BRITISH STANDARD

BS IEC 61523-3:2004

Delay and power calculation standards —

Part 3: Standard Delay Format (SDF) for the electronic design process

ICS 35.240.50

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

BSI Documents
Document Sharing Hub
British Standards

Full document details: BS IEC 61523-3:2004

National foreword

This British Standard reproduces verbatim IEC 61523-3:2004 and implements it as the UK national standard.

The UK participation in its preparation was entrusted to Technical Committee GEL/93, Design automation, which has the responsibility to:

— aid enquirers to understand the text;

— present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;

— monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

Cross-references

The British Standards which implement international publications referred to in this document may be found in the *BSI Catalogue* under the section entitled "International Standards Correspondence Index", or by using the "Search" facility of the *BSI Electronic Catalogue* or of British Standards Online.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the IEC title page, pages 2 to 90, an inside back cover and a back cover.

The BSI copyright notice displayed in this document indicates when the document was last issued.

Amendments issued since publication

Amd. No.	Date	Comments

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 15 December 2004

© BSI 15 December 2004

ISBN 0 580 44887 8

Documents
Document Sharing Hub

Full document details: BS IEC 61523-3:2004

INTERNATIONAL STANDARD

IEC 61523-3

First edition
2004-09

IEEE 1497™

Delay and power calculation standards –

Part 3:
Standard Delay Format (SDF) for the electronic design process

Reference number
IEC 61523-3(E):2004
IEEE Std. 1497(E):2001

Documents
Document Sharing Hub

Full document details: BS IEC 61523-3:2004

CONTENTS

FOREWORD 3

IEEE Introduction 7

1. Overview 8

1.1 Scope 8
1.2 Organization of this standard 8

2. References 9

3. Conventions 9

3.1 Terminology conventions 9
3.2 Syntactic conventions 9

4. SDF in the design process 12

4.1 Sharing of timing data 12
4.2 Using multiple SDF files in one design 12
4.3 Timing data and constraints 13
4.4 Timing environments 13
4.5 Back-annotation of timing data for design analysis 13
4.6 Forward-annotation of timing constraints for design synthesis 15
4.7 Timing models supported by SDF 16

5. Defining the standard delay format 18

5.1 SDF file content 18
5.2 Header section 20
5.3 Cells 25
5.4 Delays 28
5.5 Timing checks 46
5.6 Labels 60
5.7 Timing environment 62

Annex A (normative) Syntax of SDF 74

Annex B (informative) SDF file examples 84

Annex C (informative) List of Participants 89

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

INTERNATIONAL ELECTROTECHNICAL COMMISSION

DELAY AND POWER CALCULATION STANDARDS –

Part 3: Standard Delay Format (SDF) for the electronic design process

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 Publication(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. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements 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.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC 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 transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC/IEEE 61523-3 has been processed through IEC technical committee 93: Design automation.

The text of this standard is based on the following documents:

IEEE Std	FDIS	Report on voting
1497 (2001)	93/191/FDIS	93/196/RVD

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

This publication has been drafted in accordance with the ISO/IEC Directives.

The committee has decided that the contents of this publication will remain unchanged until 2006.

IEC 61523 consists of the following parts, under the general title *Delay and power calculation standards*:

IEC 61523-1, Part 1: *Integrated circuit delay and power calculation systems*

Documents
Document Sharing Hub

Full document details: BS IEC 61523-3:2004

IEC 61523-2, Part 2: *Pre-layout delay calculation specification of CMOS ASIC libraries*

IEC/IEEE 61523-3, Part 3: *Standard Delay Format (SDF) for the electronic process*

Documents
Document Sharing Hub

Full document details: BS IEC 61523-3:2004

IEC/IEEE Dual Logo International Standards

This Dual Logo International Standard is the result of an agreement between the IEC and the Institute of Electrical and Electronics Engineers, Inc. (IEEE). The original IEEE Standard was submitted to the IEC for consideration under the agreement, and the resulting IEC/IEEE Dual Logo International Standard has been published in accordance with the ISO/IEC Directives.

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

Use of an IEC/IEEE Dual Logo International Standard is wholly voluntary. The IEC and IEEE disclaim liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEC or IEEE Standard document.

The IEC and IEEE do not warrant or represent the accuracy or content of the material contained herein, and expressly disclaim any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEC/IEEE Dual Logo International Standards documents are supplied “AS IS”.

The existence of an IEC/IEEE Dual Logo International 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 IEC/IEEE Dual Logo International Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

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

In publishing and making this document available, the IEC and IEEE are not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Neither the IEC nor IEEE is undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEC/IEEE Dual Logo International Standards or IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations – Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received 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 interpretation requests except in those cases where the matter has previously received formal consideration.

Comments for revision of IEC/IEEE Dual Logo International Standards are welcome from any interested party, regardless of membership affiliation with the IEC or IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA and/or General Secretary, IEC, 3, rue de Varembé, PO Box 131, 1211 Geneva 20, Switzerland.

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

NOTE – 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 with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-3:2004

IEEE Standard for Standard Delay Format (SDF) for the Electronic Design Process

Sponsor

Design Automation Standards Committee of the IEEE Computer Society

Approved 5 December 2001

IEEE-SA Standards Board

Abstract: The Standard Delay Format (SDF) is defined in this standard. SDF is a textual file format for representing the delay and timing information of electronic systems. While both human and machine readable, in its most common usage it will be machine written and machine read in support of timing analysis and verification tools, and of other tools requiring delay and timing information. The primary audience for this standard is the implementors of tools supporting the format, but anyone with a need to understand the format's contents will find it useful.

Keywords: computer, computer languages, delay, delay backannotation, digital systems, electronic systems, hardware, hardware design, SDF, timing, timing analysis, timing backannotation, timing verification

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

IEEE Introduction

The Standard Delay Format (SDF) was designed to serve as a simple textual medium for communicating timing information and constraints between EDA tools. The original version was designed by Rajit C. Chandra in 1990 while at Cadence Design Systems, and was intended as a means of communicating macrocell and interconnect delays from Gate Ensemble to Verilog-XL, Veritime and other stand-alone tools requiring timing data.

Because it was originally targeted for annotation to tools using the Verilog language, many SDF constructs are analogous to those in Verilog specify blocks. Those already familiar with the Verilog specify block will find many of the SDF constructs familiar, such as SETUP and PATHPULSE. SDF also includes constructs for annotating interconnect delays, and can be used for forward annotation by specifying path delay constraints from timing analysis to floorplanners, and synthesis and layout tools.

SDF was first introduced into the EDA marketplace in 1991 where it won quick acceptance. Cadence placed SDF in the public domain in 1992 when it turned control over to Open Verilog International (OVI), and OVI delivered the first SDF standard, version 2.0, in June, 1993 (SDF version 1.0 was used by Cadence). OVI has since introduced version 2.1 in February, 1994, and version 3.0 in May, 1995. VHDL (IEEE 1076) also takes advantage of SDF through the VITAL standard.

In 1996 the OVI Board of Directors began an effort to establish SDF as an IEEE standard. With the approval of the IEEE Design Automation Standards Committee (DASC), the OVI Logic Modeling Technical Subcommittee became the IEEE SDF Study Group. With the approval of the Project Authorization Request (PAR) by the IEEE Standards Board on February 10, 1997, this group became the IEEE SDF Working Group.

This IEEE SDF standard builds upon OVI SDF version 3.0, and will be known as version 4.0. The changes from OVI 3.0 to IEEE 4.0 are small (LABEL construct added, NETDELAY construct restored), but the change from OVI standard to IEEE standard is significant, and so this is recognized by a new version number.

Objective

The starting point for the IEEE P1497 SDF Working Group was the OVI LRM version 3.0 SDF standard, with the goal of soliciting further enhancements and improving the quality and rigor of the LRM. Since SDF is already in widespread use, no modifications that would invalidate current usage were considered.

Acknowledgments

This standard is based on work originally developed by Cadence Design Systems, Inc. (in SDF 1.0) and Open Verilog International (in SDF 2.0, 2.1 and 3.0). The IEEE is grateful to Cadence Design Systems and Open Verilog International for permission to use their materials as the basis for this standard.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-3:2004

DELAY AND POWER CALCULATION STANDARDS –
Part 3: Standard Delay Format (SDF)
for the electronic design process

1. Overview

1.1 Scope

The Standard Delay Format (SDF) is an existing OVI standard for the representation and interpretation of timing data for use at any stage of the electronic design process. The ASCII data in the SDF file is represented in a tool and language independent way and includes path delays, timing constraint values, interconnect delays and high level technology parameters. This standard describes the IEEE version of the SDF standard.

This standard should serve as a complete specification of the Standard Delay Format (SDF). It contains:

— Detailed information on how SDF is used in the design process.
— Detailed semantic descriptions of all SDF constructs.
— The formal syntax.
— Examples.

1.2 Organization of this standard

A synopsis of the clauses and annexes of this standard is presented as a quick reference. There are five clauses and two annexes. All the clauses and annexes are normative parts of this standard, with the exception of Annex B (informative).

Clause 1: Overview—Content overview.

Clause 2: References—References to other applicable standards that are assumed or required for SDF.

Clause 3: Definitions and conventions—Introduction to syntactic style and the major syntactic components.

Clause 4: SDF in the design process—The role and use of SDF in the design process.

Clause 5: Defining the Standard Delay Format—The content of an SDF file. For each part of the file, the purpose is discussed, the syntax is specified, the semantics are explained, and examples are presented.

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

Annex A: Syntax of SDF—SDF file syntax description. The syntax of the contents of an SDF file is described in this annex.

Annex B: SDF file examples—Informative examples of SDF files.

2. References

This standard shall be used in conjunction with the following publications. When the following standards are superseded by an approved revision, the revision shall apply.

IEEE Std 1076, 2000 Edition, IEEE Standard VHDL Language Reference Manual.[1]

IEEE Std 1364-2001, IEEE Standard Verilog® Hardware Description Language.

3. Conventions

3.1 Terminology conventions

The verb "shall" is used throughout this standard to indicate mandatory requirements, whereas the verb "can" is used to indicate optional features that can be used at discretion. If "can" is used, however, one must follow the requirements set forth by the format definition. The verb "shall" denotes different meanings to different readers of this standard:

a) To the developers of tools that process SDF, the verb "shall" denotes a requirement that the standard imposes. The resulting implementation is required to enforce the requirements and to issue an error if the requirement is not met by the input.

b) To the human reader of SDF, the verb "shall" denotes that those characteristics of SDF are natural consequences of the format definition. The characteristics thereby implied in the SDF source text can be depended upon.

c) To the developer of tools that write SDF, and to the human writer of SDF, the verb "shall" denotes that those characteristics of SDF are natural consequences of the format definition. Adherence to the constraint implied by the characteristic is required.

3.2 Syntactic conventions

3.2.1 Syntactic conventions

The formal syntax of SDF is described using Backus-Naur Form (BNF). In addition, the following conventions are used:

a) Lowercase italic words, some containing embedded underscores, are used to denote syntactic tokens. For example:

   module_declaration

b) Boldface words are used to denote reserved keywords, operators, and punctuation marks as a required part of the syntax. For example:

   IOPATH
   (
)

[1]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (http://standards.ieee.org/).

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

c) A vertical bar separates alternative items unless it appears in boldface, in which case it stands for itself. In most cases each alternative appears on a separate line. For example:

character ::=
alphanumeric
|escaped_character

When the alternatives are very simple, as in the case of single characters, then they can appear on a single line or on consecutive multiple lines. For example:

decimal_digit ::= ***0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9***

d) Square brackets enclose optional items. For example:

real_number ::= integer [**.** *integer]*

e) Braces enclose a repeated item unless it appears in boldface, in which case it stands for itself. The item can appear zero or more times; the repetitions occur from left to right as with an equivalent left-recursive rule. Thus, this rules says that a CELL can contain any number of timing specifications:

cell ::= **(** ***CELL*** *celltype cell_instance { timing_spec }* **)**

`A constant-width` font is used for examples, file names, and while referring to constants, especially `0`, `1`, `x`, and `z` values.

3.2.2 Lexical tokens

An SDF file is a stream of lexical tokens in free format, each of which consists of one or more characters. Spaces and newlines serve only to separate tokens.

3.2.3 White space

Tabs, spaces, and newlines are considered white space. White space is never significant except when used within quoted strings or to separate lexical tokens.

3.2.4 Comments

Comments can be placed in SDF files using either the “C” or “C++” style.

“C”-style comments begin with /* and end with */. Nesting of “C”-style comments is not permitted. “C”-style comments can appear anywhere except within lexical tokens or quoted strings.

“C++”-style comments begin with // and continue until the end of the current line (the next newline character). Annotators shall ignore the double-slash and any text after them on any line in the file.

3.2.5 Identifiers

Identifiers can consist of alphanumeric characters and special characters. Alphanumeric characters consist of the letters of the alphabet, the numeric base-10 digits, the underscore (‘_’), and the dollar sign (‘$’). Special characters must be escaped (preceded with the backslash (‘\’) character) in order to be used in an identifier. The special characters are:

! " # $ % & « ( ) * + , - . / : ; < = > ? @ [\ ] ^ ‘ { | } ~

Any character can be escaped with a backslash, and the backslash is only required for special characters. Note that if a character normally has any special meaning in an identifier, this is lost when the character is escaped.

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

3.2.6 Quoted strings

A quoted string is a string of any legal SDF characters, including white space, that are enclosed between double-quotes (‘"’). Except for the double-quote itself, special characters lose their special meaning in a quoted string. The double-quote character may be included in a quoted string by escaping it [preceding it with the backslash (‘\’) character].

3.2.7 Bit specifications

A bit specification is indicated by an identifier with trailing paired square brackets (‘**[**’ and ‘**]**’). A single bit is indicated by a single integer between the square brackets, while a bit range is indicated by two integers separated by a colon (‘**:**’).

3.2.8 Hierarchy divider character

Either the period (‘**.**’) or the slash (‘/’) can be established as the hierarchy divider character, as described in 5.2.7. This character only has this special meaning when used to separate identifiers. An escaped hierarchy divider character loses its meaning as a hierarchy divider.

3.2.9 Data values

A number shall be an integer or a real number. Real numbers can be expressed in scientific notation, and can be signed or unsigned, but signed real numbers are not legal in all contexts.

A value consists of a real_number in parentheses, a *triple* in parentheses or an empty pair of parentheses. Empty parentheses indicate that no value is supplied for a particular data item. This is used primarily where a construct has a list of data items and it is desired to supply a value for an item further down the list but not for earlier items. The empty parentheses mark the places of the earlier items. An annotator shall take no action when it encounters empty parentheses. In particular, it shall not interpret this in the same way as a value of zero.

A *triple* consists of one, two or three colon-separated real_numbers. Each real_number corresponds to a data value in one of three data sets, commonly used (in order) as values under best case/minimum, nominal/typical and worst case/maximum operating conditions. If a real_number is omitted, then a value is not included for that data set. At least one real_number is required. Both colons must always be present.

Apart from allowing negative numbers (signed_real_number instead of real_number), *rvalue* and *rtriple* are essentially the same as *value* and *triple*.

For specifying delay values, *delval* extends *rvalue* by allowing two or three *rvalue* constructs to be grouped in a further set of parentheses. When this is used, the first *rvalue* specifies the delay, as if a single *rvalue* were given. The second specifies the pulse rejection limit, or “r-limit,” associated with this delay. The third specifies the X-limit, or “e-limit.” This allows pulse control data to be associated in a uniform way with all types of delays in SDF, i.e., **IOPATH**, **PORT**, **INTERCONNECT**, **NETDELAY**, and **DEVICE** delays. Note that since any *rvalue* can be an empty pair of parentheses, each type of delay data can be annotated or omitted as the need arises.

The meaning of *delval* constructs in an *delval_list* is different for lists of length one, two, three, six, or twelve. Lists of length four or five are interpreted in the same way as lists of length six with trailing empty parentheses. Similarly, lists of length seven to eleven are interpreted in the same way as lists of length twelve with trailing empty parentheses. A complete discussion of the use of *delval_list* is included in 5.4.1.

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

3.2.10 Operators

Operators are single-, double-, or triple-character sequences and are used in expressions.

The equality operators used in SDF conditional port expressions and timing check conditions return a logical value representing the result of the comparison, which is 1 for TRUE and 0 for FALSE, but can also be X.

`a == b` (logical equality) will be TRUE (1) only if `a` and `b` are of known logical state (0 or 1) and equal and FALSE (0) only if `a` and `b` are known and not equal. If either `a` or `b` is X or Z, then the result shall be X.

`a != b` (logical inequality) will be TRUE (1) only if `a` and `b` are known and not equal and FALSE (0) only if `a` and b are known and equal. If either `a` or `b` is X or Z, then the result will be X.

`a === b` (case equality) will be TRUE (1) if `a` and `b` are of the exact same logical state, including the X and Z states, and FALSE (0) otherwise.

`a !== b` (case inequality) will be TRUE (1) if `a` and `b` are of different logical states, including the X and Z states, and FALSE (0) otherwise.

4. SDF in the design process

4.1 Sharing of timing data

By accessing an SDF file, Electronic Design Automation (EDA) tools are assured of consistent, accurate, and up-to-date data. This means that EDA tools can use data created by other tools as input to their own processes. Sharing data in this way, layout tools can use design constraints identified during timing analysis, and simulation tools can use the post-layout delay data.

The EDA tools create, read from (to update their design), and write to SDF files.

4.2 Using multiple SDF files in one design

SDF files support hierarchical timing annotation. A design hierarchy might include several different ASICs (and/or cells or blocks within ASICs), each with its own SDF file (see Figure 1).

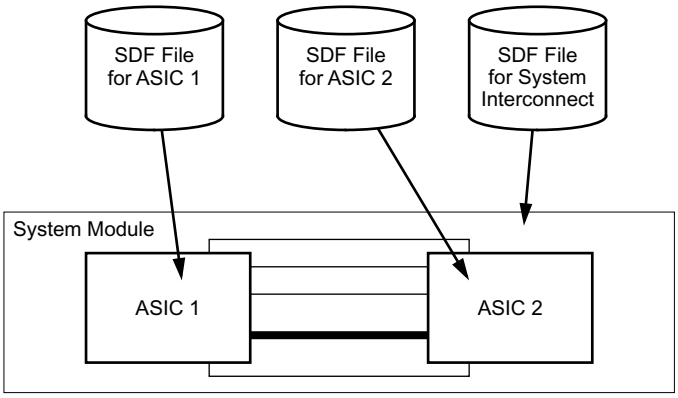

Figure 1—Multiple SDF files in a hierarchical design

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

4.3 Timing data and constraints

SDF contains constructs for the description of computed timing data for back-annotation and the specification of timing constraints for forward-annotation. There is no restriction on using both sets of constructs in the same file. Indeed, some design synthesis tools (such as floorplanners) may need access to computed timing data as well as the timing constraints intended to be meet.

Subclauses 4.5 and 4.6 discuss the use of SDF for backward- and forward-annotation of timing information.

4.4 Timing environments

SDF includes constructs for describing the intended timing environment in which a design operates. For example, a waveform to be applied at clock inputs and the arrival time of primary inputs can be specified using SDF.

4.5 Back-annotation of timing data for design analysis

Figure 2 shows the use of SDF in back-annotating timing data to an analysis tool. An advantage of this approach is that once an SDF file has been created for a design, all analysis and verification tools can access the same timing data, which ensures consistency. Note, however, that different tools can have different restrictions in the way in which the data in an SDF file is used. For example, static timing analysis tools may be able to take into account path delays that have a negative value, whereas dynamic timing simulation tools may have to interpret such negative delays as zero. Even though by using SDF the timing data used by each tool is the same, differences in tool capabilities can nevertheless result in small differences in analysis results.

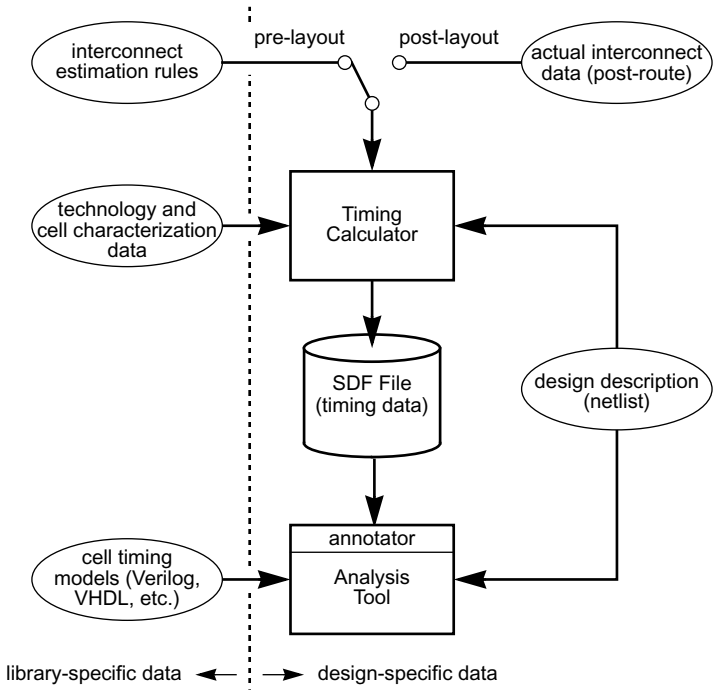

Figure 2—SDF files in timing back-annotation

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

4.5.1 The timing calculator

A timing calculator tool is responsible for generating the SDF file. To do this, the timing calculator shall examine the specific design for which it has been instructed to calculate timing data. Figure 2 shows how the timing calculator reads in the design description (netlist). The timing calculator must locate, within the design, each region for which a timing model exists and calculate values for the parameters of that timing model. Strategies for computation vary from technology to technology, but an example would be the location of each occurrence of a physical primitive from an ASIC library and the calculation of its timing properties at its boundary (pin-to-pin timing). Knowledge of the timing models can be obtained by accessing them directly (not shown) or can be built into the timing calculator and/or cell characterization data.

As the timing characteristics of ASICs are strongly influenced by interconnect effects, Figure 2 shows the timing calculator using estimation rules (pre-layout) or actual interconnect data (post-layout). Thus, SDF is suitable for both pre-layout and post-layout application.

The timing data for the design is written by the timing calculator into the SDF file. SDF imposes no restrictions on the precision to which the data is represented. Therefore, the accuracy of the data in the SDF file shall be dependent on the accuracy of the timing calculator and the information made available to it, such as pre-layout interconnect estimation methods or post-layout interconnect data extracted from the device topology.

4.5.2 The annotator

The SDF file is brought into the analysis tool through an annotator. The job of the annotator is to match data in the SDF file with the design description and the timing models. Each region in the design identified in the SDF file must be located and its timing model found. Data in the SDF file for this region shall be applied to the appropriate parameters of the timing model.

The annotator can be instructed to apply the data in the SDF file to a specific region of the design, other than at the top level of the design hierarchy. In this case, it shall search for regions identified in the SDF file starting at this point in the hierarchy. The file must clearly have been prepared with such usage in mind, otherwise the annotator will be unable to match the data found in the file with the design viewed from this point in hierarchy.

The foregoing implies that the annotator must have access to the design description and the timing models. Frequently, such access is provided via the internal representations maintained by the analysis tool. The annotator then becomes a part of the tool. As an alternative, the annotator can operate independently of the analysis tool and convert the data in the SDF file into a format suitable for the tool to read directly. If such an annotator is unable to match the SDF file to the design description and the timing models, then the effect of inconsistencies can be unpredictable. Also, certain constructs of SDF cannot be supported without access to the design description (for example, wildcard cell instance specifications).

Definition of all timing relationships, including delays and timing checks shall reside with the timing model. SDF annotation shall not be used to specify timing relationships, but only to communicate timing values.

4.5.3 Consistency between SDF file and design description

An SDF file contains timing data for a specific design. The contents of the file identifies regions of the design and provides timing data that applies to the timing properties of that region. The analysis tool or annotator cannot operate if the regions identified in the SDF file do not correspond exactly with the design description. Therefore, changes to the design generally require the timing calculator to be rerun and a new SDF file to be written.

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

Of equal importance to the logic of the design is the naming of design objects. Even if the same cells are present and are connected in the same way, annotation cannot succeed if the names by which these cells and nets are known differ in the SDF file and design description. The naming of objects must be consistent in these two places.

During annotation, inconsistencies between the SDF file and the design description shall be considered errors.

4.5.4 Consistency between SDF file and timing models

An SDF file contains only timing data. It does not contain instructions to the analysis tool concerning how to model the timing properties of the design. The SDF keywords and constructs that surround the data in the file describe the timing relationships between elements in the design only so that the data can be identified by the annotator and applied to the timing model in the correct way. It is assumed that the timing models used by the design are described to the analysis tool by some means other than the SDF file. Thus, when using SDF, it is crucial that the data in the SDF file be consistent with the timing models.

For example, if the SDF file identifies an occurrence of a 2-input NAND gate ASIC library cell in the design and states that the input-output path delay from the A input to the Y output is 0.34ns, then it is imperative that the timing model for this cell has an input port A, an output port Y and that the cell's delays are described in terms of pin-to-pin delays (as opposed to distributed delays or a single all-inputs-to-the-output delay).

Some analysis tools and the corresponding annotators can extend the timing models in certain ways. Specifically, an interconnect timing model is often not explicitly stated in the cell timing models or in the design description. The tool and/or annotator cooperate to add this information when the design and timing are loaded or merged in the tool. In this case, the SDF file shall contain data that has no obvious placeholders in the models. Nevertheless, the data must be consistent with the capabilities of the tool to model circuit timing using that data. For example, if interconnect timing is described in the SDF file in a point-to-point fashion, but the analysis tool can only represent interconnect timing as delay at cell inputs, then the tool can reject this data or perform a mapping to input delays, possibly losing information in the process.

During annotation, inconsistencies between the SDF file and the timing models are considered errors.

4.6 Forward-annotation of timing constraints for design synthesis

In addition to the back-annotation of timing data for analysis, SDF supports the forward-annotation of timing constraints to logic synthesis, floorplanner, and layout and routing tools. Timing constraints are the requirements imposed on the overall timing properties of the design, often modified and broken down by previous steps in the design process. Figure 3 shows a typical scenario of SDF in a design synthesis environment.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-3:2004

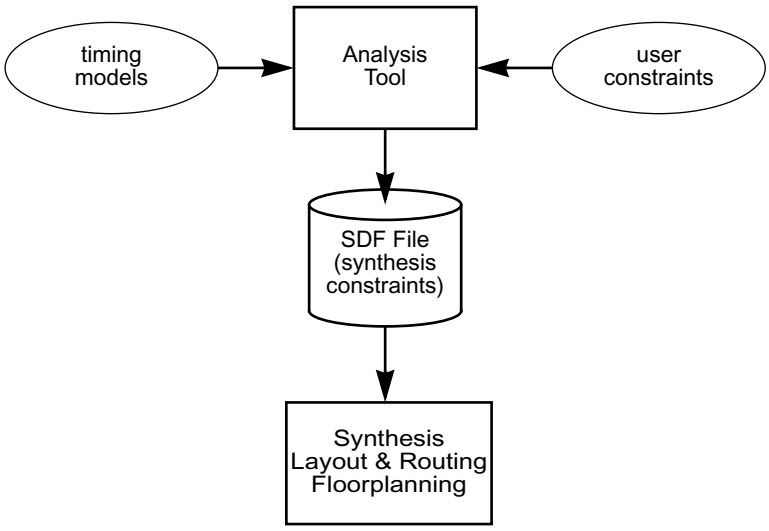

Figure 3—SDF files in constraint forward-annotation

For example, the initial requirement might be that the primary clock should run at 50 MHz. A static timing analysis of the design might identify the critical paths and the available "slack" time on these paths and pass constraints for these paths to the floorplanner, and layout and routing (physical synthesis) tools so that the final design is not degraded beyond the requirement. Alternatively, if after layout and routing, the requirement cannot be met, constraints for the problem paths might be constructed and passed back to a logic synthesis tool so that it can "try again" and leave more slack for physical synthesis.

Constraints can also be originated by an analysis tool alone. Consider a synchronous system in which the clock distribution system is to be synthesized. A static timing analysis may be able to determine the maximum permissible skew over the distribution network and provide this as a constraint to clock synthesis. In turn, this tool may break down the skew constraint into individual path constraints and forward this to physical synthesis.

NOTE—The term *timing constraint* is also in use to describe what in SDF are called timing checks. When viewed as statements of the form "this condition must be met or the circuit won't work," they are indeed the same. Perhaps the only distinction is that timing checks are applied to an analysis tool, which is only in a position to check to see if they are met and indicate a violation if they are not, whereas constraints are applied to a synthesis tool, which may adapt its operation to ensure that the specified condition is met.

In this standard, *timing check* implies a test that an analysis tool performs to make sure that a circuit, as presently constructed, shall operate reliably. The terms *timing constraint* or *constraint* imply a restriction on the timing properties of a design specified to a tool that constructs or modifies some aspect of the design (e.g., logic, layout, or routing).

4.7 Timing models supported by SDF

The importance of the consistency of an SDF file with the timing models has been stressed in 4.5.4. SDF provides a variety of ways to describe the timing of a circuit, allowing considerable flexibility in the design of the timing models. Subclauses 4.7.1 through 4.7.5 describe some modeling methodologies supported by SDF and establishes a consistent terminology used later in describing SDF itself.

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

4.7.1 Modeling circuit delay

SDF supports both a pin-to-pin and a distributed delay modeling style.

A pin-to-pin modeling style is generally one in which each physical cell in an ASIC library has its timing properties described at its boundary, i.e., with direct reference only to the ports of the cell. The timing model is frequently distinct from the functional part of the model and has the appearance of a "shell," intercepting transitions entering and leaving the functional model and applying appropriate delays to output transitions.

— The **IOPATH** construct shall apply delay data to input-to-output path delays across cells described in this way.

— The **COND** construct shall allows any path delay to be made conditional, that is, the delay value applies only when the specified condition is true. This allows for state-dependency of path delays where the path appears more than once in the timing model with conditions to identify the applicable circuit state.

— A distributed modeling style is generally one in which the timing properties of the cell are embedded in the description of the cell as a network of modeling primitives. The primitives provided by analysis tools such as simulators and timing analyzers usually have simple timing capabilities built into them, such as the ability to delay an output signal transition. The delay properties of the cell are constructed by the careful arrangement of modeling primitives and their delays. The **DEVICE** construct shall apply delay data to modeling primitives in distributed delay models.

4.7.2 Modeling output pulse propagation

SDF supports the specification of how short pulses propagate to the output of a cell described using a pin-to-pin delay model. A limit can be established for the shortest pulse that shall affect the output and a larger limit can be established for the shortest pulse that shall appear with its true logical value, rather than appearing as a "glitch" to the unknown state.

— The **PATHPULSE** construct shall allow these limits to be specified as time values.

— The **PATHPULSEPERCENT** construct shall allow these limits to be specified as percentages of the path delay.

4.7.3 Modeling timing checks

SDF timing check constructs permit values for the following categories of timing checks to be specified:

— setup
— hold
— recovery
— removal
— maximum skew
— minimum pulse width
— minimum period
— no-change

Library models can specify timing checks with respect to both external ports and internal signals. Conditions for ports and signals of timing checks can be specified using the **COND** construct. Negative values are permitted on timing checks where this is meaningful, although analysis tools that cannot use negative values can substitute a value of zero.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-3:2004

4.7.4 Modeling interconnect delays

SDF supports three styles of interconnect delay modeling.

— The **INTERCONNECT** construct shall allow interconnect delays to be specified on a point-to-point source/load basis. This is the most specific method for specifying interconnect delays. It provides the greatest power and flexibility in defining unique source/load delays.
— The **PORT** construct shall allow interconnect delays to be specified to load ports without regard to which source the signal arrives from. This is equivalent to the **INTERCONNECT** construct for wires/nets that have only one driver or source. However, for nets with more than one driver it is not be possible to represent the unique delay from each source.
— The **NETDELAY** construct shall allow the delays to all the load ports of a net to be given the same interconnect delay value. This is the least specific method for specifying interconnect delay, as it is not possible to specify either sources or loads.

4.7.5 Using internal nodes

SDF allows ports to be specified which are neither external connections of an ASIC library physical primitive nor connections between levels in the design hierarchy. Such *internal nodes* may have no corresponding terminal or net in the physical design but may instead be artifacts of the way the timing and/or functional model is constructed. For specific tools, the use of internal nodes can increase the flexibility and accuracy of the models. However, because the annotator must be able to match data in the SDF file to the models, SDF files referencing internal nodes cannot be portable to tools that do not share the same concept of internal nodes or have models constructed with or without different internal nodes.

5. Defining the standard delay format

Clause 5 describes the content of an SDF file. For each part of the file, the purpose is discussed, the syntax is specified, the semantics are explained, and examples are presented. A complete, formal definition of the file syntax is contained in Annex A.

5.1 SDF file content

SDF files shall be ASCII text files. Every SDF file shall contain a header section followed by one or more cells. The syntax is given in Syntax 1.

```
delay_file ::=
        ( DELAYFILE sdf_header cell { cell } )
```

Syntax 1: Syntax for delay format

The header section, *sdf_header*, shall contain information relevant to the entire file such as the design name, tool used to generate the SDF file, parameters used to identify the design and operating conditions (see 5.2).

Each cell, *cell*, shall identify part of the design (a "region" or "scope") and contain data for delays, timing checks, constraints and the timing environment (see 5.3). A *cell* can be a physical primitive from the ASIC library, a modeling primitive for a specific analysis tool or some user-created part of the design hierarchy. In fact, a *cell* can encompass the entire design.

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

Example (1):

```
(DELAYFILE                  // Header Section
   (SDFVERSION  "4.0")
   (DESIGN      "BIGCHIP")
   (DATE        "March 12, 1995 09:46")
   (VENDOR      "Southwestern ASIC")
   (PROGRAM     "Fast program")
   (VERSION     "1.2a")
   (DIVIDER     /)
   (VOLTAGE     5.5:5.0:4.5)
   (PROCESS     "best:nom:worst")
   (TEMPERATURE -40:25:125)
   (TIMESCALE   100 ps)


   (CELL                    // Cell 1
      (CELLTYPE "BIGCHIP")
      (INSTANCE top)
      (DELAY
         (ABSOLUTE
            (INTERCONNECT mck b/c/clk (.6:.7:.9))
            (INTERCONNECT d[0] b/c/d (.4:.5:.6))
         )
      )
   )

   (CELL                    // Cell 2
      (CELLTYPE "AND2")
      (INSTANCE top/b/d)
      (DELAY
         (ABSOLUTE
            (IOPATH a y (1.5:2.5:3.4) (2.5:3.6:4.7))
            (IOPATH b y (1.4:2.3:3.2) (2.3:3.4:4.3))
         )
      )
   )

   (CELL                    // Cell 3
      (CELLTYPE "DFF")
      (INSTANCE top/b/c)
      (DELAY
         (ABSOLUTE
            (IOPATH (posedge clk) q (2:3:4) (5:6:7))
            (PORT clr (2:3:4) (5:6:7))
         )
      )
      (TIMINGCHECK
         (SETUPHOLD d (posedge clk) (3:4:5) (-1:-1:-1))
         (WIDTH clk (4.4:7.5:11.3))
      )
   )
   (CELL                    // Cell4.. Cell n
   . . .
   )
)
```


Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

5.2 Header section

The header section of an SDF file shall contain information that relates to the file as a whole. Except for the SDF version, other items in the header sections shall be optional. The header section syntax defines a strict order for header items and those that are present shall follow this order. The items in the header section are also referred to as *entries* or *fields*.

Most entries are for documentation purposes and do not affect the meaning of the data in the rest of the file. However, the SDF version, hierarchy dividers and time scales shall, if present, change the way in which the file is interpreted. The SDF header syntax is given in Syntax 2.

sdf_header ::=
    sdf_version [*design_name*] [*date*] [*vendor*] [*program_name*] [*program_version*]
        [*hierarchy_divider*] [*voltage*] [*process*] [*temperature*] [*time_scale*]

Syntax 2: Syntax for the SDF header section

5.2.1 SDF version

The SDF version shall identify the version of the Standard Delay Format specification to which the file conforms. Syntax 3 gives the syntax for SDF version.

sdf_version ::=
    (SDFVERSION *qstring* **)**

Syntax 3: Syntax for the SDF version

qstring shall be a character string, in double quotes. The first substring within *qstring* that matches one of the strings “1.0,” “2.0,” “2.1,” “3.0,” or “4.0” shall identify the SDF version. A matching substring shall be present. Other characters before and after this substring shall be permitted and shall be ignored by readers when determining the SDF version.

Example (2):

```
(SDFVERSION "IEEE 1497 4.0")
```

Version strings “1.0,” “2.0,” “2.1,” and “3.0” are versions of the OVI SDF standard. This standard is backward compatible with the predecessor OVI version 3.0. The **SDFVERSION** statement is required.

5.2.2 Design name

The design name shall be an optional field that specifies the name of the design to which the timing data in the file shall apply. It shall be for documentation purposes and shall not affect the meaning of the data in the file. The design name has the form given in Syntax 4.

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

design_name ::=
(DESIGN *qstring* **)**

Syntax 4: Syntax for design name

qstring shall be a name that identifies the design.

5.2.3 Date

The date shall be an optional field that specifies the currency of the data in the file. It shall be for documentation purposes and shall not affect the meaning of the data in the file. The syntax for date is given in Syntax 5.

date ::=
(DATE *qstring* **)**

Syntax 5: Syntax for the date

qstring can be any legal string, but it is intended to be a character string that represents the date and/or time when the data in the SDF file was generated.

Example (3):

```
(DATE "Friday, September 17, 1993 - 7:30 p.m.")
```

5.2.4 Vendor

The vendor field shall be an optional field that specifies the name of the company manufacturing the device to which the data in the file applies or who originated the program that created the file. It shall be for documentation purposes and shall not affect the meaning of the data in the file. The syntax for vendor field is described in Syntax 6.

vendor ::=
(VENDOR *qstring* **)**

Syntax 6: Syntax for the vendor

qstring can be any legal string, but it is intended to be a character string containing the name of the vendor whose tools generated the SDF file.

Example (4):

```
(VENDOR "Acme Semiconductor")
```

5.2.5 Program name

The program name shall be an optional field that specifies the name of the program that created the file. It shall be for documentation purposes and shall not affect the meaning of the data in the file. The program name has the syntax shown in Syntax 7.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-3:2004

program_name ::=
(PROGRAM *qstring***)**

Syntax 7: Syntax for program name

qstring can be any legal string, but it is intended to be a character string containing the name of the program used to generate the SDF file.

Example (5):

```
(PROGRAM "timcalc")
```

5.2.6 Program version

The program version shall be an optional field that specifies the version of the program that created the file. It shall be for documentation purposes and shall not affect the meaning of the data in the file. The syntax for program version is given in Syntax 8.

program_version ::=
(VERSION *qstring* **)**

Syntax 8: Syntax for program version

qstring can be any legal string, but it is intended to be a character string containing the program version number used to generate the SDF file.

Example (6):

```
(VERSION "version 1.3")
```

5.2.7 Hierarchy divider

The hierarchy divider shall be an optional field that specifies which of the two permissible characters shall be used in the file to separate elements of a hierarchical path. The hierarchy divider has the form given in Syntax 9.

hierarchy_divider ::=
(DIVIDER *hchar* **)**

Syntax 9: Syntax for hierarchy divider

hchar shall be either a period ('.'), or a slash ('/'). It shall not be in quotes.

Example (7):

```
(DIVIDER /)
. . .
    (INSTANCE a/b/c)
    . . .
```


Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

In Example (7), the hierarchy divider is specified to be the slash ('/') character and hierarchical paths use slash rather than period ('.') to separate elements.

If the SDF file does not contain a hierarchy divider then the default hierarchy divider shall be the period ('.'). See also the descriptions of *identifier* and *hierarchical_identifier* in A.1.

5.2.8 Voltage

The voltage shall be an optional field that specifies the operating voltage or voltages for which the data in the file was computed. It shall be for documentation purposes and shall not affect the meaning of the data in the SDF file. The syntax for voltage is given in Syntax 10.

voltage ::=
    (VOLTAGE *rtriple* **)**
  | **(VOLTAGE** *signed_real_number***)**

Syntax 10: Syntax for the voltage

rtriple or *signed_real_number* shall indicate the operating voltage (in volts) at which the design timing was calculated or the constraints are to apply.

Example (8):

```
(VOLTAGE 5.5:5.0:4.5)
```

Although this information cannot used by the annotator, it shall be borne in mind that the order of delay and timing check limit values in *triple*s is minimum:typical:maximum. Since minimum delays usually occur at the highest supply voltage, it is more consistent with the use of *triple*s elsewhere in the file if the highest voltage is specified first in the voltage and the lowest voltage is specified last.

5.2.9 Process

The process shall be an optional field that specifies the process factor for which the data in the file was computed. It shall be for documentation purposes and shall not affect the meaning of the data in the file. The syntax for process is described in Syntax 11.

process ::=
    (PROCESS *qstring* **)**

Syntax 11: Syntax for process

qstring shall be a character string which specifies the process operating envelope.

Example (9):

```
(PROCESS "best=0.65:nom=1.0:worst=1.8")
```


Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

5.2.10 Temperature

The temperature shall be an optional field that specifies the operating temperature for which the data in the file was computed. It shall be for documentation purposes and shall not affect the meaning of the data in the file. The syntax for temperature is given in Syntax 12.

temperature ::=
    (TEMPERATURE *rtriple* **)**
  | **(TEMPERATURE** *signed_real_number* **)**

Syntax 12: Syntax for temperature

rtriple or *signed_real_number* shall indicate the operating ambient temperature(s) of the design in degrees Celsius (centigrade).

Example (10):

```
(TEMPERATURE -25.0:25.0:85.0)
```

5.2.11 Timescale

The timescale shall be an optional field that specifies the units used for all time values in the SDF file. The syntax for time scale is shown in Syntax 13.

time_scale ::=
    (TIMESCALE *timescale_number timescale_unit* **)**
timescale_number ::= **1** | **10** | **100** | **1.0** | **10.0** | **100.0**
timescale_unit ::= **s** | **ms** | **us** | **ns** | **ps** | **fs**

Syntax 13: Syntax for the timescale

TIMESCALE accepts a number followed by a unit. The number can be 1, 10, 100, 1.0, 10.0, or 100.0. The unit can be s, ms, us, ns, ps, or fs, representing seconds, milliseconds, microseconds, nanoseconds, picoseconds, and femtoseconds, respectively. One or more space characters can optionally separate the number and the unit.

Example (11):

```
(TIMESCALE 100 ps)
. . .
    (IOPATH (posedge clk) q (2:3:4) (5:6:7))
    . . .
```

Example (11) indicates that all time values in the file are to be multiplied by 100 picoseconds. Thus, the values supplied in the **IOPATH** are (0.2ns:0.3ns:0.4ns) and (0.5ns:0.6ns:0.7ns).

If the SDF file does not contain a timescale then all time values in the file shall be assumed to be in nanoseconds. This has the same effect as a timescale of 1ns.

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

5.3 Cells

A cell shall identify a particular “region” or “scope” within a design and shall contain timing data to be applied there. For example, a cell may identify a unique occurrence of an ASIC physical primitive, such as a 2-input NAND gate, in the design and provide values for its timing properties, such as the input-to-output path delays. The cells can also identify all occurrences of a particular ASIC library physical primitive, such as a certain type of gate or flip-flop. Data shall be applied to all such library-specific regions in the design. The syntax for a cell is given in Syntax 14.

cell ::=
(CELL *celltype cell_instance* { *timing_spec* } **)**

Syntax 14: Syntax for cell

The *celltype* and *cell_instance* fields shall identify a region of the design. The *timing_spec* field shall contain the timing data.

Example (12):

```
(CELL
   (CELLTYPE “DFF”)
   (INSTANCE a/b/c)
   (DELAY
      (ABSOLUTE
         (IOPATH (posedge clk) q (2:3:4) (5:6:7) )
      )
   )
)
```

An SDF file shall contain any number of cells (other than zero). The order of the cells shall be significant only if they have overlapping effect, in other words, if data from two different cells applies to the same timing property in the design. In this situation, the cells shall be processed strictly from the beginning of the file towards the end and the data they contain shall be applied in sequence to whatever region is appropriate to that cell. If data is applied to a timing property previously referenced by an SDF file, the new data shall be applied over the old and the final value shall be the cumulative effect according to whether the data is applied as a replacement for the old value (**ABSOLUTE** and **TIMINGCHECK** sections) or is added to it (**INCREMENTAL** section).

5.3.1 Cell type

The **CELLTYPE** shall indicate the name of the cell. The syntax for cell type is described in Syntax 15.

celltype ::=
(CELLTYPE *qstring* **)**

Syntax 15: Syntax for cell type

qstring shall be a character string. If the region of the design identified by this cell is an occurrence of a physical primitive from an ASIC library, then *qstring* shall be the name by which the cell is known in the library.

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

Example (13):

```
(CELLTYPE "DFF")
```

In Example (13), the cell identifies an occurrence of a cell which has the name “DFF” (perhaps a D-type flip-flop).

If the cell identifies a region of the design which is a user-created level in the hierarchy, or, for example, the very top level, then *qstring* shall be the user-created name for that part of the design.

Example (14):

```
(CELLTYPE "TOP")
```

In Example (14), the cell identifies a user-created design block which the user has named “TOP”.

If the cell identifies a modeling primitive, in other words something that is not part of the physical design but is part of the implementation of a model in a particular analysis tool, then *qstring* shall be the name by which the modeling primitive is known in that tool.

Example (15):

```
(CELLTYPE "buf")
```

In Example (15), the cell identifies a “buf” modeling primitive in an analysis tool, perhaps a buf “gate” in a Verilog model.

5.3.2 Cell instance

The cell instance shall identify the region or scope of the design for which the cell contains timing data. The name by which this region is known in the design shall be consistent with the **CELLTYPE** for the cell. If the annotator locates the region and finds that its name does not match the **CELLTYPE**, it shall indicate an error. The cell instance has the form given in Syntax 16.

cell_instance ::=
  (INSTANCE [*hierarchical_identifier*] **)**
| **(INSTANCE *)**

Syntax 16: Syntax for cell instance

The first form of the cell instance identifies an unique occurrence in the design of the region named in the cell type. If, for example, the cell is a physical primitive from an ASIC library, then a single occurrence of that cell on the chip shall be identified. To do this, the cell instance provides a complete path through the design hierarchy to the cell or region of interest.

The hierarchical path shall start at the level in the design at which the annotator shall be instructed to apply the SDF file. Frequently, this is the topmost level. The path is extended down through the hierarchy by adding further levels to *hierarchical_identifier*.

Documents
Document Sharing Hub
Full document details: BS IEC 61523-3:2004

Example (16):

```
(CELL
   (CELLTYPE "DFF")
   (INSTANCE a1.b1.c1)
    . . .
)
```

In Example (16), the complete hierarchical path is specified as `a1.b1.c1` following the **INSTANCE** keyword. The region identified is cell/block `c1` within block `b1`, which is in turn within block `a1`. The SDF file must be applied at the level containing `a1`. The period character separates levels or elements of the path. The example assumes that the hierarchy divider in the SDF header specified the hierarchy divider as the period character or, since period is the default, the entry was absent.

The timing data in the timing specifications of this cell shall apply only to the identified region of the design. If *hierarchical_identifier* is not specified, the default shall be the region (hierarchical level) in the design at which the annotator is instructed to apply the SDF file (see 4.5.2). This can be useful for gathering all interconnect information into a top-level cell.

The second form of the cell instance can be used to associate timing data with all occurrences of the specified cell type. Instead of a hierarchical path, the wildcard character ('*') after the **INSTANCE** keyword shall be specified.

Cells using the wildcard cell instance specification are processed in sequence just like any other cells. No special action shall be taken to consolidate data in this cell with cells with the same cell type earlier or later in the file.

Example (17):

```
(CELL
   (CELLTYPE "DFF")
   (INSTANCE *)
    . . .
)
```

In Example (17), every DFF cell instances shall receive the timing data. Note, however, that only cells contained within the region to which the annotator is instructed to apply the SDF file shall be affected.

5.3.3 Timing specifications

Each cell in the SDF file shall include zero or more timing specifications that contain the actual timing data associated with that cell. There are four types of timing specifications that are identified by the **DELAY**, **TIMINGCHECK**, **TIMINGENV**, and **LABEL** keywords. Syntax 17 describes the syntax for timing specification.

Documents
Document Sharing Hub

Full document details: BS IEC 61523-3:2004