

Supplement to IEEE Std 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture

Sponsor
**Test Technology Standards Committee
of the
IEEE Computer Society**

Approved September 22, 1994
IEEE Standards Board

Abstract: A language to describe components that conform to IEEE Std 1149.1-1990 is described in this supplement. The language is based on the VHSIC Hardware Description Language (VHDL). General characteristics, the overall structure of a Boundary-Scan Description Language (BSDL) description, special cases, and example packages are included.

Keywords: boundary scan, boundary scan architecture, boundary-scan description language, BSDL, test access port, VHDL, VHSIC Hardware Description Language

The Institute of Electrical and Electronics Engineers, Inc.

345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1995 by the Institute of Electrical and Electronics Engineers, Inc.

All rights reserved. Published 1995. Printed in the United States of America.

ISBN 1-55937-497-7

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

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

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. 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.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

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 all 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 technical 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 on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

IEEE Standards documents may involve the use of patented technology. Their approval by the Institute of Electrical and Electronics Engineers does not mean that using such technology for the purpose of conforming to such standards is authorized by the patent owner. It is the obligation of the user of such technology to obtain all necessary permissions.

Introduction

(This introduction is not a part of IEEE Std 1149.1b-1994, Supplement to IEEE Std 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture.)

This document is a supplement to IEEE Std 1149.1-1990 (includes IEEE Std 1149.1a-1993), IEEE Standard Test Access Port and Boundary-Scan Architecture. The document contains a proposal for a language to describe components that conform to this standard and is published as an additional annex to this standard.

This supplement was developed from a proposal made by Kenneth P. Parker and Stig Oresjo in 1990. Between that time and the date of approval of this supplement, several individuals contributed to the development of Boundary-Scan Description Language (BSDL). The working group would like to acknowledge James Beausang, Bill DenBeste, Dick Chiles, Paul Ocampo, Larry Saunders, Don McLean, Adam Sheppard, Alain Plassart, and Rene Segers for contributing to the development of the first draft that was presented to the working group.

The following persons participated in the working group that developed and prepared this supplement.

Najmi Jarwala, *Chair*
Adam W. Ley, *(Vice Chair North America)*
Math Muris, *(Vice Chair Europe)*
Colin Maunder, *(Chair Emeritus)*
Kenneth P. Parker, *(Editor)*
Stig Oresjo, *(Editor)*

John Andrews
Dilip K. Bhavsar
Harry Bleeker
Terry Borroz
Berhard Geisberger
Grady L. Giles

Peter L. Hansen
Patrick F. McHugh
Dieter Ohnesorge
Michel Parot
Gordon D. Robinson

Markus Robinson
Robert J. Russell
Rodham E. Tulloss
Dirk van de Lagemaat
Lee Whetsel
T. W. Williams

The following persons were on the balloting committee:

Magdy S. Abadir
M. B. E. Abdelrazik
Robert Aitken
Charles Albert
Anthony Ambler
John Andrews
Chid Annamalai
Chris Bagge
Stefano Barbagallo
Paul Bardell
Jim Beals
James Beausang
R. Bennetts
Dilip K. Bhavsar
Sol Black
Harry Bleeker
Federico Bonzanigo
Terry Borroz
Melvin Breuer
Alan Bridges
Bill Bruce

Michael L. Bushnell
Hamid H. Butt
Barry Caldwell
John Campbell
Gunnar Carlsson
Terence Chan
R. Chandramouli
Shir-Shen Chang
Harry Chen
Tony Cheng
Wilton Chiles
John Chin
Winston Chung
Henry Cox
John Crownover
R. Dandapani
Wayne Daniel
Scott Davidson
Stephen P. Denker
Bulent Dervisoglu
Michael Donahue

Peter Eckstein
Nick G. Eskandari
Patrick Fasang
Carlos Ferrer
Tammy Fisher
Andrew Flint
W. Kent Fuchs
Grady L. Giles
Raymond Gocal
Steven Gonet
Peter L. Hansen
Hong Hao
Kazumi Hatayama
John P. Hayes
Wei-Ching Her
Hezi Himelfarb
Randy Holmberg
Lee F. Horney
S. Hossein Hosseini
Axel Hunger
Kazuhiko Iwasaki

Neal Jaarsma
Kamran Jamal
J. Jamieson
Najmi Jarwala
Frank V. Jorgensen
Carl Kagy
Nick Kanopoulos
Jake Karrfalt
David C. Keezer
Ilyoung Kim
Christian Kimstedt
Bruce Kraemer
Gunter T. Krampf
Thomas Kropf
Gerd Kruger
Arun Kumar
Sy-Yen Kuo
Steven K. Ladd
Dirk van de Lagemaat
David Landis
Thomas Langford
Luis Larzabal
Johnny J. Leblanc
Ben Lee
J. Leenstra
Donald H. Lenhert
Regis Leveugle
Adam W. Ley
Ting-Ting Lin
G. E. Lousberg
Hede Ma
Chittaranja Mallipeddi
Kenneth Mandl
David Marquart
Colin Maunder
Peter C. Maxwell
John McCann

Patrick F. McHugh
David Medina
Earl Meiers
Zeljko Miksic
Richard Miller
Steven D. Millman
James A. Monzel
John T. Moynihan
Joachim Mucha
Math Muris
Katsunobu Muroi
Benoit Nadeau-Dostie
Arnold Nordsieck
Bill O'Grady
Jim O'Reilly
Tom O'Rourke
Paul Ocampo
Dieter Ohnesorge
Sungju Park
Kenneth P. Parker
Michel Parot
Raymond A. Paul
Michael Paulson
Bruce Peterson
Herman Plott
Dhiraj K. Pradhan
Paolo Prinetto
Jeffrey Quay
Edward Ramirez
Hira Ranga
Gordon D. Robinson
Markus Robinson
Benjamin Rogel-Favila
Robert Rolfe
Daniel Romanchik
Kaushik Roy
Kamalesh N. Ruparel
Robert J. Russell

William E. Russell, Jr.
Bennett Scharf
Dana Schow
Lahouari Sebaa
Jonjen Sern
Ashok K. Sharma
John W. Sheppard
Majid Shoraka
William R. Simpson
David Simpson
William H. Smith
Mani Soma
David Stannard
Bret A. Stewart
Tom Storey
William R. Sullivan
Xiaoling Sun
Gordon Taylor
Jacques Tazartes
Rodham E. Tulloss
Jon L. Turino
A. J. van de Goor
James Vanderwiele
Bas C. Verhelst
Spyros Villios
Ken D. Wagner
Russell J. Wagner
Michael G. Wahl
Malcolm Wallace
Thomas J. Wey
Lee Whetsel
Brian Wilkins
T. W. Williams
Wermer Wolz
Cheng-Wen Wu
Hans-Joachim Wunderlich
Weider D. Yu

When the IEEE Standards Board approved this standard on September 22, 1994, it had the following membership:

Wallace S. Read, *Chair*
Donald C. Loughry, *Vice Chair*
Andrew G. Salem, *Secretary*

Gilles A. Baril
Bruce B. Barrow
José A. Berrios de la Paz
Clyde R. Camp
James Costantino
Stephen L. Diamond
Donald C. Fleckenstein
Jay Forster*
Ramiro Garcia

Donald N. Heirman
Richard J. Holleman
Jim Isaak
Ben C. Johnson
Sonny Kasturi
Lorraine C. Kevra
E. G. “Al” Kiener
Ivor N. Knight

Joseph L. Koepfinger*
D. N. “Jim” Logothetis
L. Bruce McClung
Marco W. Migliaro
Mary Lou Padgett
Arthur K. Reilly
Ronald H. Reimer
Gary S. Robinson
Leonard L. Tripp

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
James Beall

Richard B. Engelman
Robert E. Hebner

David E. Soffrin

Mary Lynne Nielsen
IEEE Standards Project Editor

The figure on page 59 is reprinted with the permission of Texas Instruments, Inc.

CLAUSE	PAGE
Annex B Boundary-Scan Description Language (Informative).....	2
B.1 General information.....	2
B.2 Purpose of BSDL.....	3
B.3 Scope of BSDL	3
B.4 Relationship of BSDL to VHDL	4
B.5 Lexical elements of BSDL	5
B.6 Notes on syntax definition.....	9
B.7 Components of a BSDL description.....	10
B.8 The entity description	10
B.9 The Standard VHDL Package STD_1149_1_1994.....	45
B.10 User-Supplied VHDL packages	49
B.11 Example applications of BSDL.....	55
B.12 A typical application of BSDL	60
B.13 The 1990 version of BSDL	62

Supplement to IEEE Std 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture

Changes to IEEE Std 1149.1-1990 (includes IEEE Std 1149.1a-1993)

- a) Clause 2.1 shall be modified so that the note at the bottom of page 2-1 (“NOTE—The descriptive material contained in this standard is for illustrative purposes only and does not define a preferred implementation”) is replaced by the following text, which shall be a normal font paragraph in the Description section of 2.1:

CAUTION — The descriptive material contained in this standard is for illustrative purposes only and does not define a preferred implementation. Examples are provided throughout the standard to illustrate possible circuit implementations. Where discrepancies between examples and specifications may occur, the specifications always take precedence. Readers should exercise caution when using these examples to ensure full compliance in their specific application. In particular, it is emphasized that the examples are designed to effectively communicate the meaning of this standard. As such, they are logically correct; however, as always, a particular implementation may not operate properly with respect to timing and other parametric characteristics. One example of this concern is that the sample TAP Controller implementation depicted in figure 5-5 reasonably assumes a significantly greater delay in the flip-flop sourcing A and A* than in the inverter sourcing TCK*. It is possible to design a circuit where this assumption is violated, causing a critical race to occur that would invalidate the behavior of the TAP controller. Therefore, it is recommended that implementations be fully verified to ensure compliance under required operating conditions.

- b) Clause 2.4, References:
Add the following reference:
IEEE Std 1076-1993, IEEE Standard VHDL Language Reference Manual (ANSI).
- c) Chapter 3
Rule 3.3.1b: Amend to read “. . . produces a logical response . . .”
Rule 3.4.1b: Amend to read “. . . produces a logical response . . .”
Rule 3.6.1b: Amend to read “. . . produces a logical response . . .”
- d) Replace rule 7.9.1c with the following:
 - c) Where a test data register is required to be initialized prior to execution of the self-test, this shall occur at the start of the self-test without any requirement to shift data into the component (i.e., there shall be no requirement to enter seed values into any test data register).

NOTE — As per rule 7.9.1k (i), the boundary-scan register may (optionally) need to be initialized to define the state of signals driven from system output pins. However, this value should not be used as a seed for the self-test operation, since it may be board dependent.

Annex B Boundary-Scan Description Language

(Informative)

This annex defines a machine-readable language that allows rigorous description of testability features in components that comply with this standard. The language is called the Boundary-Scan Description Language (BSDL). It is a subset and standard practice of the VHSIC Hardware Description Language (VHDL) (IEEE Std 1076-1993).

B.1 General information

B.1.1 Document outline

Clauses B.2 to B.4 define the purpose of BSDL, its scope, and its relationship to VHDL. Clauses B.5 to B.7 describe general characteristics of the language. Clause B.8, the Entity description, describes the overall structure of a BSDL description. B.8.2 to B.8.18 are detailed descriptions of each of the mandatory and optional sections of a BSDL description. They are documented in the order they should appear in such a description. Clause B.9 describes the standard VHDL package and clause B.10 describes a user-supplied VHDL package. Clause B.11 includes some special cases for purposes of illustration and indicates how such components can be specified in BSDL. Clause B.12 shows a typical BSDL description. Clause B.13 briefly documents the precursor language to BSDL (see also B.1.3).

In B.8.2 to B.8.18, the detailed descriptions of each of the mandatory and optional sections of a BSDL description are organized in the following way:

- Short introduction
- Syntax
- Notes
- Example and additional text
- Semantic checks

The syntax and semantic subclauses are where the language is defined; however, sometimes material in short introductions, notes, and examples is key to full understanding of the language. Therefore, such text is also part of this description. The notes often provide the semantic interpretation of various lexical elements.

Commonly used syntactic elements are defined in B.6.3.

B.1.2 Conventions

- a) Examples are printed in Courier font.
- b) See B.6 for conventions relating to syntax.
- c) For clarity, all reserved words, predefined words, and punctuations are shown in **bold Helvetica** type within this document. VHDL reserved and predefined words will be shown in **lowercase** letters, and BSDL reserved words will be shown in **UPPERCASE** letters. BSDL itself is case-insensitive; this convention is adopted for clarity.)

B.1.3 BSDL history

The development of BSDL started soon after the first promulgation of this standard. When, out of common self-interest, an industry-wide group of companies implementing tools to support this standard realized that a single language for describing the boundary-scan implementations in components would be of benefit, many tools were built using early draft specifications of the BSDL language.

Developments made both to the standard and to the BSDL definition since the language was first proposed in 1990 have resulted in the obsolescence of some of the constructs from the first draft versions of the language. Some constructs have been rendered unnecessary as a result of the standardization of the *CLAMP* and *HIGHZ* instructions in 1993, while others were found to duplicate information provided elsewhere in a BSDL description and were thus removed as redundant.

Because a significant number of BSDL descriptions have been written based on the draft 1990 version of the language, and because these descriptions are likely to remain in circulation for some time, implementors of tools based on BSDL may wish to design them to read both the BSDL language defined in this annex and the earlier version. Information on how to do this is contained in clause B.13.

B.2 Purpose of BSDL

BSDL provides a machine-readable means of representing some parts of the documentary information specified in 12.3. The scope of the language is defined in B.3.

The goal of the language is to facilitate communication between companies, individuals, and tools that need to exchange information on the design of test logic that complies with this standard, for example

- A vendor of a component that supports this standard may supply a BSDL description to purchasers.
- Automated test-generation tools may use a library of BSDL descriptions to allow generation of a test for a particular loaded board.
- The test logic defined by this standard could be synthesized with the support of a BSDL description.

BSDL describes “finished” silicon, not “work-in-progress.” For example, when a bare die conforming to this standard is produced, a BSDL description can be provided for it. A die may be inserted into one or more types of component packages as well, with each variation described in BSDL (see B.8.7). BSDL for partially synthesized test logic is considered “work-in-progress,” is not necessarily compliant with this annex and, in most cases, should not be transmitted beyond the synthesis environment.

B.3 Scope of BSDL

BSDL is not a general-purpose hardware description language—it is intended solely as a means of describing key aspects of the implementation of this standard within a particular component. A BSDL description is not itself a simulation model. Examples of features that are and are not described using BSDL are listed in table B-1.

Note that the language describes only features of the test logic that can vary from component to component, depending on the choice of the component designer. Features that are completely specified by this standard (without option) are not required to be described in BSDL, but may be described if users so wish.

Further, BSDL does not have a general means for providing for the specification of logic levels, timing parameters, power requirements, and other similar factors. These data do not affect the logical behavior of an implementation and are most likely already described in other parts of the specification of any given component.

Table B-1—Scope of BSDL

Features described by BSDL	Features that cannot or need not be described
Length and structure of the boundary-scan register	Tap controller state diagram
Availability of the optional TRST* pin	Bypass register
Physical locations of the TAP pins	Length of the device-identification register
Instruction codes	Provision of <i>SAMPLE/PRELOAD</i> , <i>BYPASS</i> , and <i>EXTEST</i> instructions
Device-identification code	Operation of user-defined instructions

B.4 Relationship of BSDL to VHDL

BSDL is a subset of VHDL (IEEE Std 1076-1993) and is fully conformant to the requirements of the VHDL standard. However, the user must be prepared to modify a BSDL description in the face of implementation dependencies in VHDL-based tools. No way has been found of avoiding this small amount of effort without introducing further undesirable complications. Specifically, the <standard use statement> (see B.8.4) and the <use statement> (see B.8.5) may require editing because of tool and file system dependencies. Syntax of the statements as defined are legal VHDL; however, an additional prefix (identifying a library in which the Standard VHDL Package will be found) will need to be added for some applications. A syntax lacking such a prefix has been chosen to force an error in such an application rather than risk unpredictable and confusing errors due to inclusion of an inappropriate prefix.

NOTE — In the event of an error or omission in this annex regarding VHDL syntax, IEEE Std 1076-1993 will take precedence.

Let it be noted that

- a) BSDL does not employ all the syntactic elements of VHDL. Only those elements required to meet the scope of BSDL are used. The following VHDL statements are employed in BSDL:

attribute	package body
constant	port
end	subtype
entity	type
generic	use
package	

In some cases, only a subset of a particular VHDL language element syntax is used in BSDL. Descriptions of the lexical elements and statement syntax for each syntax element are contained in B.5–B.8 and B.10. Historical information on such elements is found in B.13.

- b) For cases in which a feature could be described in several ways within VHDL, a restricted set of ways has been selected and defined explicitly as the standard practice for BSDL. This restriction simplifies the application of the VHDL subset for BSDL, particularly for tools that are only required to read or generate BSDL (i.e., tools that have no requirement to read or write the full VHDL language).
- c) BSDL imposes additional requirements on the syntax and content of certain character strings—that is, sequences of characters between quotation marks (e.g., “**EXTEST**”). A VHDL parser will not check the information in these strings. In contrast, a BSDL parser will check that the information in the strings is appropriate for the relevant parameters or attributes for which such strings might be values. There are several string syntaxes defined within BSDL.

B.5 Lexical elements of BSDL

The lexical elements of BSDL are a subset and standard practice of those of VHDL as defined in IEEE Std 1076-1993. The following subclauses enumerate the lexical elements needed to understand the BSDL language definition.

B.5.1 Character set

The language is not case sensitive-the character **a** is considered identical to the character **A**. Therefore, the following names are identical:

FRED Fred fred

The following characters are permitted within the language:

- a) Upper- and lowercase letters: **A** to **Z** and **a** to **z** (the language is not case sensitive)
- b) Digits: **0** to **9**
- c) Special characters: " & ' () * , - . : ; < = > _ (Note that this list is smaller than that of VHDL.) Other special characters are allowed as part of a comment (see B.5.3) or as part of the string element of a design warning (see B.8.18).
- d) Logical separators: The space character and VHDL format effectors are used as logical separators. VHDL format effectors are the characters called horizontal tabulation, vertical tabulation, carriage return, line feed, and form feed.

Logical separators have two purposes. They are used to eliminate lexical ambiguity by separating lexical tokens such as reserved words and/or identifiers. For example, the reserved word **entity** must be separated from the component name identifier that immediately follows it, rather than being run together with it. Logical separators are also used in combinations to create visually appealing layouts.

B.5.1.1 Identifiers

Identifiers are user-supplied names and reserved words functioning as names. Identifiers have to start with a letter and may contain letters, digits, or the underscore character. For example, the following are valid identifiers:

BSDL
IEEE_Std_1149_1

There is no upper limit to the number of characters in an identifier.

The underscore character (**_**) is not allowed to be the last character in an identifier (by VHDL).

IEEE_STD_1149_ -- This is not a legal identifier.

Adjacent underscore characters (**_ _**) are not allowed (by VHDL).

B.5.1.2 BSDL reserved words

The identifiers listed in this subclause are BSDL reserved words and have a fixed significance in the language. These identifiers cannot be used for any other purposes in a BSDL description, other than as part of a comment. For example, a reserved word cannot be used as an explicitly declared identifier. **BC_0** to **BC_99** are variable names used in the Standard VHDL Package. Names **BC_0** through **BC_7** are used today, while **BC_8** through **BC_99** are reserved for future use. Similarly, the names **STD_1149_1_1990**, **STD_1149_1_1993**, and **STD_1149_1_1994** have been reserved, with the potential for new names to be added later. Therefore, avoid using identifiers that start with **STD_1149_**.

AT_PINS	INSTRUCTION_OPCODE
BC_0 to BC_99	INSTRUCTION_PRIVATE
BIDIR	INTERNAL
BIDIR_IN	INTEST
BIDIR_OUT	INTEST_EXECUTION
BOTH	LOW
BOUNDARY	OBSERVE_ONLY
BOUNDARY_LENGTH	OBSERVING
BOUNDARY_REGISTER	ONE
BSCAN_INST	OUTPUT2
BSDI_EXTENSION	OUTPUT3
BYPASS	PHYSICAL_PIN_MAP
CAP	PI
CAP_DATA	PIN_MAP
CAPTURES	PIN_MAP_STRING
CELL_DATA	PO
CELL_INFO	PORT_GROUPING
CELL_TYPE	PULL0
CLAMP	PULL1
CLOCK	REGISTER_ACCESS
CLOCK_INFO	RUNBIST
CLOCK_LEVEL	RUNBIST_EXECUTION
COMPLIANCE_PATTERNS	SAMPLE
COMPONENT_CONFORMANCE	STD_1149_1_1990
CONTROL	STD_1149_1_1993
CONTROL	STD_1149_1_1994
DESIGN_WARNING	TAP_SCAN_CLOCK
DEVICE_ID	TAP_SCAN_IN
DIFFERENTIAL_CURRENT	TAP_SCAN_MODE
DIFFERENTIAL_VOLTAGE	TAP_SCAN_OUT

EXPECT_DATA	TAP_SCAN_RESET
EXTEST	UPD
HIGHZ	USERCODE
ID_BITS	USERCODE_REGISTER
ID_STRING	WAIT_DURATION
IDCODE	WEAKO
IDCODE_REGISTER	WEAK1
INPUT	X
INSTRUCTION_CAPTURE	Z
INSTRUCTION_LENGTH	ZERO

B.5.1.3 VHDL reserved and predefined words

The identifiers listed below are called VHDL (IEEE Std 1076–1993) reserved and predefined words and have a fixed significance in the language. These identifiers may not be used for any other purpose in a BSDL description, other than as part of a comment. For example, a reserved word cannot be used as an explicitly declared identifier. Reserved words shown in the list below in lowercase letters are part of the BSDL subset of VHDL. Those in uppercase letters are not part of BSDL, but cannot be used as identifiers. The VHDL reserved words are shown in this subclause for convenience, *but the latest edition of the VHDL standard shall be consulted as the final authority.*

ABS	GUARDED	record
ACCESS	IF	REGISTER
AFTER	IMPURE	REJECT
ALIAS	in	REM
all	INERTIAL	REPORT
AND	inout	RETURN
ARCHITECTURE	is	ROL
array	LABEL	ROR
ASSERT	LIBRARY	SELECT
attribute	linkage	SEVERITY
BEGIN	LITERAL	SHARED
bit	LOOP	signal
bit_vector	MAP	SLA
BLOCK	MOD	SLL
body	NAND	SRA
buffer	NEW	SRL
BUS	NEXT	string

CASE	NOR	subtype
COMPONENT	NOT	THEN
CONFIGURATION	NULL	to
constant	of	TRANSPORT
DISCONNECT	ON	true
downto	OPEN	type
ELSE	OR	UNAFFECTED
ELSIF	OTHERS	UNITS
end	out	UNTIL
entity	package	use
EXIT	port	VARIABLE
FALSE	positive	WAIT
FILE	POSTPONED	WHEN
FOR	PROCEDURE	WHILE
FUNCTION	PROCESS	WITH
GENERATE	PURE	XNOR
generic	range	XOR

B.5.2 Strings

A string is defined as a sequence of zero or more characters enclosed between quotation marks. A quotation mark character is not allowed within a string in BSDL. For example

```
"Mary had a little lamb"           -- Allowed
"Fred said " "HELP" " "          -- Not allowed
```

Strings are used extensively in BSDL. Since many of the BSDL strings are potentially much longer than a single line, the concatenation operator **&** is used to break them into manageable pieces. For example:

```
"Mary had a little lamb. " &
"Its fleece was white as snow."
```

is a single string, identical to

```
"Mary had a little lamb. Its fleece was white as snow."
```

BSDL does not permit replacement of the quotation mark with any other character.

A string literal has to fit on one line since it is a lexical element. Therefore, the only legal VHDL format effector in a string literal is horizontal tabulation.

B.5.3 Comments

Text between a double dash (--) symbol and the end of a line is treated as a comment. The text is allowed to contain any special character allowed by VHDL, not just those given in item c) of B.5.1. Comments syntactically terminate a line of a description. Comments may be interspersed with lexical elements. For example, the following is an example of a single VHDL string:

```
"This is all" &          -- An example of a string split by a comment
" a single string"
```

B.6 Notes on syntax definition

B.6.1 BNF conventions

The syntax of BSDL is presented in Backus-Naur Form (BNF) as follows:

- Any item enclosed in chevrons (i.e., between the character "<" and the character ">") is the name of a syntax item that will be defined in this annex.
- Items enclosed between braces (i.e., between the character "{" and the character "}") may be omitted or included one or more times.
- Items enclosed between square brackets (i.e., between the character "[" and the character "]") may be omitted or included only one time.
- Items enclosed between the characters "}" and "{" may appear in any order.
- Except with regard to case, text shown in **bold Helvetica** type has to be included exactly as it is presented in this annex.
- Alternative syntaxes are separated by a vertical bar("|").
- The symbol "::=" should be read as "is defined as." Note that the nonboldface "::=" is only part of a BNF description; in the BSDL file, the boldface characters "≐" are used to indicate assignment.
- White space (spaces, tabulation, carriage returns, etc.) is used in these BNF descriptions to provide enhanced readability and is not part of the syntax. However, white space needed for resolving lexical ambiguity (logical separation) is required as described in B.5.1

B.6.2 Lexical atoms

- A <VHDL identifier> is any valid identifier, chosen as a name for an item (see B.5.1.1).
- An <integer> is any valid unsigned VHDL integer made up of an unsigned numeric character sequence not containing an underscore (_) character and not using an exponent field.
- A <real number> is any valid VHDL real number of the form <integer>.<integer> or <integer>.<integer>**E**<integer> all written contiguously without spaces or format effectors. Note **1E3** is not real because it does not contain a decimal point. The number **20.0E6** is real, as is **20000000.0**.
- A <pattern> is a contiguous sequence of one or more **0**, **1**, and **X** characters containing no spaces or format effectors. For example, **001X00** and **XX010X** are legal. However, **100 X00** is not legal because of the embedded space. A low state (see 2.3.22) is denoted by **0**, a high state (see 2.3.13) is denoted by **1**, and a don't care value is denoted by **X**.
- A <32-bit pattern> is a <pattern> with exactly 32 characters in its character sequence.
- A <left bracket> is the left bracket character (**[**).
- A <right bracket> is the right bracket character (**]**).
- A <string> is any valid string or concatenated string structure meeting the definition in B.5.2.

Lexical ambiguity exists in certain situations and has to be resolved by context. For example, a <pattern> that starts with an **X** has to be differentiated from a <VHDL identifier> by context derived from the syntax. Similarly, a <pattern> that does not contain an **X** has to be differentiated from an integer such as **100** (one hundred).

B.6.3 Commonly used syntactic elements

- a) A <port ID> identifies a component *signal* that may be used to interface to external signals. A port may be dimensioned as a **bit** or a **bit_vector** (see B.8.3). Subscripted names are allowed only when **bit_vector**-dimensioned port signals are used.

```
<port ID> ::= <port name> 1 <subscripted port name>
<port name> ::= <VHDL identifier>
<subscripted port name> ::= <VHDL identifier> (<subscript>)
<subscript> ::= <integer>
```

- b) An <instruction name> is an instruction name defined in this standard or an instruction named by the manufacturer of a component.

```
<instruction name> ::= BYPASS | CLAMP | EXTEST | HIGHZ | IDCODE | INTTEST |  
RUNBIST | SAMPLE | USERCODE | <VHDL identifier>
```

NOTE — **SAMPLE** is used as an abbreviated name for *SAMPLE/PRELOAD*.

B.7 Components of a BSDL description

A BSDL description is composed of the following parts, as shown in part in figure B-1. These parts are

- a) *The entity description.* An entity description has to be written for each component. It specifies component-specific parameters of the test logic and might, for example, be written by the designer of a component.
- b) *The Standard VHDL Package and Standard VHDL Package Body.* These contain two types of information:
- 1) The Standard VHDL Package gives a definition of the BSDL subset of VHDL, which is required to allow BSDL descriptions to be read by a VHDL-conformant parser.
 - 2) The Standard VHDL Package Body gives definitions of commonly used types of boundary-scan cells. Typically, the Standard VHDL Package and Standard VHDL Package Body would reside with system-accompanying software that utilizes BSDL descriptions. Individual BSDL descriptions are not burdened with having to include all the elements contained in the Standard VHDL Package and Standard VHDL Package Body.

NOTE — The Standard VHDL Package and Standard VHDL Package Body have to be exactly as listed in B.9. They cannot be modified and would not normally be shipped along with BSDL files that describe particular components. Normally, the Standard VHDL Package and Standard VHDL Package Body will be supplied as a part of a BSDL-compliant tool and will be maintained by the tool supplier.

The cell definitions provided in the Standard VHDL Package Body could have been given within the Standard VHDL Package in a full VHDL implementation. The advantage of a package body is that the information it contains can be updated without causing the need for recompilation of all entities that reference the package. If a package is modified, then recompilation is necessary. The package with package body structure is a standard practice of BSDL.

- c) *User-specified VHDL packages and VHDL package bodies.* Users may provide VHDL packages and package bodies that define boundary-scan cell designs specific to any group of components. A vendor of application-specific ICs (ASICs), for example, could provide a user-specified VHDL package and package body to describe the particular boundary-scan cell designs offered. Global BSDL extensions could also be provided in user-specified VHDL packages (see B.10 and B.8.17).

B.8 The entity description

The entity description and supporting VHDL packages make up a BSDL model of the component and is, in effect, the electronic data sheet for its test logic. It contains statements through which parameters that may vary from one chip to another are defined, as discussed in B.3.

B.8.1 Overall structure of the entity description

B.8.1.1 Syntax

The BSDL entity description has to have the following structure:

```

<BSDL description>::=
  entity <component name> is
    <generic parameter>
    <logical port description>
    <standard use statement>
    {<use statement>}
    <component conformance statement>
    <device package pin mappings>
    [<grouped port identification>]
    <scan port identification>
    [<compliance enable description>]
    <instruction register description>
    [<optional register description>]
    [<register access description>]
    <boundary-scan register description>
    [<runbist description>]
    [<intest description>]
    {<BSDL extensions>}
    [<design warning>]
  end <component name>;

<component name>::= <VHDL identifier>

```

NOTE — While VHDL permits some elements within an entity description to be in an arbitrary order, the fixed ordering above is required for BSDL. This ordering is defined to ease the development of tools that are not themselves required to be fully VHDL compliant.

The <component name> identifies a particular integrated circuit and should be distinct from the names of all other components that may be used together (i.e., that may be assembled onto a single loaded board).

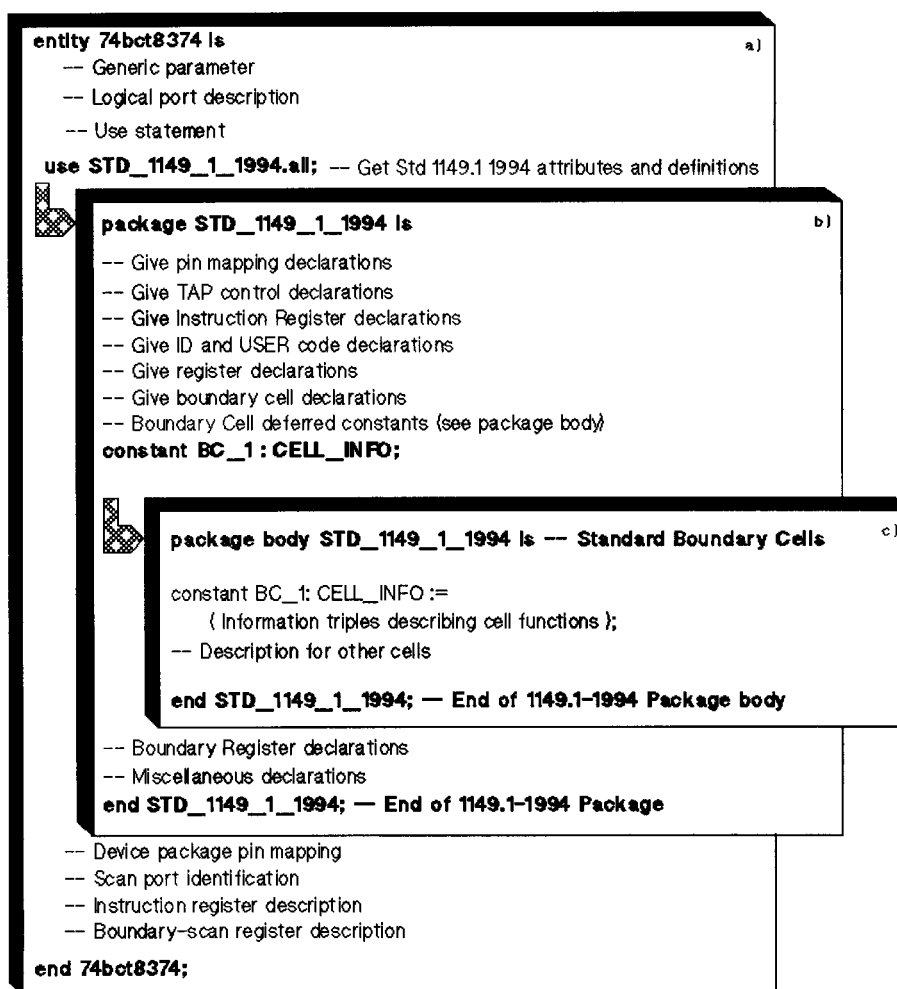


Figure B-1—Components of a BSDL description

B.8.1.2 Semantic Checks

- a) Any <component name> referenced in any attribute statement has to be the same as the component name declared in the entity description.

B.8.2 Generic parameter statement

The <generic parameter> statement facilitates selection between multiple component packaging options that are described within the BSDL description (see B.8.7). Each such option defines a mapping between the physical package pins of the component and the <port ID> elements of the component (see B.6.3).

The component package option relevant to a particular instance of a component is identified each time a given BSDL description is referenced.

B.8.2.1 Syntax

```
<generic parameter>::=
    generic (PHYSICAL_PIN_MAP: string); |
    generic (PHYSICAL_PIN_MAP: string:= <default device package type>);
<default device package type>::= " <VHDL identifier>"
```

In the first alternative <generic parameter> statement syntax, a <pin mapping name> (see B.8.7) that identifies a component package option has to be supplied with the BSDL description when it is referenced. In the second alternative syntax, a value is given for the <default device package type> that will be used as the default in the case in which no <pin mapping name> is supplied with the BSDL description when it is referenced. The <default device package type> has to be the quoted name of a <pin mapping name> used to specify the pin mapping for the component (see B.8.7).

B.8.2.2 Examples

```
generic (PHYSICAL_PIN_MAP: string);

or

generic (PHYSICAL_PIN_MAP: string:= "DW");
```

NOTE — It is recommended that the component package name from the data sheet of a given component be used as the relevant <pin mapping name>; for example, SSOP_56, PQFP_84, or PGA_18x18.

B.8.2.3 Semantic Checks

- a) It is an error if
 - 1) No <default device package type> is specified in the <generic parameter> statement of a BSDL description, and no package <pin mapping name> (see B.8.7) is specified when a BSDL description is processed
 - 2) The <default device package type> specified in the <generic parameter> statement is not identical to any <pin mapping name> appearing in any <pin mapping> (see B.8.7), and no <pin mapping name> is specified when the BSDL description is processed
 - 3) A <pin mapping name> is specified when a given BSDL description is processed, and that <pin mapping name> is not identical to any <pin mapping name> appearing in any <pin mapping> (see B.8.7) of that description

B.8.3 Logical port description statement

The BSDL port description is a specialization of the VHDL port list. It is used to assign meaningful symbolic names to the pins of a component. These symbolic names, which will be used in subsequent statements in the description, allow the majority of such statements to be independent of a renumbering or other reorganization of the pins of the component.

It is strongly recommended that the existence of nondigital pins (such as power, ground, no-connects, or analog signals) be recorded in the BSDL description of the component through use of the linkage <pin type> defined in the following syntax. BSDL applications may then be able to support substantially improved error detection during board testing; however, only a warning can be issued in the complete absence of such pins within a description since it is technologically possible for a component not to have any such pins.

B.8.3.1 Syntax

```

<logical port description>::= port (<pin spec> {;<pin spec> });
<pin spec>::= <identifier list>:<pin type><port dimension>
<identifier list>::= <port name>{,<port name>}
<pin type>::= in | out | buffer | inout | linkage
<port dimension>::= bit | bit_vector (<range>)
<range>::=<integer_1> to <integer_2> | <integer_2> downto <integer_1>
<integer_1>::=<integer>
<integer_2>::=<integer>

```

The definitions of the possible values of <pin type> are given in table B-2, and <port name> is defined in B.6.3.

Table B-2—Pin types

Value	Meaning
in	An input-only pin
out	An output-only pin that may be connected to a bus wire driven by multiple drivers (e.g., a 3-state or open-collector output)
buffer	A 2-state output-only pin where either state is always actively driven (e.g., cannot be connected to a bus wire)
inout	A bidirectional pin
linkage	Other pin types such as power, ground, analog, or no-connect

The <port dimension> defines the number of signals comprising a port. If the <port dimension> is **bit**, then one <port name> corresponds to that one signal. If the <port dimension> is **bit_vector**, then one <port name> corresponds to a collection of *n* signals that can be individually referenced by subscripting the port name, i.e., by a <subscripted port name> (see B.6.3).

NOTE — The signal types **bit** and **bit_vector** are signal types known to VHDL and are the only signal types permitted in BSDL. Also, while VHDL allows multiple, possibly broken ranges to be specified for a **bit_vector**, BSDL syntax only allows a single unbroken range.

B.8.3.2 Examples

```

port(  TDI, TMS, TCK: in bit;
      TDO: out bit; IN1, IN2: in bit;
      OUT1: out bit; OUT2: buffer bit;
      OUT3: out bit_vector (1 to 8);
      OUT4: out bit_vector (4 downto 1);
      BIDIR1, BIDIR2, BIDIR3: inout bit;
      GND, VCC: linkage bit);

```

B.8.3.3 Semantic Checks

- A **bit_vector** declared in a <logical port description> statement has to have the value of <integer_1> less than or equal to the value of <integer_2>.
- Each <port name> appearing in an <identifier list> of a <logical port description> has to occur only once within the <logical port description> statement.

B.8.4 Standard use statement

B.8.4.1 Syntax and semantics

The <standard use statement> identifies one Standard VHDL Package in which attributes, types, constants, and other elements are defined that will be referenced elsewhere in the BSDL description.

B.8.4.1.1 Syntax

<standard use statement>::= **use** <standard VHDL package identifier>.**all**;

<standard VHDL package identifier>::= **STD_1149_1_1994** | <other package identifier>

<other package identifier>::=<VHDL identifier>

The <standard VHDL package identifier> is the name of the Standard VHDL Package that contains the information to be included. The suffix **.all** indicates that all declarations within the VHDL package are to be used.

Values for <other package identifier> may be assigned with future revisions of this standard as the language evolves. For toolmakers desiring to support the earlier draft version of BSDL as well as the version specified in this standard, the value of <other package identifier> may be **STD_1149_1_1990** (see B.13.1).

NOTE — The construct of the <standard use statement> may not be syntactically complete for use by a given VHDL analyzer. This is because a library or default working area has not been specified. In such a case, a complete statement is “use work.STD_1149_1_1994.all;” in which the preface “work.” tells a VHDL analyzer to find the Standard VHDL Package in the current work area. The field “work.” could be replaced by an arbitrary library name such as “BSCAN.” telling a VHDL analyzer where in its system of libraries to find the Standard VHDL Package. Since there is no standardization of library structures from one VHDL environment to another, some editing of BSDL files to specify the location of Standard VHDL Packages is generally unavoidable. The specification used in this subclause may cause an error in a VHDL analyzer, forcing the user to edit the BSDL file for the correct location of the Standard VHDL Package information.

The <standard VHDL package identifier> indicates

- An instruction to tools to read a standard package that contains BSDL syntax definitions
- The version of the BSDL language (shown by the year number embedded in the identifier) that was used when creating the BSDL file

As a general rule, future enhancements to the BSDL language will be designed as extensions to previous versions of the language. Therefore, the version information contained in the <standard VHDL package identifier> can be used by BSDL-specific tools to indicate which tool versions will be able to process the information in the input file. For example, if the input file records the version as “_1994”, then tools that can process any language variant “_1994” or later will be able to process the input file correctly.

NOTE — Versions of the BSDL language should not be confused with the version of the standard itself to which a given IC may conform. The <component conformance statement> identifies the version of the standard (see B.8.6).

Within a specific system processing BSDL’ descriptions, the Standard VHDL Package may be a file somewhere in the file system of the host computer. The **.all** suffix is meaningful to VHDL and is not part of a file name. While VHDL permits the use of a wider range of suffixes, **.all** is the only suffix permitted in BSDL.

The content of the Standard VHDL Package is the current definition of the BSDL language and cannot be modified by users.

The <standard use statement> has to appear in every BSDL description before any <use statement> (see B.8.5) so that tools that are fully VHDL compliant can locate information relevant to all components that conform to this standard. (Tools that are limited to the BSDL language always use this information.)

B.8.4.1.2 Examples

```
use STD_1149_1_1994.all;      -- Today

or

use STD_1149_1_2012.all;      -- Sometime in the future
```

The contents of the **STD_1149_1_1994** Standard VHDL Package and its associated Standard VHDL Package Body are listed in B.9. (The contents of the **STD_1149_1_1990** Standard VHDL Package and its associated Standard VHDL Package Body are listed in B.13.1.)

B.8.4.1.3 Semantic Checks

- a) The Standard VHDL Package content has to be used when processing a BSDL description.

The <standard use statement> is also used for version control (see B.8.4.2).

B.8.4.2 Version control

At the time this standard was approved, there were two versions of BSDL descriptions, the 1990 version (see B.13) and the version described herein. The <standard use statement> indicates whether the BSDL description has been written using the provisional syntax published in the *Proceedings of the International Test Conference* in 1990 or according to this annex. This additional application of the <standard use statement> is intended to provide backwards compatibility to all BSDL descriptions already written and can be used in a similar manner for future revisions of this standard. It is intended that a parser handling a form of BSDL described in a version of this annex *approved by the IEEE Standards Board and published by the IEEE* will handle all forms of BSDL described in previous versions of this annex *approved by the IEEE Standards Board and published by the IEEE*.

B.8.4.2.1 Examples

```
use STD_1149_1_1994.all;
```

In the 1990 version of BSDL, the following syntactic elements are not supported:

```
<component conformance statement>
<grouped port identification>
<compliance enable description>
<runbist description>
<intest description>
<BSDL extensions>
```

Also, in the 1990 version, cell types **BC_0** and **BC_7** that have been identified by the 1994 version need to be specified in a user-supplied VHDL package if they are to be referenced.

Software that processes BSDL descriptions needs to have access to all Standard VHDL Packages related to BSDL. See B.9 and B.13 for the two that are currently defined.

B.8.4.2.2 Semantic Checks

- a) A BSDL parser is expected to reject and/or issue warning/error messages when it encounters unrecognized text.

NOTE — Specifically constructed foreign attributes may be added via BSDL Extensions (see B.8.17).

B.8.5 Use statement

The optional <use statement> identifies a VHDL package in which specific attributes or constants are defined so that they can be referenced elsewhere in a BSDL description.

B.8.5.1 Syntax

```
<use statement> ::= use <user VHDL package identifier>.all;  
<user VHDL package identifier> ::= <VHDL identifier>
```

The <user VHDL package identifier> in the <use statement> is the name of the VHDL package that contains the information to be included. The suffix **.all** indicates that all declarations within the VHDL package are to be used. The **.all** suffix is meaningful to VHDL and is not part of a file name. While VHDL permits the use of a wider range of suffixes, **.all** is the only suffix permitted in BSDL. See B.10.1 for the content of a user-supplied package.

B.8.5.2 Examples

```
use Private_Package.all;    -- Identifies the proprietary VHDL package of a user
```

NOTE — The construct of the <use statement> may not be syntactically complete for use by a given VHDL analyzer. This is because a library or default working area has not been specified. In such a case, a complete statement is “use work. Private.all;” in which the preface “work.” tells a VHDL analyzer to find the user-supplied VHDL package in the current work area. The field “work.” could be replaced by an arbitrary library name such as “BSCAN.” telling a VHDL analyzer where in its system of libraries to find the user-supplied VHDL package. Since there is no standardization of library structures from one VHDL environment to another, some editing of BSDL files to specify the location of a user-supplied VHDL package is generally unavoidable. This specification may cause an error in a VHDL analyzer, forcing the user to edit the BSDL file for the correct location of the user-supplied VHDL package information.

In full VHDL, two use statements may be given, referencing VHDL packages that each contain different definitions of the same item, such as a Boundary Cell. Later, the ambiguity of which package is being referenced can be removed by qualifying each reference in a specified manner. This facility is not supported in BSDL.

B.8.5.3 Semantic Checks

- a) Information in a VHDL package named in a <use statement> always has to be used when processing a BSDL description (see B.10).
- b) Identifiers declared in VHDL packages cannot be defined
 - 1) More than once in a single VHDL package
 - 2) In more than one VHDL package referenced by a <use statement> or <standard use statement> in a single BSDL description

B.8.6 Component conformance statement

The <component conformance statement> identifies the edition of the standard to which the testability circuitry of a physical component conforms. It is possible for a component designed in 1990 to be described by this standard, but this cannot imply that the component conforms to the rules of IEEE Std 1149.1a-1993. For example, IEEE Std 1149.1-1990 allowed (by omission of rules) two drivers controlled by a single control cell to be disabled by opposing values loaded