



IEC 61691-7

Edition 1.0 2009-12

INTERNATIONAL STANDARD

IEEE Std 1666™

**Behavioural languages –
Part 7: SystemC® Language Reference Manual**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE **XH**

ICS 25.040, 35.060

ISBN 978-0-7381-6284-3

CONTENTS

Foreword.....	xi
IEEE Introduction	xvi
1. Overview.....	1
1.1 Scope.....	1
1.2 Purpose.....	1
1.3 Subsets	1
1.4 Relationship with C++.....	1
1.5 Guidance for readers	2
2. References.....	3
3. Terminology and conventions used in this standard.....	4
3.1 Terminology.....	4
3.1.1 Shall, should, may, can.....	4
3.1.2 Implementation, application.....	4
3.1.3 Call, called from, derived from	4
3.1.4 Specific technical terms	4
3.2 Syntactical conventions	6
3.2.1 Implementation-defined.....	6
3.2.2 Disabled	6
3.2.3 Ellipsis (...).....	6
3.2.4 Class names.....	6
3.2.5 Embolded text	7
3.3 Semantic conventions	7
3.3.1 Class definitions and the inheritance hierarchy	7
3.3.2 Function definitions and side-effects	7
3.3.3 Functions whose return type is a reference or a pointer.....	7
3.3.4 Namespaces and internal naming.....	9
3.3.5 Non-compliant applications and errors	9
3.4 Notes and examples	10
4. Elaboration and simulation semantics	11
4.1 Elaboration.....	11
4.1.1 Instantiation	11
4.1.2 Process macros.....	13
4.1.3 Port binding and export binding	13
4.1.4 Setting the time resolution	14
4.2 Simulation.....	14
4.2.1 The scheduling algorithm	15
4.2.2 Cycles in the scheduling algorithm.....	17
4.3 Running elaboration and simulation	18
4.3.1 Function declarations.....	18
4.3.2 Function <code>sc_elab_and_sim</code>	18
4.3.3 Functions <code>sc_argc</code> and <code>sc_argv</code>	19
4.3.4 Running under application control using functions <code>sc_main</code> and <code>sc_start</code>	19
4.3.5 Running under control of the kernel	20
4.4 Elaboration and simulation callbacks	20
4.4.1 <code>before_end_of_elaboration</code>	21
4.4.2 <code>end_of_elaboration</code>	22
4.4.3 <code>start_of_simulation</code>	23

4.4.4	end_of_simulation	23
4.5	Other functions related to the scheduler	24
4.5.1	Function declarations	24
4.5.2	Function sc_stop, sc_set_stop_mode, and sc_get_stop_mode	24
4.5.3	Function sc_time_stamp	25
4.5.4	Function sc_delta_count	26
4.5.5	Function sc_is_running	26
5.	Core language class definitions	27
5.1	Class header files	27
5.1.1	#include "systemc"	27
5.1.2	#include "systemc.h"	27
5.2	sc_module	29
5.2.1	Description	29
5.2.2	Class definition	29
5.2.3	Constraints on usage	31
5.2.4	kind	31
5.2.5	SC_MODULE	31
5.2.6	Constructors	32
5.2.7	SC_CTOR	32
5.2.8	SC_HAS_PROCESS	33
5.2.9	SC_METHOD, SC_THREAD, SC_CTHREAD	33
5.2.10	Method process	34
5.2.11	Thread and clocked thread processes	35
5.2.12	Clocked thread processes and reset_signal_is	36
5.2.13	sensitive	37
5.2.14	dont_initialize	38
5.2.15	set_stack_size	39
5.2.16	next_trigger	39
5.2.17	wait	41
5.2.18	Positional port binding	43
5.2.19	before_end_of_elaboration, end_of_elaboration, start_of_simulation, end_of_simulation	44
5.2.20	get_child_objects	45
5.2.21	sc_gen_unique_name	45
5.2.22	sc_behavior and sc_channel	46
5.3	sc_module_name	47
5.3.1	Description	47
5.3.2	Class definition	47
5.3.3	Constraints on usage	47
5.3.4	Module hierarchy	48
5.3.5	Member functions	48
5.4	sc_sensitive†	50
5.4.1	Description	50
5.4.2	Class definition	50
5.4.3	Constraints on usage	50
5.4.4	operator<<	50
5.5	sc_spawn_options and sc_spawn	52
5.5.1	Description	52
5.5.2	Class definition	52
5.5.3	Constraints on usage	53
5.5.4	Constructors	53
5.5.5	Member functions	53

5.5.6	sc_spawn	54
5.5.7	SC_FORK and SC_JOIN	56
5.6	sc_process_handle	58
5.6.1	Description	58
5.6.2	Class definition	58
5.6.3	Constraints on usage	59
5.6.4	Constructors	59
5.6.5	Member functions	59
5.6.6	sc_get_current_process_handle	61
5.7	sc_event_finder and sc_event_finder_t	62
5.7.1	Description	62
5.7.2	Class definition	62
5.7.3	Constraints on usage	62
5.8	sc_event_and_list† and sc_event_or_list†	65
5.8.1	Description	65
5.8.2	Class definition	65
5.8.3	Constraints on usage	65
5.8.4	Event lists	65
5.9	sc_event	66
5.9.1	Description	66
5.9.2	Class definition	66
5.9.3	Constraints on usage	66
5.9.4	notify and cancel	66
5.9.5	Event lists	67
5.9.6	Multiple event notifications	67
5.10	sc_time	68
5.10.1	Description	68
5.10.2	Class definition	68
5.10.3	Time resolution	69
5.10.4	Functions and operators	69
5.10.5	SC_ZERO_TIME	69
5.11	sc_port	71
5.11.1	Description	71
5.11.2	Class definition	71
5.11.3	Template parameters	72
5.11.4	Constraints on usage	73
5.11.5	Constructors	74
5.11.6	kind	74
5.11.7	Named port binding	74
5.11.8	Member functions for bound ports and port-to-port binding	75
5.11.9	before_end_of_elaboration, end_of_elaboration, start_of_simulation, end_of_simulation	79
5.12	sc_export	80
5.12.1	Description	80
5.12.2	Class definition	80
5.12.3	Template parameters	81
5.12.4	Constraints on usage	81
5.12.5	Constructors	81
5.12.6	kind	81
5.12.7	Export binding	82
5.12.8	Member functions for bound exports and export-to-export binding	83
5.12.9	before_end_of_elaboration, end_of_elaboration, start_of_simulation, end_of_simulation	84

5.13	sc_interface	85
5.13.1	Description	85
5.13.2	Class definition	85
5.13.3	Constraints on usage	85
5.13.4	register_port	86
5.13.5	default_event	86
5.14	sc_prim_channel	88
5.14.1	Description	88
5.14.2	Class definition	88
5.14.3	Constraints on usage	89
5.14.4	Constructors	89
5.14.5	kind	89
5.14.6	request_update and update	89
5.14.7	next_trigger and wait	90
5.14.8	before_end_of_elaboration, end_of_elaboration, start_of_simulation, end_of_simulation	90
5.15	sc_object	92
5.15.1	Description	92
5.15.2	Class definition	92
5.15.3	Constraints on usage	93
5.15.4	Constructors and hierarchical names	93
5.15.5	name, basename, and kind	94
5.15.6	print and dump	95
5.15.7	Functions for object hierarchy traversal	95
5.15.8	Member functions for attributes	97
5.16	sc_attr_base	99
5.16.1	Description	99
5.16.2	Class definition	99
5.16.3	Member functions	99
5.17	sc_attribute	100
5.17.1	Description	100
5.17.2	Class definition	100
5.17.3	Template parameters	100
5.17.4	Member functions and data members	100
5.18	sc_attr_cltn	101
5.18.1	Description	101
5.18.2	Class definition	101
5.18.3	Constraints on usage	101
5.18.4	Iterators	101
6.	Predefined channel class definitions	103
6.1	sc_signal_in_if	103
6.1.1	Description	103
6.1.2	Class definition	103
6.1.3	Member functions	103
6.2	sc_signal_in_if<bool> and sc_signal_in_if<sc_dt::sc_logic>	104
6.2.1	Description	104
6.2.2	Class definition	104
6.2.3	Member functions	105
6.3	sc_signal_inout_if	106
6.3.1	Description	106
6.3.2	Class definition	106
6.3.3	write	106

6.4	sc_signal	107
6.4.1	Description	107
6.4.2	Class definition	107
6.4.3	Template parameter T	108
6.4.4	Reading and writing signals	108
6.4.5	Constructors	109
6.4.6	register_port	109
6.4.7	Member functions for reading	109
6.4.8	Member functions for writing	110
6.4.9	Member functions for events	110
6.4.10	Diagnostic member functions	110
6.4.11	operator<<	111
6.5	sc_signal<bool> and sc_signal<sc_dt::sc_logic>	113
6.5.1	Description	113
6.5.2	Class definition	113
6.5.3	Member functions	114
6.6	sc_buffer	116
6.6.1	Description	116
6.6.2	Class definition	116
6.6.3	Constructors	116
6.6.4	Member functions	117
6.7	sc_clock	119
6.7.1	Description	119
6.7.2	Class definition	119
6.7.3	Characteristic properties	120
6.7.4	Constructors	120
6.7.5	write	120
6.7.6	Diagnostic member functions	120
6.7.7	before_end_of_elaboration	121
6.7.8	sc_in_clk	121
6.8	sc_in	122
6.8.1	Description	122
6.8.2	Class definition	122
6.8.3	Member functions	123
6.8.4	Function sc_trace	123
6.8.5	end_of_elaboration	123
6.9	sc_in<bool> and sc_in<sc_dt::sc_logic>	124
6.9.1	Description	124
6.9.2	Class definition	124
6.9.3	Member functions	126
6.10	sc_inout	127
6.10.1	Description	127
6.10.2	Class definition	127
6.10.3	Member functions	128
6.10.4	initialize	128
6.10.5	Function sc_trace	128
6.10.6	end_of_elaboration	129
6.10.7	Binding	129
6.11	sc_inout<bool> and sc_inout<sc_dt::sc_logic>	130
6.11.1	Description	130
6.11.2	Class definition	130
6.11.3	Member functions	132
6.12	sc_out	133
6.12.1	Description	133

6.12.2	Class definition	133
6.12.3	Member functions	133
6.13	sc_signal_resolved	134
6.13.1	Description	134
6.13.2	Class definition	134
6.13.3	Constructors	134
6.13.4	Resolution semantics	135
6.13.5	Member functions	136
6.14	sc_in_resolved	137
6.14.1	Description	137
6.14.2	Class definition	137
6.14.3	Member functions	137
6.15	sc_inout_resolved	138
6.15.1	Description	138
6.15.2	Class definition	138
6.15.3	Member functions	138
6.16	sc_out_resolved	139
6.16.1	Description	139
6.16.2	Class definition	139
6.16.3	Member functions	139
6.17	sc_signal_rv	140
6.17.1	Description	140
6.17.2	Class definition	140
6.17.3	Semantics and member functions	140
6.18	sc_in_rv	142
6.18.1	Description	142
6.18.2	Class definition	142
6.18.3	Member functions	142
6.19	sc_inout_rv	143
6.19.1	Description	143
6.19.2	Class definition	143
6.19.3	Member functions	143
6.20	sc_out_rv	145
6.20.1	Description	145
6.20.2	Class definition	145
6.20.3	Member functions	145
6.21	sc_fifo_in_if	146
6.21.1	Description	146
6.21.2	Class definition	146
6.21.3	Member functions	146
6.22	sc_fifo_out_if	148
6.22.1	Description	148
6.22.2	Class definition	148
6.22.3	Member functions	148
6.23	sc_fifo	150
6.23.1	Description	150
6.23.2	Class definition	150
6.23.3	Template parameter T	151
6.23.4	Constructors	151
6.23.5	register_port	152
6.23.6	Member functions for reading	152
6.23.7	Member functions for writing	152
6.23.8	The update phase	153
6.23.9	Member functions for events	153

6.23.10	Member functions for available values and free slots	153
6.23.11	Diagnostic member functions.....	154
6.23.12	operator<<.....	154
6.24	sc_fifo_in	156
6.24.1	Description	156
6.24.2	Class definition.....	156
6.24.3	Member functions	156
6.25	sc_fifo_out	157
6.25.1	Description	157
6.25.2	Class definition.....	157
6.25.3	Member functions	157
6.26	sc_mutex_if.....	160
6.26.1	Description.....	160
6.26.2	Class definition	160
6.26.3	Member functions.....	160
6.27	sc_mutex	161
6.27.1	Description	161
6.27.2	Class definition	161
6.27.3	Constructors	161
6.27.4	Member functions	162
6.28	sc_semaphore_if	163
6.28.1	Description	163
6.28.2	Class definition	163
6.28.3	Member functions	163
6.29	sc_semaphore.....	164
6.29.1	Description	164
6.29.2	Class definition	164
6.29.3	Constructors	164
6.29.4	Member functions	165
6.30	sc_event_queue.....	166
6.30.1	Description	166
6.30.2	Class definition.....	166
6.30.3	Constraints on usage.....	166
6.30.4	Constructors	166
6.30.5	kind.....	167
6.30.6	Member functions	167
7.	Data types	169
7.1	Introduction.....	169
7.2	Common characteristics.....	172
7.2.1	Initialization and assignment operators.....	172
7.2.2	Precision of arithmetic expressions.....	173
7.2.3	Base class default word length	174
7.2.4	Word length.....	175
7.2.5	Bit-select.....	175
7.2.6	Part-select	175
7.2.7	Concatenation.....	176
7.2.8	Reduction operators.....	178
7.2.9	Integer conversion	178
7.2.10	String input and output.....	178
7.2.11	Conversion of application-defined types in integer expressions.....	179
7.3	String literals.....	180
7.4	sc_value_base†	182

7.4.1	Description	182
7.5	Limited-precision integer types	184
7.5.1	Type definitions	184
7.5.2	sc_int_base	185
7.5.3	sc_uint_base	190
7.5.4	sc_int	194
7.5.5	sc_uint	197
7.5.6	Bit-selects	199
7.5.7	Part-selects	203
7.6	Finite-precision integer types	209
7.6.1	Type definitions	209
7.6.2	Constraints on usage	209
7.6.3	sc_signed	209
7.6.4	sc_unsigned	216
7.6.5	sc_bigint	223
7.6.6	sc_biguint	225
7.6.7	Bit-selects	227
7.6.8	Part-selects	231
7.7	Integer concatenations	236
7.7.1	Description	236
7.7.2	Class definition	236
7.7.3	Constraints on usage	237
7.7.4	Assignment operators	238
7.7.5	Implicit type conversion	238
7.7.6	Explicit type conversion	238
7.7.7	Other member functions	238
7.8	Generic base proxy class	239
7.8.1	Description	239
7.8.2	Class definition	239
7.8.3	Constraints on usage	239
7.9	Logic and vector types	240
7.9.1	Type definitions	240
7.9.2	sc_logic	240
7.9.3	sc_bv_base	244
7.9.4	sc_lv_base	251
7.9.5	sc_bv	257
7.9.6	sc_lv	259
7.9.7	Bit-selects	261
7.9.8	Part-selects	264
7.9.9	Concatenations	270
7.10	Fixed-point types	278
7.10.1	Fixed-point representation	278
7.10.2	Fixed-point type conversion	279
7.10.3	Fixed-point data types	279
7.10.4	Fixed-point expressions and operations	281
7.10.5	Bit and part selection	284
7.10.6	Variable-precision fixed-point value limits	284
7.10.7	Fixed-point word length and mode	284
7.10.8	Conversions to character string	287
7.10.9	Finite word-length effects	289
7.10.10	sc_fxnum	313
7.10.11	sc_fxnum_fast	317
7.10.12	sc_fxval	322
7.10.13	sc_fxval_fast	326

7.10.14	sc_fix	331
7.10.15	sc_ufix	334
7.10.16	sc_fix_fast.....	337
7.10.17	sc_ufix_fast.....	340
7.10.18	sc_fixed.....	343
7.10.19	sc_ufixed.....	345
7.10.20	sc_fixed_fast.....	347
7.10.21	sc_ufixed_fast.....	350
7.10.22	Bit-selects	352
7.10.23	Part-selects.....	354
7.11	Contexts	361
7.11.1	sc_length_param.....	361
7.11.2	sc_length_context.....	362
7.11.3	sc_fxtype_params.....	363
7.11.4	sc_fxtype_context.....	366
7.11.5	sc_fxcast_switch.....	367
7.11.6	sc_fxcast_context	368
7.12	Control of string representation	370
7.12.1	Description	370
7.12.2	Class definition	370
7.12.3	Functions.....	370
8.	Utility class definitions	371
8.1	Trace files	371
8.1.1	Class definition and function declarations	371
8.1.2	sc_trace_file	371
8.1.3	sc_create_vcd_trace_file	372
8.1.4	sc_close_vcd_trace_file	372
8.1.5	sc_write_comment	372
8.1.6	sc_trace.....	372
8.2	sc_report.....	375
8.2.1	Description	375
8.2.2	Class definition	375
8.2.3	Constraints on usage	375
8.2.4	sc_severity.....	376
8.2.5	Copy constructor and assignment	376
8.2.6	Member functions	376
8.3	sc_report_handler.....	378
8.3.1	Description	378
8.3.2	Class definition.....	378
8.3.3	Constraints on usage.....	380
8.3.4	sc_actions	380
8.3.5	report	380
8.3.6	set_actions	381
8.3.7	stop_after.....	381
8.3.8	get_count	382
8.3.9	suppress and force	383
8.3.10	set_handler	383
8.3.11	get_new_action_id	384
8.3.12	sc_interrupt_here and sc_stop_here	384
8.3.13	get_cached_report and clear_cached_report	384
8.3.14	set_log_file_name and get_log_file_name.....	385
8.4	sc_exception.....	386

8.4.1	Description	386
8.4.2	Definition	386
8.5	Utility functions	387
8.5.1	Function declarations	387
8.5.2	sc_abs	387
8.5.3	sc_max	387
8.5.4	sc_min	387
8.5.5	sc_copyright	387
8.5.6	sc_version	388
8.5.7	sc_release	388
Annex A (informative) Introduction to SystemC		389
Annex B (informative) Glossary		393
Annex C (informative) Deprecated features		403
Annex D (informative) Changes between the different SystemC versions		405
Annex E (informative) List of Participants		407
Index		409

INTERNATIONAL ELECTROTECHNICAL COMMISSION

BEHAVIOURAL LANGUAGES –

Part 7: SystemC[®] Language Reference Manual

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61691-7/IEEE Std 1666 has been processed through IEC technical committee 93: *Design automation*.

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

IEEE Std	FDIS	Report on voting
1666 (2005)	93/279/FDIS	93/285/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.

A list of parts of the IEC 61691 series can be found on the IEC web site.

The committee has decided that the contents of this publication will remain unchanged until the maintenance result 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,
- replaced by a revised edition, or
- amended.

IEC/IEEE Dual Logo International Standards

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

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

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

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

The existence of an IEC/IEEE Dual Logo International Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEC/IEEE Dual Logo International Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

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

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

Interpretations – Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

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

Secretary, IEEE-SA Standards Board, 445 Hoes Lane, Piscataway, NJ 08854, USA and/or General Secretary, IEC, 3, rue de Varembé, PO Box 131, 1211 Geneva 20, Switzerland.

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

NOTE – Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

IEEE Standard SystemC[®] Language Reference Manual

Sponsor

Design Automation Standards Committee
of the
IEEE Computer Society

Approved 28 March 2006

American National Standards Institute

Approved 6 December 2005

IEEE-SA Standards Board

Grateful acknowledgment is made to Open SystemC Initiative for the permission to use the following source material:
SystemC[®] Language Reference Manual Version 2.1

Abstract: SystemC^{®1} is defined in this standard. SystemC is an ANSI standard C++ class library for system and hardware design for use by designers and architects who need to address complex systems that are a hybrid between hardware and software. This standard provides a precise and complete definition of the SystemC class library so that a SystemC implementation can be developed with reference to this standard alone. The primary audiences for this standard are the implementors of the SystemC class library, the implementors of tools supporting the class library, and users of the class library.

Keywords: C++, computer languages, digital systems, discrete event simulation, electronic design automation, electronic systems, electronic system level, embedded software, fixed-point, hardware description language, hardware design, hardware verification, SystemC, system modeling, system-on-chip, transaction level

¹SystemC[®] is a registered trademark of Open SystemC Initiative.

IEEE introduction

This document defines SystemC, which is a C++ class library.

As the electronics industry builds more complex systems involving large numbers of components including software, there is an increasing need for a modeling language that can manage the complexity and size of these systems. SystemC provides a mechanism for managing this complexity with its facility for modeling hardware and software together at multiple levels of abstraction. This capability is not available in traditional hardware description languages.

Stakeholders in SystemC include Electronic Design Automation (EDA) companies who implement SystemC class libraries and tools, Integrated Circuit (IC) suppliers who extend those class libraries and use SystemC to model their intellectual property, and end users who use SystemC to model their systems.

Before the publication of this standard, SystemC was defined by an open source proof-of-concept C++ library, also known as *the reference simulator*, available from the Open SystemC Initiative (OSCI). In the event of discrepancies between the behavior of the reference simulator and statements made in this standard, this standard shall be taken to be definitive.

This standard is not intended to serve as a users' guide or to provide an introduction to SystemC. Readers requiring a SystemC tutorial or information on the intended use of SystemC should consult the OSCI Web site (www.systemc.org) to locate the many books and training classes available.

Notice to users

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 standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

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>.

BEHAVIOURAL LANGUAGES –

Part 7: SystemC[®] Language Reference Manual

1. Overview

1.1 Scope

This standard defines SystemC[®]¹ as an ANSI standard C++ class library for system and hardware design.

1.2 Purpose

The general purpose of SystemC is to provide a C++-based standard for designers and architects who need to address complex systems that are a hybrid between hardware and software.

The specific purpose of this standard is to provide a precise and complete definition of the SystemC class library so that a SystemC implementation can be developed with reference to this standard alone. This standard is not intended to serve as a users' guide or to provide an introduction to SystemC, but does contain useful information for end users.

1.3 Subsets

It is anticipated that tool vendors will create implementations that support only a subset of this standard or that impose further constraints on the use of this standard. Such implementations are not fully compliant with this standard but may nevertheless claim partial compliance with this standard and may use the name SystemC.

1.4 Relationship with C++

This standard is closely related to the C++ programming language and adheres to the terminology used in ISO/IEC 14882:2003. This standard does not seek to restrict the usage of the C++ programming language; a SystemC application may use any of the facilities provided by C++, which in turn may use any of the facilities provided by C. However, where the facilities provided by this standard are used, they shall be used in accordance with the rules and constraints set out in this standard.

This standard defines the public interface to the SystemC class library and the constraints on how those classes may be used. The SystemC class library may be implemented in any manner whatsoever, provided only that the obligations imposed by this standard are honored.

A C++ class library may be extended using the mechanisms provided by the C++ language. Implementors and users are free to extend SystemC in this way, provided that they do not violate this standard.

¹SystemC[®] is a registered trademark of Open SystemC Initiative.

NOTE—It is possible to create a well-formed C++ program that is legal according to the C++ programming language standard but that violates this standard. An implementation is not obliged to detect every violation of this standard.²

1.5 Guidance for readers

Readers who are not entirely familiar with SystemC should start with Annex A, “Introduction to SystemC,” which provides a brief informal summary of the subject intended to aid in the understanding of the normative definitions. Such readers may also find it helpful to scan the examples embedded in the normative definitions and to see Annex B, “Glossary.”

Readers should pay close attention to Clause 3, “Terminology and conventions used in this standard.” An understanding of the terminology defined in Clause 3 is necessary for a precise interpretation of this standard.

Clause 4, “Elaboration and simulation semantics,” defines the behavior of the SystemC kernel and is central to an understanding of SystemC. The semantic definitions given in the subsequent clauses detailing the individual classes are built upon the foundations laid in Clause 4.

The clauses from Clause 5 onward define the public interface to the SystemC class library. The following information is listed for each class:

- a) A C++ source code listing of the class definition
- b) A statement of any constraints on the use of the class and its members
- c) A statement of the semantics of the class and its members
- d) For certain classes, a description of functions, typedefs, and macros associated with the class.
- e) Informative examples illustrating both typical and atypical uses of the class

Readers should bear in mind that the primary obligation of a tool vendor is to implement the abstract semantics defined in Clause 4, using the framework and constraints provided by the class definitions starting in Clause 5.

Annex A is intended to aid the reader in the understanding of the structure and intent of the SystemC class library.

Annex B is a glossary giving informal descriptions of the terms used in this standard.

Annex C lists the deprecated features, that is, features that were present in version 2.0.1 of the Open SystemC Initiative (OSCI) open source proof-of-concept SystemC implementation but are not part of this standard.

Annex D lists the changes between SystemC version 2.0.1 and version 2.1 Beta Oct 12 2004, and the changes between SystemC 2.1 Beta Oct 12 2004 and this standard.

²Notes in text, tables, and figures are given for information only, and do not contain requirements needed to implement the standard.

2. References

The following documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the document (including any amendments or corrigenda) applies.

This standard shall be used in conjunction with the following publications:

ISO/IEC 14882:2003, Programming Languages—C++.³

IEC 61691-4:2004, Behavioural languages - Part 4: Verilog® hardware description language | IEEE Std 1364™-2001, IEEE Standard Verilog® Hardware Description Language.^{4, 5, 6}

³IEC publications are available from the Sales Department of The International Electrotechnical Commission, Case Postale 131, 3, rue de Varembe, CH-1211, Genève 20, Switzerland/Suisse (<http://www.iec.ch>). ISO publications are available from the ISO Central Secretariat, 1 chemin de la Voie-Creuse, CP 56, CH-1211, Genève 20, Switzerland/Suisse (<http://www.iso.ch>). ISO/IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (<http://www.ansi.org/>).

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 clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

⁶IEEE Std 1364-2001 was adopted as IEC 61691-4:2004