        </Voltage>
        <Voltage name="Vbs">
            <ExpressionParameters>
                <Parameter name="coef22" value="0.1532846850"/>
                <Parameter name="coef23" value="0.0000031389"/>
                <Parameter name="coef24" value="0.4984048047"/>
                <Parameter name="coef25" value="0.1988633830"/>
                <Parameter name="coef26" value="-0.000001849"/>
                <Parameter name="coef27" value="0.6224635543"/>
                <Parameter name="coef28" value="-0.000405297"/>
                <Parameter name="coef29" value="0.0001087206"/>
                <Parameter name="coef30" value="0.2105971177"/>
            </ExpressionParameters>
            <Expression>
                (coef22 + coef23*exp(coef24/(W*1e-3))) +
                (coef25 + coef26*exp(coef27/(W*1e-3)))*Vi +
                (coef28 + coef29*pow(coef30 - (W*1e-3),2))*Tk
            </Expression>
        </Voltage>
    </Values>
```

```
    </Device>
</PowerContributor>
<PowerContributor name="CONTRIB_5_CHANNEL_2_N_1_NETS"
type="subthreshold_leakage">
    <Device type="N_planar">
        <Values>
            <Temperature name="Temp_t">
                <ExpressionParameters>
                    <Parameter name="coef01" value="0.0217902744"/>
                    <Parameter name="coef02" value="-0.035202996"/>
                    <Parameter name="coef03" value="0.0626654458"/>
                    <Parameter name="coef04" value="0.7058009605"/>
                    <Parameter name="coef05" value="-17.94880840"/>
                    <Parameter name="coef06" value="-0.769982e-4"/>
                    <Parameter name="coef07" value="0.8815349e-4"/>
                </ExpressionParameters>
                <Expression>
                    (coef01 + coef02/(W*1e-3)) + (coef03 +
                     coef04*exp(coef05*(W*1e-3)))*Vi +
                    (coef06 + coef07/(W*1e-3))*Tk
                </Expression>
            </Temperature>
            <Voltage name="Vds">
                <ExpressionParameters>
                    <Parameter name="coef08" value="-0.023506391"/>
                    <Parameter name="coef09" value="-0.000687161"/>
                    <Parameter name="coef10" value="0.8930143764"/>
                    <Parameter name="coef11" value="0.0037462969"/>
                    <Parameter name="coef12" value="0.3091028e-4"/>
                    <Parameter name="coef13" value="0.2680432e-4"/>
                </ExpressionParameters>
                <Expression>
                    (coef08 + coef09/(W*1e-3)) + (coef10 +
                     coef11/(W*1e-3))*Vi +
                    (coef12 + coef13*(W*1e-3))*Tk
                </Expression>
            </Voltage>
            <Voltage name="Vgs">
                <ExpressionParameters>
                    <Parameter name="coef14" value="0.0000094233"/>
                    <Parameter name="coef15" value="0.0000602276"/>
                    <Parameter name="coef16" value="0.1423098207"/>
                    <Parameter name="coef17" value="-0.404547e-5"/>
                    <Parameter name="coef18" value="0.0052073e-5"/>
                    <Parameter name="coef19" value="-0.000000026"/>
                    <Parameter name="coef20" value="-0.000000342"/>
                    <Parameter name="coef21" value="0.1761933998"/>
                </ExpressionParameters>
                <Expression>
                    (coef14 + coef15*pow(coef16 - (W*1e-3),2)) +
                    (coef17 + coef18/(W*1e-3))*Vi +
                    (coef19 + coef20*pow(coef21 - (W*1e-3),2))*Tk
                </Expression>
            </Voltage>
            <Voltage name="Vbs">
                <ExpressionParameters>
                    <Parameter name="coef22" value="0.2175549235"/>
                    <Parameter name="coef23" value="-0.189965801"/>
                    <Parameter name="coef24" value="0.1086394452"/>
                    <Parameter name="coef25" value="0.2409653409"/>
```

```
                    <Parameter name="coef26" value="0.0607190976"/>
                    <Parameter name="coef27" value="-22.16430318"/>
                    <Parameter name="coef28" value="-0.577307e-3"/>
                    <Parameter name="coef29" value="0.0967192e-3"/>
                </ExpressionParameters>
                <Expression>
                    (coef22 + coef23*pow(coef24 - (W*1e-3),2)) +
                    (coef25 + coef26*exp(coef27*(W*1e-3)))*Vi +
                    (coef28 + coef29*(W*1e-3))*Tk
                </Expression>
            </Voltage>
        </Values>
    </Device>
</PowerContributor>
<PowerContributor name="CONTRIB_6_CHANNEL_1_N"
type="subthreshold_leakage">
    <Device type="N_planar">
        <Values>
            <Temperature name="Temp_t">
                <ExpressionParameters>
                    <Parameter name="coef01" value="-0.146853839"/>
                    <Parameter name="coef02" value="-1.155693908"/>
                    <Parameter name="coef03" value="-15.12874073"/>
                    <Parameter name="coef04" value="0.0791537150"/>
                    <Parameter name="coef05" value="0.6446569662"/>
                    <Parameter name="coef06" value="-14.78037540"/>
                    <Parameter name="coef07" value="0.0003454224"/>
                    <Parameter name="coef08" value="0.0030980485"/>
                    <Parameter name="coef09" value="-16.07366603"/>
                </ExpressionParameters>
                <Expression>
                    (coef01 + coef02*exp(coef03*(W*1e-3))) +
                    (coef04 + coef05*exp(coef06*(W*1e-3)))*Vi +
                    (coef07 + coef08*exp(coef09*(W*1e-3)))*Tk
                </Expression>
            </Temperature>
            <Voltage name="Vds">
                <ExpressionParameters>
                    <Parameter name="coef10" value="0.0039538766"/>
                    <Parameter name="coef11" value="-0.000000002"/>
                    <Parameter name="coef12" value="0.9996104963"/>
                    <Parameter name="coef13" value="0.0150140533"/>
                    <Parameter name="coef14" value="0.2442505212"/>
                    <Parameter name="coef15" value="-1.27672e-05"/>
                </ExpressionParameters>
                <Expression>
                    (coef10 + coef11/pow((W*1e-3),5)) +
                    (coef12 + coef13*pow(coef14 - (W*1e-3),2))*Vi +
                     coef15*Tk
                </Expression>
            </Voltage>
            <Voltage name="Vgs">
                <ExpressionParameters>
                    <Parameter name="coef16" value="0.0000091685"/>
                    <Parameter name="coef17" value="0.0000007990"/>
                    <Parameter name="coef18" value="5.0524833366"/>
                    <Parameter name="coef19" value="-0.000003665"/>
                    <Parameter name="coef20" value="-0.000003181"/>
                    <Parameter name="coef21" value="-0.200945585"/>
                    <Parameter name="coef22" value="-0.000000030"/>
```

```
                        <Parameter name="coef23" value="-0.000000193"/>
                        <Parameter name="coef24" value="0.1539378757"/>
                    </ExpressionParameters>
                    <Expression>
                        (coef16 + coef17*exp(coef18*(W*1e-3))) +
                        (coef19 + coef20*exp(coef21/(W*1e-3)))*Vi +
                        (coef22 + coef23*pow(coef24 - (W*1e-3),2))*Tk
                    </Expression>
                </Voltage>
                <Voltage name="Vbs">
                    <ExpressionParameters>
                        <Parameter name="coef25" value="0.2555722750"/>
                        <Parameter name="coef26" value="-0.005831548"/>
                        <Parameter name="coef27" value="0.0685074474"/>
                        <Parameter name="coef28" value="0.2291718808"/>
                        <Parameter name="coef29" value="0.0189376669"/>
                        <Parameter name="coef30" value="0.0501894114"/>
                        <Parameter name="coef31" value="-0.1622547e-3"/>
                        <Parameter name="coef32" value="0.0202915e-3"/>
                    </ExpressionParameters>
                    <Expression>
                        (coef25 + coef26*exp(coef27/(W*1e-3))) +
                        (coef28 + coef29*exp(coef30/(W*1e-3)))*Vi +
                        (coef31 + coef32*(W*1e-3))*Tk
                    </Expression>
                </Voltage>
            </Values>
        </Device>
    </PowerContributor>
    <EnergyContributor name='switched_cap" type="switched_cap">
        <Expression>
            capacitance * Vi * Vi
        </Expression>
    </EnergyContributor>
    <EnergyContributor name="wire_cap" type="wire_cap">
        <Expression>
            capacitance * Vi * Vi
        </Expression>
    </EnergyContributor>
</ModelContributors>

<!—
#####################################################################
Cell-Level Power Models
#####################################################################
NOTE: For purposes of brevity, all but 1 cell models have been
    removed.
#####################################################################
-->
<Cell name="INV_X1">
    <Pins>
        <Pin name="A" direction="input" type=”signal”
          capacitance="2.29"/>
        <Pin name="ZN" direction="output" type=”signal”/>
        <Pin name="VDD" direction="input" type=”primary_power”/>
        <Pin name="VSS" direction="input" type=”primary_ground”/>
    </Pins>
    <Modes mutuallyExclusive="true">
```

```
            <Mode name="M0" when="(Not A)">
                <PowerContributor name="CONTRIB_6_CHANNEL_1_N"
                  quantity="1" width="090" length="50"/>
                <PowerContributor name="CONTRIB_3_GATE_1_P"
                  quantity="1" width="135" length="50"/>
            </Mode>
                <Mode name="M1" when="(A)">
                <PowerContributor name="CONTRIB_2_GATE_1_N"
                  quantity="1" width="090" length="50"/>
                <PowerContributor name="CONTRIB_4_CHANNEL_1_P"
                  quantity="1" width="135" length="50"/>
            </Mode>
        </Modes>
        <Events>
            <Event name="ZNf" inputPin="A"
              outputTransition="rising" style="pinTransition">
                <EnergyContributor name="switched_cap" quantity="1"
                  capacitance="-0.248"/>
            </Event>
            <Event name="ZNr" inputPin="A"
              outputTransition="falling" style="pinTransition">
                <EnergyContributor name="switched_cap" quantity="1"
                 capacitance="2.062"/>
            </Event>
        </Events>
    </Cell>
</Library>
```

B.2 Units examples

The first **Units** element (unnamed) defines a set of default units, while the subsequent (named) **Units** element sets all of the units to meter, kilogram, second (MKS), overriding the default values, as follows:

```
<Units>
    <Unit name=”capacitanceUnit” value=”pF” />
    <Unit name=”resistanceUnit” value=”ohm” />
    <Unit name=”dimensionUnit” value=”nM” />
    <Unit name=”frequencyUnit” value=”Hz” />
    <Unit name=”powerUnit” value=”nW” />
    <Unit name=”energyUnit” value=”pJ” />
    <Unit name=”voltageUnit” value=”V” />
    <Unit name=”temperatureTkUnit” value=”K” />
    <Unit name=”temperatureTcUnit” value=”C” />
    <Unit name=”timeUnit” value=”S” />
</Units>

<Units name=”MKS”>
    <Unit name=”capacitanceUnit” value=”F” />
    <Unit name=”resistanceUnit” value=”ohm” />
    <Unit name=”dimensionUnit” value=”M” />
    <Unit name=”frequencyUnit” value=”Hz” />
    <Unit name=”powerUnit” value=”W” />
    <Unit name=”energyUnit” value=”J” />
    <Unit name=”voltageUnit” value=”V” />
    <Unit name=”temperatureTkUnit” value=”K” />
    <Unit name=”temperatureTcUnit” value=”C” />
    <Unit name=”timeUnit” value=”S” />
</Units>
```

The named set of **Units** is specified for an individual **Cell** with the **unitsSet** attribute. The following example sets the units for all parameters in the **Cell** to the values defined in the MKS set:

```
<Cell name="CPU" unitsSet="MKS">
    ...
</Cell>
```

Additionally, the **units** attribute can be used to set a single value (as opposed to using the **unitsSet** attribute, which sets values for all the **Unit** elements in the set). The following **units** attribute on the **States** element is used to set the power unit to milliwatts for all data in all states. In this case, MKS units apply everywhere in the **Cell** element, except for power defined in the **States** element and its subelements:

```
<Cell name="CPU" unitsSet="MKS">
   ...
   <States units="mW">
      <State name="Active">
         <DynamicPower value="scalar" />
         <StaticPower value="scalar" />
      </State>
      ...
   </States>
</Cell>
```

The **units** attribute can also be used to set a units value for a single **State**, as shown as follows:

```
<Cell name="CPU" unitsSet="MKS">
   ...
   <States>
      <State name="Active" units="mW">
         <DynamicPower value="scalar" />
         <StaticPower value="scalar" />
      </State>
      ...
   </States>
</Cell>
```

If more specification granularity is needed, the units attribute may be set on individual **DynamicPower** and **StaticPower** elements, as follows:

```
<Cell name="CPU" unitsSet="MKS">
   ...
   <States>
      <State name="Active">
         <DynamicPower value="scalar" units="mW" />
         <StaticPower value="scalar" units="uW" />
      </State>
      ...
   </States>
</Cell>
```

In this code snippet, the units for this cell are specified to be "MKS," thus, overriding the defaults. However, the units for **DynamicPower** and **StaticPower** are overridden for the Active state and are different from each other.

B.3 AND2 bit-level model with contributors

The following AND gate model uses both **PowerContributors** and **EnergyContributors**:

```
<Cell name="AND2_X1">
    <Pins>
        <Pin name="A1" direction="input" type=”signal”
          capacitance="1.99"/>
        <Pin name="A2" direction="input" type=”signal”
          capacitance="2.23"/>
        <Pin name="ZN" direction="output" type=”signal”/>
        <Pin name="VDD" direction="input" type=”primary_power”/>
        <Pin name="VSS" direction="input" type=”primary_ground”/>
    </Pins>
    <Modes mutuallyExclusive="true">
        <Mode name="M00" when="(And (Not A1) (Not A2))">
          <PowerContributor name="CONTRIB_1_CHANNEL_2_N_1_NETS"
          quantity="1" width="130" length="50"/>
          <PowerContributor name="CONTRIB_2_GATE_1_N" quantity="1"
          width="090" length="50"/>
          <PowerContributor name="CONTRIB_3_GATE_1_P" quantity="2"
          width="135" length="50"/>
          <PowerContributor name="CONTRIB_4_CHANNEL_1_P"
          quantity="1" width="135" length="50"/>
        </Mode>
        <Mode name="M10" when="(And A1 (Not A2))">
          <PowerContributor name="CONTRIB_5_CHANNEL_2_N_1_NETS"
          quantity="1" width="130" length="50"/>
          <PowerContributor name="CONTRIB_2_GATE_1_N" quantity="1"
          width="090" length="50"/>
          <PowerContributor name="CONTRIB_3_GATE_1_P" quantity="1"
          width="135" length="50"/>
          <PowerContributor name="CONTRIB_4_CHANNEL_1_P"
          quantity="1" width="135" length="50"/>
        </Mode>
        <Mode name="M01" when="(And (Not A1) A2)">
          <PowerContributor name="CONTRIB_6_CHANNEL_1_N"
          quantity="1" width="130" length="50"/>
          <PowerContributor name="CONTRIB_2_GATE_1_N" quantity="1"
          width="130" length="50"/>
          <PowerContributor name="CONTRIB_2_GATE_1_N" quantity="1"
          width="090" length="50"/>
          <PowerContributor name="CONTRIB_3_GATE_1_P" quantity="1"
          width="135" length="50"/>
          <PowerContributor name="CONTRIB_4_CHANNEL_1_P"
          quantity="1" width="135" length="50"/>
        </Mode>
        <Mode name="M11" when="(And A1 A2)">
          <PowerContributor name="CONTRIB_2_GATE_1_N" quantity="2"
          width="130" length="50"/>
          <PowerContributor name="CONTRIB_6_CHANNEL_1_N"
          quantity="1" width="090" length="50"/>
          <PowerContributor name="CONTRIB_4_CHANNEL_1_P"
          quantity="2" width="135" length="50"/>
          <PowerContributor name="CONTRIB_3_GATE_1_P" quantity="1"
          width="135" length="50"/>
```

```
    </Mode>
</Modes>
<Events>
    <Event name="A1r" when="(And (Not A1) (Not A2))" inputPin="A1"
    inputTransition="rising" style="pinTransition">
      <EnergyContributor name="switched_cap" quantity="1"
        capacitance="-1.040"/>
    </Event>
    <Event name="A2r" when="(And (Not A1)(Not A2))" inputPin="A2"
    inputTransition="rising" style="pinTransition">
      <EnergyContributor name="switched_cap" quantity="1"
      capacitance="-0.948"/>
    </Event>
    <Event name="A1f" when="(And A1 (Not A2))" inputPin="A1"
    inputTransition="falling" style="pinTransition">
      <EnergyContributor name="switched_cap" quantity="1"
      capacitance="1.310"/>
    </Event>
    <Event name="A2r_ZNr" when="(And A1 (Not A2))" inputPin="A2"
      inputTransition="rising" outputPin="ZN"
      outputTransition="rising" style="pinTransition">
        <EnergyContributor name="switched_cap" quantity="1"
        capacitance="1.733"/>
    </Event>
    <Event name="A1r_ZNr" when="(And (Not A1) A2))" inputPin="A1"
     inputTransition="rising" outputPin="ZN"
     outputTransition="rising" style="pinTransition">
        <EnergyContributor name="switched_cap" quantity="1"
        capacitance="1.733"/>
    </Event>
    <Event name="A2f" when="(And (Not A1) A2))" inputPin="A2"
     inputTransition="falling" style="pinTransition">
        <EnergyContributor name="switched_cap" quantity="1"
        capacitance="1.026"/>
    </Event>
    <Event name="A1f_ZNf" when="(And A1 A2)" inputPin="A1"
     inputTransition="falling" outputPin="ZN"
     outputTransition="falling" style="pinTransition">
        <EnergyContributor name="switched_cap" quantity="1"
        capacitance="3.341"/>
    </Event>
    <Event name="A2f_ZNf" when="(And A1 A2)" inputPin="A2"
     inputTransition="falling" outputPin="ZN"
     outputTransition="falling" style="pinTransition">
       <EnergyContributor name="switched_cap" quantity="1"
        capacitance="4.034"/>
        </Event>
    </Events>
</Cells>
```

B.4 LFSR bit-level model with contributors

A 4-bit LFSR is modeled at the bit level in the following example. This model defines four modes, according to the states of the input pins: `Active`, `Set`, `Doze`, and `Sleep`.

Note that aggregation—the combination of multiple similar contributors into a single equivalent contributor—can significantly reduce model size but has not been employed in this example. Also, this is an abstract model; no underlying structure in terms of hierarchy or netlists is defined or implied. Finally, some of the modes and events have been omitted herein for brevity's sake:

```
<Cell name="LFSR">
    <Pins>
        <Pin name="PW_ENB" direction="input" type=”signal”
          capacitance="9.998"/>
        <Pin name="CLK" direction="input" type=”clock”
          capacitance="1.991"/>
        <Pin name="CLK_EN" direction="input" type=”signal”
          capacitance="2.232"/>
         <Pin name="SET" direction="input" type=”signal”
          capacitance="2.229"/>
        <Pin name="OUTN" direction="output" type=”signal”/>
        <Pin name="VDD" direction="input" type=”primary_power”/>
        <Pin name="VSS" direction="input" type=”primary_ground”/>
    </Pins>
    <Modes mutuallyExclusive="false" defaultMode="ACTIVE"
     initialMode="ACTIVE">
       <Mode name="DOZE" when="(And (Not PW_ENB) (Not CLK_EN)
         (Not SET)) ">
         <PowerContributor name="CONTRIB_6_CHANNEL_1_N"
          quantity="1" width="130" length="50"/>
         <PowerContributor name="CONTRIB_2_GATE_1_N" quantity="1"
          width="130" length="50"/>
         <PowerContributor name="CONTRIB_2_GATE_1_N" quantity="1"
          width="090" length="50"/>
         <PowerContributor name="CONTRIB_3_GATE_1_P" quantity="1"
          width="135" length="50"/>
         <PowerContributor name="CONTRIB_4_CHANNEL_1_P"
          quantity="1" width="135" length="50"/>
         <PowerContributor name="CONTRIB_6_CHANNEL_1_N"
          quantity="1" width="090" length="50"/>
          ...
         <PowerContributor name="CONTRIB_3_GATE_1_P" quantity="1"
          width="135" length="50"/>
       </Mode>
       <Mode name=ACTIVE" when="(And (Not PW_ENB) CLK_EN (Not SET))">
       ...
       </Mode>
       <Mode name="SET" when="(And (Not PW_ENB) CLK_EN SET)">
       ...
       </Mode>
       <Mode name="SLEEP" when="(PW_ENB)">
       ...
       </Mode>
    </Modes>
    <Events>
        <Event name="Active_CLK_R" when="(And (Not PW_ENB) CLK_EN
          (Not Set))" inputPin="CLK" inputTransition="rising"
         style="pinTransition">
           <EnergyContributor name="switched_cap" quantity="1"
           capacitance="64.167"/>
        </Event>
        <Event name="Active_CLK_F" when="(And (Not PW_ENB) CLK_EN
```