

IEC 62531

Edition 2.0 2012-06

IEEE Std 1850[™]

INTERNATIONAL

Property Specification Language (PSL)



THIS PUBLICATION IS COPYRIGHT PROTECTED Copyright © 2010 IEEE

All rights reserved. IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Inc.

Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the IEC Central Office.

Any questions about IEEE copyright should be addressed to the IEEE. Enquiries about obtaining additional rights to this publication and other information requests should be addressed to the IEC or your local IEC member National Committee.

IEC Central Office 3, rue de Varembé CH-1211 Geneva 20 Switzerland Tel.: +41 22 919 02 11 Fax: +41 22 919 03 00 info@iec.ch www.iec.ch Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue New York, NY 10016-5997 United States of America stds.info@ieee.org www.ieee.org

About the IEC

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

About IEC publications

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

Useful links:

IEC publications search - www.iec.ch/searchpub

The advanced search enables you to find IEC publications by a variety of criteria (reference number, text, technical committee,...).

It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available on-line and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary (IEV) on-line.

Customer Service Centre - webstore.iec.ch/csc

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



IEC 62531

Edition 2.0 2012-06

IEEE Std 1850™

INTERNATIONAL STANDARD - Changer is

Property Specification Language (PSL)

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE

ICS 25.040; 35.060

ISBN 978-2-83220-106-0

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

Contents

1.1 Scope 1 1.2 Purpose 1 1.2.1 Background 2 1.2.2 Motivation 2 1.2.3 Goals 2 1.3 L Sage 2 1.3.1 Functional specification 3 1.3.2 Functional verification 3 2. Normative references 7 3. Definitions, acronyms, and abbreviations 9 3.1 Definitions 9 3.2 Acronyms and abbreviations 12 3.3 Special terms 12 4.1 Organization 15 4.1.1 Layers 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Identifiers 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Conventions 24 4.3.1 Conventions 24 4.3.2 Stortey set inclosed evaluation 29 4.4.3 Sinpteendenen	1.	Ove	erview	1				
12 Purpose 1 1.2.1 Background 2 1.2.2 Motivation 2 1.3.1 Functional specification 3 1.3.2 Functional verification 3 1.3.2 Functional verification 3 2. Normative references 7 3. Definitions, acronyms, and abbreviations 9 3.1 Definitions 9 3.2 Acronyms and abbreviations 12 3.3 Special terms 12 3.3 Special terms 12 4.1 Organization 15 4.1.2 Flavors 15 4.1.2 Flavors 15 4.1.2 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 29 4.4.3 Unert ws. branching logic 30 4.4.4.5 Finitte-length vs. liminite-length behavior 3								
12.1 Background 2 1.2.2 Motivation 2 1.3 Voiration 2 1.3 Usage 2 1.3 L'accondustry 2 1.3 L'accondustry 3 1.3.2 Functional specification 3 1.3.2 Functional specification 3 2. Normative references 7 3. Definitions, acronyms, and abbreviations 9 3.1 Definitions 9 3.1 Definitions 12 3.3 Special terms 12 3.3 Special terms 12 3.3 Special terms 12 4. Organization 15 4.1 Abstract structure 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Identifiers 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros								
12.2 Motivation 2 1.3 Goals 2 1.3 Functional specification 3 1.3.1 Functional verification 3 1.3.2 Functional verification 3 2. Normative references 7 3. Definitions, acronyms and abbreviations 9 3.1 Definitions 9 3.2 Acronyms and abbreviations 12 3.3 Special terms 12 3.3 Special terms 12 4. Organization 15 4.1 Abstract structure 15 4.1.2 Flavors 15 4.1.2 Flavors 15 4.1.2 Flavors 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.3.2 Operators 17 4.3.2 Operators 17 4.3.3 HDL dependencies 29 4.4.3 <t< td=""><td></td><td>1.2</td><td></td><td></td></t<>		1.2						
1.3 Usage 2 1.3 Usage 2 1.3.1 Functional specification 3 1.3.2 Functional verification 3 2. Normative references 7 3. Definitions, acronyms, and abbreviations 9 3.1 Definitions 9 3.2 Acronyms and abbreviations 9 3.3 Special terms 12 4. Organization 15 4.1 Abstract structure 15 4.1.2 Flavors 15 4.1.2 Flavors 15 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength. 31 5. Boolean layer 33 5.1.1 Bit expressions 33 5.1.2 String expressions 33 5.1.3 Stilty cettor expressions 35 5.1.4 Nu			1.2.1 Background	2				
1.3 Usage 2 1.3.1 Functional specification 3 1.3.2 Functional verification 3 2. Normative references 7 3. Definitions, acronyms, and abbreviations 9 3.1 Definitions 9 3.2 Acronyms and abbreviations 9 3.3 Special terms 12 3.3 Special terms 12 3.3 Special terms 12 4. Organization 15 4.1 Abstract structure 15 4.1.1 Layors 15 4.1.2 Flavors 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Vintax 24 4.3 Vintax 24 4.3 Undependencies 25 4.4 Structure vs. branching logic 30 4.4.4 Si			1.2.2 Motivation	2				
1.3.1 Functional specification 3 1.3.2 Functional verification 3 2. Normative references 7 3. Definitions, acronyms, and abbreviations 9 3.1 Definitions, acronyms and abbreviations 9 3.1 Definitions, acronyms and abbreviations 9 3.2 Acronyms and abbreviations 12 3.3 Special terms 12 3.3 Special terms 12 4. Organization 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Very ords 16 4.2.1 Identifiers 16 4.2.2 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Stores sproperties 30 4.4.1 Licocked vs. unclocked evaluat			1.2.3 Goals	2				
1.3.1 Functional specification 3 1.3.2 Functional verification 3 2. Normative references 7 3. Definitions, acronyms, and abbreviations 9 3.1 Definitions, acronyms and abbreviations 9 3.1 Definitions, acronyms and abbreviations 9 3.2 Acronyms and abbreviations 12 3.3 Special terms 12 3.3 Special terms 12 4. Organization 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Very ords 16 4.2.1 Identifiers 16 4.2.2 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Stores sproperties 30 4.4.1 Licocked vs. unclocked evaluat		1.3	Usage	2				
1.3.2 Functional verification 3 2. Normative references 7 3. Definitions, acronyms, and abbreviations 9 3.1 Definitions 9 3.2 Acronyms and abbreviations 12 3.3 Special terms 12 3.4 Organization 12 4. Organization 15 4.1 Abstract structure 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Verywords 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.4 Astrocos 22 4.2.5 Comments 24 4.3 Oroventions 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Sinching logic 30 4.4.3 Linear vs. branching logic 30 4.4.4 Sinpeitse subset 30 4.4.4 Sinpeis ubset 30 4.4.4								
2. Normative references 7 3. Definitions, acronyms, and abbreviations 9 3.1 Definitions 9 3.2 Acronyms and abbreviations 12 3.3 Special terms 12 3.4 Organization 15 4.1 Abstract structure 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Flavors 15 4.1.2 Verical structure 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3 Linear vs. branching logic 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5.1 Expression type classes <								
3. Definitions, acronyms and abbreviations. 9 3.1 Definitions 9 3.2 Acronyms and abbreviations 12 3.3 Special terms 12 3.4 Organization 12 4.1 Organization 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Flavors 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.4 Macros 22 4.2.5 Comments 24 4.3.1 Conventions 24 4.3.3 HDL dependencies 25 4.4 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Simple subset 30 4.4.3 Simple subset 30 4.4.4 Simple subset 30 4.4.5 Fini								
3.1 Definitions 9 3.2 Acronyms and abbreviations 12 3.3 Special terms 12 4. Organization 15 4.1 Abstract structure 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Versions 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 33 5.1.2 Boolean expressions 33 5.1.3 BitVector expressions 35 5.1.4 Numeric expressions 35 5.1.5 Str	2.	Nor	mative references	7				
3.1 Definitions 9 3.2 Acronyms and abbreviations 12 3.3 Special terms 12 4. Organization 15 4.1 Abstract structure 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Versions 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 33 5.1.2 Boolean expressions 33 5.1.3 BitVector expressions 35 5.1.4 Numeric expressions 35 5.1.5 Str								
3.2 Acronyms and abbreviations 12 3.3 Special terms 12 4. Organization 15 4.1 Abstract structure 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Flavors 15 4.1.2 Flavors 16 4.2.1 Identifiers 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Thile supressions 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 33 5.1	3.	Def	Definitions, acronyms, and abbreviations					
3.2 Acronyms and abbreviations 12 3.3 Special terms 12 4. Organization 15 4.1 Abstract structure 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Flavors 15 4.1.2 Flavors 16 4.2.1 Identifiers 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Thile supressions 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 33 5.1								
3.3 Special terms 12 4. Organization 15 4.1 Abstract structure 15 4.1.2 Flavors 15 4.1.2 Flavors 15 4.1.2 Flavors 15 4.1.2 Flavors 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs, unclocked evaluation 29 4.4.2 Safety vs, liveness properties 30 4.4.3 Simple subset 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs, infinite-length behavior 31 5.1 Boolean layer 33 5.1.1 Bit expressions 33 5.1.2 Boolean expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 36 5.2 Expression forms 36		3.1	Definitions	9				
4. Organization. 15 4.1 Abstract structure 15 4.1.1 Layers. 15 4.1.2 Flavors 15 4.1.2 Flavors 15 4.1.3 Lexical structure 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Santics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties. 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 5.1 Boolean layer 33 5.1.1 Bit expressions 33 5.1.2 Boolean expressions 35								
4.1 Abstract structure 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Lexical structure 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 33 5.1.2 Boolean expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 36 5.2 Expression forms 36		3.3	Special terms					
4.1 Abstract structure 15 4.1.1 Layers 15 4.1.2 Flavors 15 4.1.2 Lexical structure 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 33 5.1.2 Boolean expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 36 5.2 Expression forms 36		_						
4.1.1 Layers	4.	Org	anization	15				
4.1.1 Layers		4 1	A hatmast atmusture	15				
4.1.2 Flavors 15 4.2 Lexical structure 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5. Boolean layer 33 5.1 Bit expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 35 5.1.4 Numeric expressions <td< td=""><td></td><td>4.1</td><td></td><td> 13</td></td<>		4.1		13				
4.2 Lexical structure 16 4.2.1 Identifiers 16 4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Safety vs. liveness properties 30 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 35 5.1.4 Numeric expressions 35 5.1								
4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit vector expressions 34 5.1.3 BitVector expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 36								
4.2.2 Keywords 16 4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit vector expressions 34 5.1.3 BitVector expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 36		4.2	Lexical structure	16				
4.2.3 Operators 17 4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 34 5.1.3 BitVector expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 36								
4.2.4 Macros 22 4.2.5 Comments 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Safety vs. luclocked evaluation 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 34 5.1.3 BitVector expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 36			4.2.2 Keywords	16				
42.5 Comments 24 4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.2 Boolean expressions 34 5.1.3 BitVector expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 36 5.2 Expression forms 36								
4.3 Syntax 24 4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.2 Boolean expressions 33 5.1.3 BitVector expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expression forms 36								
4.3.1 Conventions 24 4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 33 5.1.2 Boolean expressions 34 5.1.3 BitVector expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 36								
4.3.2 HDL dependencies 25 4.4 Semantics 29 4.4.1 Clocked vs. unclocked evaluation 29 4.4.2 Safety vs. liveness properties 30 4.4.3 Linear vs. branching logic 30 4.4.4 Simple subset 30 4.4.5 Finite-length vs. infinite-length behavior 31 4.4.6 The concept of strength 31 5. Boolean layer 33 5.1 Expression type classes 33 5.1.1 Bit expressions 33 5.1.2 Boolean expressions 34 5.1.3 BitVector expressions 35 5.1.4 Numeric expressions 35 5.1.5 String expressions 36		4.3						
4.4 Semantics294.4.1 Clocked vs. unclocked evaluation294.4.2 Safety vs. liveness properties304.4.3 Linear vs. branching logic304.4.4 Simple subset304.4.5 Finite-length vs. infinite-length behavior314.4.6 The concept of strength315. Boolean layer335.1 Expression type classes335.1.1 Bit expressions335.1.2 Boolean expressions345.1.3 BitVector expressions355.1.4 Numeric expressions355.1.5 String expressions36			4.3.1 Conventions					
4.4 Semantics294.4.1 Clocked vs. unclocked evaluation294.4.2 Safety vs. liveness properties304.4.3 Linear vs. branching logic304.4.4 Simple subset304.4.5 Finite-length vs. infinite-length behavior314.4.6 The concept of strength315. Boolean layer335.1 Expression type classes335.1.1 Bit expressions335.1.2 Boolean expressions345.1.3 BitVector expressions355.1.4 Numeric expressions355.1.5 String expressions36			4.3.2 HDL dependencies					
4.4.2Safety vs. liveness properties.304.4.3Linear vs. branching logic304.4.4Simple subset304.4.5Finite-length vs. infinite-length behavior314.4.6The concept of strength315.Boolean layer335.1Expression type classes335.1.1Bit expressions335.1.2Boolean expressions345.1.3BitVector expressions355.1.4Numeric expressions355.1.5String expressions365.2Expression forms36		4.4						
4.4.2Safety vs. liveness properties.304.4.3Linear vs. branching logic304.4.4Simple subset304.4.5Finite-length vs. infinite-length behavior314.4.6The concept of strength315.Boolean layer335.1Expression type classes335.1.1Bit expressions335.1.2Boolean expressions345.1.3BitVector expressions355.1.4Numeric expressions355.1.5String expressions365.2Expression forms36			4.4.1 Clocked vs. unclocked evaluation					
4.4.3Linear vs. branching logic304.4.4Simple subset304.4.5Finite-length vs. infinite-length behavior314.4.6The concept of strength315.Boolean layer335.1Expression type classes335.1.1Bit expressions335.1.2Boolean expressions345.1.3BitVector expressions355.1.4Numeric expressions355.1.5String expressions36								
4.4.4Simple subset304.4.5Finite-length vs. infinite-length behavior314.4.6The concept of strength315.Boolean layer335.1Expression type classes335.1.1Bit expressions335.1.2Boolean expressions345.1.3BitVector expressions355.1.4Numeric expressions355.1.5String expressions36								
4.4.5Finite-length vs. infinite-length behavior314.4.6The concept of strength315.Boolean layer335.1Expression type classes335.1.1Bit expressions335.1.2Boolean expressions345.1.3BitVector expressions355.1.4Numeric expressions355.1.5String expressions365.2Expression forms36								
4.4.6 The concept of strength			445 Finite-length vs infinite-length behavior	31				
5.Boolean layer335.1Expression type classes335.1.1Bit expressions335.1.2Boolean expressions345.1.3BitVector expressions355.1.4Numeric expressions355.1.5String expressions365.2Expression forms36								
5.1 Expression type classes335.1.1 Bit expressions335.1.2 Boolean expressions345.1.3 BitVector expressions355.1.4 Numeric expressions355.1.5 String expressions365.2 Expression forms36								
5.1 Expression type classes335.1.1 Bit expressions335.1.2 Boolean expressions345.1.3 BitVector expressions355.1.4 Numeric expressions355.1.5 String expressions365.2 Expression forms36	5.	Boo	olean layer					
5.1.1Bit expressions335.1.2Boolean expressions345.1.3BitVector expressions355.1.4Numeric expressions355.1.5String expressions365.2Expression forms36								
5.1.2Boolean expressions345.1.3BitVector expressions355.1.4Numeric expressions355.1.5String expressions365.2Expression forms36		5.1	Expression type classes					
5.1.3BitVector expressions355.1.4Numeric expressions355.1.5String expressions365.2Expression forms36								
5.1.3BitVector expressions355.1.4Numeric expressions355.1.5String expressions365.2Expression forms36								
5.1.4 Numeric expressions355.1.5 String expressions365.2 Expression forms36								
5.1.5String expressions365.2Expression forms36								
5.2 Expression forms								
		5.2						
			*					

2	5.2.2 PSL expressions5.2.3 Built-in functions	
	5.2.4 Union expressions	45
	5.3 Clock expressions	
	5.4 Default clock declaration	47
6.	Temporal layer	49
	6.1 Sequential expressions	
	6.1.1 Sequential Extended Regular Expressions (SEREs)	
	6.1.2 Sequences	
	6.2 Properties	
	6.2.1 FL properties	
	6.2.2 Optional Branching Extension (OBE) properties	
	6.2.3 Replicated properties	
	6.3 Local variables	
	6.4 Procedural blocks	
	6.5 Property and sequence declarations	
	6.5.1 Parameters	
	6.5.2 Declarations	
	6.5.3 Instantiation	107
-	Verification layer	
7.	Verification layer	
	7.1 Verification directives	111
	7.1.1 assert	
	7.1.2 assume	
	7.1.3 restrict	
	7.1.4 restrict!	
	7.1.5 cover	
	7.1.6 fairness and strong_fairness	
	7.2 Verification units	
	7.2.1 Verification unit binding	
	7.2.2 Verification unit instantiation	
	7.2.3 Verification unit inheritance	
	7.2.4 Overriding assignments	
0	Modeling layer	100
8.	Modeling layer	129
		100
	8.1 Integer ranges	
	8.2 Structures	
0		101
9.	Scope and visibility rules	
		121
	9.1 Immediate scope	
	9.2 Extended scope	
	9.3 Direct and indirect name references	
Anne	x A (normative) Syntax rule summary	135
		$\langle \rangle \rangle$
Anne	x B (normative) Formal Syntax and Semantics of IEEE Std 1850 Property Specification La	
	(PSL)	
Anne	x C (informative) Bibliography	
Anne	x D (informative) List of IEEE Participants	169

Property Specification Language (PSL)

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.

IEEE Standards documents are developed within IEEE Societies and Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. IEEE develops its standards through a consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of IEEE and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards. Use of IEEE Standards documents is wholly voluntary. IEEE documents are made available for use subject to important notices and legal disclaimers (see http://standards.ieee.org/IPR/disclaimers.html for more information).

IEC collaborates closely with IEEE in accordance with conditions determined by agreement between the two organizations.

- 2) The formal decisions of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees. The formal decisions of IEEE on technical matters, once consensus within IEEE Societies and Standards Coordinating Committees has been reached, is determined by a balanced ballot of materially interested parties who indicate interest in reviewing the proposed standard. Final approval of the IEEE standards document is given by the IEEE Standards Association (IEEE-SA) Standards Board.
- 3) IEC/IEEE Publications have the form of recommendations for international use and are accepted by IEC National Committees/IEEE Societies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC/IEEE Publications is accurate, IEC or IEEE cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications (including IEC/IEEE Publications) transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC/IEEE Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC and IEEE do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC and IEEE are not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or IEEE or their directors, employees, servants or agents including individual experts and members of technical committees and IEC National Committees, or volunteers of IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board, for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC/IEEE Publication or any other IEC or IEEE Publications.
- 8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that implementation of this IEC/IEEE Publication may require use of material covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. IEC or IEEE shall not be held responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility.

International Standard IEC 62531/ IEEE Std 1850-2010 has been processed through IEC technical committee 93: Design automation, under the IEC/IEEE Dual Logo Agreement.

This second edition cancels and replaces the first edition, published in 2007, and constitutes a technical revision.

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

IEEE Std	FDIS	Report on voting
IEEE Std 1850-2010	93/319/FDIS	93/326/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.

The IEC Technical Committee and IEEE Technical Committee have decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed. •
- withdrawn, .
- ore tien on or or or of the or of th replaced by a revised edition, or •
- amended. •

IEEE Standard for Property Specification Language (PSL)

4

Sponsor

Design Automation Standards Committee of the **IEEE Computer Society**

and the IEEE Standards Association Corporate Advisory Group

Approved 25 March 2010 **IEEE-SA Standards Board**

Grateful acknowledgment is made to Accellera Organization, Inc. for the permission to use the following source material:

Accellera Property Specification Language Reference Manual (version 1.1), Accellera

GDL: General Description Language, Accellera, Mar. 2005

Abstract: The IEEE Property Specification Language (PSL) is defined. PSL is a formal notation for specification of electronic system behavior, compatible with multiple electronic system design languages, including IEEE Std 1076[™] (VHDL[®]), IEEE Std 1354 (Verilog[®]), IEEE Std 1666[™] (SystemC[®]), and IEEE Std 1800[™] (SystemVerilog[®]), thereby enabling a common specification and verification flow for multi-language and mixed-language designs. PSL captures design intent in a form suitable for simulation, formal verification, formal analysis, and hybrid verification tools. PSL enhances communication among architects, designers, and verification engineers to increase productivity throughout the design and verification process. The primary audiences for this standard are the implementors of tools supporting the language and advanced users of the language.

Keywords: ABV, assertion, assertion-based verification, assumption, cover, model checking, property, PSL, specification, temporal logic, verification

IEEE, 802, and POSIX are registered trademarks in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

VHDL and Verilog are both registered trademarks of Cadence Design Systems, Inc.

SystemVerilog is a registered trademark of Accellera Organization, Inc.

SystemC is a registered trademark of Synopsys, Inc.

IEEE Introduction

This introduction is not part of IEEE Std 1850-2010, IEEE Standard for Property Specification Language (PSL).

IEEE Std 1850 Property Specification Language (PSL) is based upon the Accellera Property Specification Language (Accellera PSL), a language for formal specification of electronic system behavior, which was developed by Accellera, a consortium of Electronic Design Automation (EDA), semiconductor, and system companies. IEEE Std 1850 PSL 2010 refines IEEE Std 1850 PSL 2005 by providing extensions for improved verification IP reuse (e.g., the vpkg type of vunit) and interaction between the assertions and the simulation environment (local variables), and by addressing minor technical issues. The formal semantics were updated to reflect these changes.

Notice to users

Laws and regulations

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

Copyrights

This document is copyrighted by the IEEE. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association website at http://ieeexplore.ieee.org/xpl/standards.jsp, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA website at http://standards.ieee.org.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http:// standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/ index html.

Patents

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

Property Specification Language (PSL)

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

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

1. Overview

1.1 Scope

This standard defines the property specification language (PSL), which formally describes electronic system behavior. This standard specifies the syntax and semantics for PSL and also clarifies how PSL interfaces with various standard electronic system design languages.

1.2 Purpose

The purpose of this standard is to provide a well-defined language for formal specification of electronic system behavior, one that is compatible with multiple electronic system design languages, including IEEE Std 1076[™] (VHDL[®]),¹ IEEE Std 1364[™] (Verilog[®]), IEEE Std 1800[™] (SystemVerilog[®]), and IEEE Std 1666[™] (SystemC[®]), to facilitate a common specification and verification flow for multi-language and mixed-language designs.

This standard creates an updated IEEE standard based upon IEEE Std 1850-2005. The updated standard will refine IEEE standard, addressing errata, minor technical issues, and proposed extensions specifically related to property reuse and improved simulation usability.

¹Information on references can be found in Clause 2.

1.2.1 Background

The complexity of Very Large Scale Integration (VLSI) has grown to such a degree that traditional approaches have begun to reach their limitations, and verification costs have reached 60%–70% of development resources. The need for advanced verification methodology, with improved observability of design behavior and improved controllability of the verification process, has become critical. Over the last decade, a methodology based on the notion of "properties" has been identified as a powerful verification paradigm that can assure enhanced productivity, higher design quality, and, ultimately, faster time to market and higher value to engineers and end-users of electronics products. Properties, as used in this context, are concise, declarative, expressive, and unambiguous specifications of desired system behavior that are used to guide the verification process. IEEE 1850 PSL is a standard language for specifying electronic system behavior using properties. PSL facilitates property-based verification using both simulation and formal verification, thereby enabling a productivity boost in functional verification.

1.2.2 Motivation

Ensuring that a design's implementation satisfies its specification is the foundation of hardware verification. Key to the design and verification process is the act of specification. Yet historically, the process of specification has consisted of creating a natural language description of a set of design requirements. This form of specification is both ambiguous and, in many cases, unverifiable due to the lack of a standard machine-executable representation. Furthermore, ensuring that all functional aspects of the specification have been adequately *verified* (that is, covered) is problematic.

The IEEE PSL was developed to address these shortcomings. It gives the design architect a standard means of specifying design properties using a concise syntax with clearly-defined formal semantics. Similarly, it enables the RTL implementer to capture design intent in a verifiable form, while enabling the verification engineer to validate that the implementation satisfies its specification through *dynamic* (that is, simulation) and *static* (that is, formal) verification means. Furthermore, it provides a means to measure the quality of the verification process through the creation of functional coverage models built on formally specified properties. In addition, it provides a standard means for hardware designers and verification engineers to create a rigorous and machine-executable design specification.

1.2.3 Goals

PSL was specifically developed to fulfill the following general hardware functional specification requirements:

- Easy to learn, write, and read
- Concise syntax
- Rigorously well-defined formal semantics
- Expressive power, permitting specifications of a large class of real-world design properties
- Known efficient underlying algorithms in simulation, as well as formal verification

1.3 Usage

PSL is a language for the formal specification of hardware. It is used to describe properties that are required to hold in the design under verification. PSL provides a means to write specifications that are both easy to read and mathematically precise. It is intended to be used for functional specification on the one hand and as input to functional verification tools on the other. Thus, a PSL specification is an executable specification of a hardware design.

1.3.1 Functional specification

PSL can be used to capture requirements regarding the overall behavior of a design, as well as assumptions about the environment in which the design is expected to operate. PSL can also capture internal behavioral requirements and assumptions that arise during the design process. Both enable more effective functional verification and reuse of the design.

One important use of PSL is for documentation, either in place of or along with an English specification. A PSL specification can describe simple invariants (for example, signals read_enable and write_enable are never asserted simultaneously) as well as multi-cycle behavior (for example, correct behavior of an interface with respect to a bus protocol or correct behavior of pipelined operations).

A PSL specification consists of *assertions* regarding *properties* of a design under a set of *assumptions*. A *property* is built from three kinds of elements: *Boolean expressions*, which describe behavior over one cycle; *sequential expressions*, which can describe multi-cycle behavior; and *temporal operators*, which describe temporal relationships among Boolean expressions and sequences. For example, consider the following Verilog Boolean expression:

ena || enb

This expression describes a cycle in which at least one of the signals ena and enb are asserted. The PSL sequential expression

{req; ack; !cancel}

describes a sequence of cycles, such that req is asserted in the first cycle, ack is asserted in the second cycle, and cancel is deasserted in the third cycle. The following property, obtained by applying the temporal operators always and |=> to these expressions,

always {req;ack;!cancel} |=> (ena || enb)

means that always (that is, in every cycle), if the sequence {req;ack;!cancel} occurs, then either ena or enb is asserted one cycle after the sequence ends. Adding the directive assert as follows:

assert always {req;ack;!cancel} |=> (ena || enb);

completes the specification, indicating that this property is expected to hold in the design and that this expectation needs to be verified.

1.3.2 Functional verification

PSL can also be used as input to verification tools, for both verification by simulation, as well as formal verification using a model checker or a theorem prover. Each of these is discussed in the subclauses that follow.

1.3.2.1 Simulation

A PSL specification can also be used to automatically generate checks of simulated behavior. This can be done, for example, by directly integrating the checks in the simulation tool; by interpreting PSL properties in a testbench automation tool that drives the simulator; by generating HDL monitors that are simulated alongside the design; or by analyzing the traces produced during simulation.

For instance, the following PSL property:

Property 1: always (req -> next !req)

states that signal req is a pulsed signal, i.e., if it is high in some cycle, then it is low in the following cycle. Such a property can be easily checked using a simulation checker written in some HDL that has the functionality of the finite state machine (FSM) shown in Figure 1.

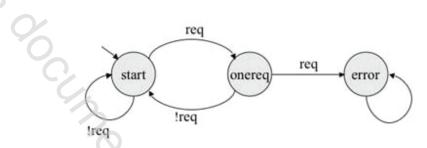


Figure 1—A simple (deterministic) FSM that checks Property 1

For properties more complicated than the property shown in Figure 1, manually writing a corresponding checker is painstaking and error-prone, and maintaining a collection of such checkers for a constantly changing design under development is a time-consuming task. Instead, a PSL specification can be used as input to a tool that automatically generates simulatable checkers.

Although in principle, all PSL properties can be checked for finite paths in simulation, the implementation of the checks is often significantly simpler for a subset called the *simple subset* of PSL. Informally, in this subset, composition of temporal properties is restricted to ensure that time *moves forward* from left to right through a property, as it does in a timing diagram. (See 4.4.4 for the formal definition of the simple subset.) For example, the property

```
Property 2: always (a -> next[3] b)
```

which states that, if a is asserted, then b is asserted three cycles later, belongs to the simple subset, because a appears to the left of b in the property and also appears to the left of b in the timing diagram of any behavior that is not a violation of the property. Figure 2 shows an example of such a timing diagram.

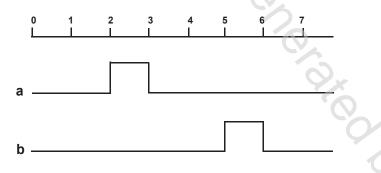


Figure 2—A trace that satisfies Property 2

An example of a property that is not in this subset is the property

Property 3: always ((a && next[3] b) -> c)

which states that, if a is asserted and b is asserted three cycles later, then c is asserted (in the same cycle as a). This property does not belong to the simple subset, because although c appears to the right of a and b in

the property, it appears to the left of b in a timing diagram that is not a violation of the property. Figure 3 shows an example of such a timing diagram.

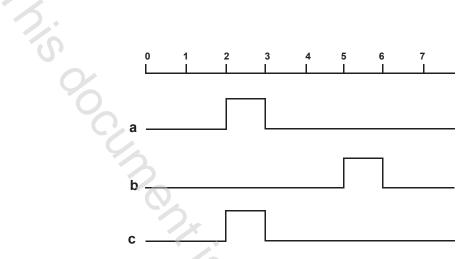


Figure 3—A trace that satisfies Property 3

1.3.2.2 Formal verification

PSL is an extension of the standard temporal logics Linear-Time Temporal Logic (LTL) and Computation Tree Logic (CTL). A specification in the PSL Foundation Language (respectively, the PSL Optional Branching Extension) can be *compiled down* to a formula of pure LTL (respectively, CTL), possibly with some auxiliary HDL code, known as a *satellite*.

IEC 62531:2012

this document is a proview of menanda by FLS

2. Normative references

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

"General Description Language," Accellera, Napa, CA, Mar. 2005.²

IEC/IEEE 62142 (IEEE Std 1364.1), Standard for Verilog Register Transfer Level Synthesis.³

IEEE Std 1076[™], IEEE Standard VHDL Language Reference Manual.^{4, 5}

IEEE Std 1076.6[™], IEEE Standard for VHDL Register Transfer Level (RTL) Synthesis.

IEEE Std 1364[™], IEEE Standard for Verilog Hardware Description Language.

IEEE Std 1666[™], IEEE Standard for the SystemC Language.

IEEE Std 1800[™], IEEE Standard for the SystemVerilog Language.

ren SystemVe. ²This document is available from the IEEE Standards World Wide Web site, at http://standards.ieee.org/downloads/1850/1850-2005/ gdl.pdf.

³IEC publications are available from the Sales Department of the International Electrotechnical Commission, Case Postale 131, 3, rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iec.ch/). IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA. ⁴IEEE 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/).

⁵The IEEE standards or products referred to in this standard are trademarks of the Institute of Electrical and Electronics Engineers, Inc.