INTERNATIONAL STANDARD

IEC 61691-3-2

First edition
2001-06

Behavioural languages –

Part 3-2: Mathematical operation in VHDL

Reference number
IEC 61691-3-2:2001(E)

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology. Information relating to this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site (www.iec.ch)**

- **Catalogue of IEC publications**

  The on-line catalogue on the IEC web site (www.iec.ch/catlg-e.htm) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.

- **IEC Just Published**

  This summary of recently issued publications (www.iec.ch/JP.htm) is also available by email. Please contact the Customer Service Centre (see below) for further information.

- **Customer Service Centre**

  If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

  Email: custserv@iec.ch
  Tel: +41 22 919 02 11
  Fax: +41 22 919 03 00

INTERNATIONAL STANDARD

IEC 61691-3-2

First edition
2001-06

Behavioural languages –

Part 3-2: Mathematical operation in VHDL

International Electrotechnical Commission 3, rue de Varembé Geneva, Switzerland
Telefax: +41 22 919 0300 e-mail: inmail@iec.ch IEC web site http://www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE W

For price, see current catalogue

INTERNATIONAL ELECTROTECHNICAL COMMISSION

BEHAVIOURAL LANGUAGES –

Part 3-2: Mathematical operation in VHDL

FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the 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, the IEC publishes International Standards. 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. The 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 the 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 National Committees.
3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.
4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.
6) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61691-3-2 has been prepared by IEC technical committee 93: Design automation.

This standard is based on IEEE Std 1076-2 (1996): *IEEE Standard VHDL mathematical packages.*

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

FDIS	Report on voting
93/131/FDIS	93/141/RVD

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

This standard does not follow the rules for the structure of international standards given in Part 3 of the ISO/IEC Directives.

IEC 61691 consists of the following parts, under the general title: *Behavioural languages:*

IEC 61691-1:1997, VHDL language reference manual [1)]

IEC 61691-2:2001, Part 2: VHDL multilogic system for model interoperability

———————

1) The edition 2 with the title: VHSIC hardware description language VHDL (1076a) (under consideration) will replace it.

IEC 61691-3-1, Part 3-1: Analog description in VHDL (under consideration)

IEC 61691-3-2:2001, Part 3-2: Mathematical operation in VHDL

IEC 61691-3-3:2001, Part 3-3: Synthesis in VHDL

IEC 61691-3-4, Part 3-4: Timing expressions in VHDL (under consideration)

IEC 61691-3-5, Part 3-5: Library utilities in VHDL (under consideration)

The committee has decided that the contents of this publication will remain unchanged until 2004. At this date, the publication will be

- reconfirmed;
- withdrawn;
- replaced by a revised edition, or
- amended.

INTRODUCTION

This set of packages provides a standard for the declaration of most frequently used real and complex elementary functions required for numerically oriented modeling applications. Use of these packages with their defined data types, constants, and functions is intended to provide a mechanism for writing VHDL models (compliant with IEEE Std 1076-1993) that are portable and interoperable with other VHDL models adhering to this standard. The standard serves a broad class of applications with reasonable ease of use and requires implementations that are of high quality.

This standard includes package bodies, as described in annex B, which are available in electronic format either on a diskette affixed to the back cover, or as a downloadable file from the IEC Web Store.

BEHAVIOURAL LANGUAGES -
Part 3-2: Mathematical operation in VHDL

1. Overview

1.1 Scope

This standard is embodied in the MATH_REAL and MATH_COMPLEX package declarations, and in the semantics of the standard mathematical definition and the conventional meaning of the functions that are part of this standard, along with 1.3. The information in annex A is a guide to users and implementors and is not a normative part of this standard, but suggests ways in which one might use this set of packages. The information in annex B is provided as a guideline for implementors and is not a normative part of this standard, but suggests ways in which implementors may implement this standard. The functions in this set of packages were chosen because of their widespread utility, as well as because they are needed to support general floating-point usage and to build other generic packages.

1.2 Constants, types, and functions provided

The following constants of type REAL are provided:

MATH_E	MATH_LOG_OF_2	MATH_DEG_TO_RAD
MATH_1_OVER_E	MATH_LOG_OF_10	MATH_RAD_TO_DEG
	MATH_LOG2_OF_E	
MATH_PI	MATH_LOG10_OF_E	
MATH_2_PI		
MATH_1_OVER_PI	MATH_SQRT_2	
MATH_PI_OVER_2	MATH_1_OVER_SQRT_2	
MATH_PI_OVER_3	MATH_SQRT_PI	
MATH_PI_OVER_4		
MATH_3_PI_OVER_2		

The following functions/procedures of type REAL are provided:

SIGN(X)	EXP(X)	SINH(X)
CEIL(X)	LOG(X)	COSH(X)
FLOOR(X)	LOG2(X)	TANH(X)
ROUND(X)	LOG10(X)	
TRUNC(X)	LOG(X, BASE)	ARCSINH(X)
"MOD"(X, Y)		ARCCOSH(X)
	SIN(X)	ARCTANH(X)
REALMAX(X, Y)	COS(X)	
REALMIN(X, Y)	TAN(X)	
UNIFORM(SEED1, SEED2, X)		
SQRT(X)	ARCSIN(X)	
CBRT(X)	ARCCOS(X)	
"**"(X, Y)	ARCTAN(Y)	
	ARCTAN(Y, X)	

The following types and subtypes are provided:

COMPLEX	POSITIVE_REAL
COMPLEX_POLAR	PRINCIPAL_VALUE

The following constants of type COMPLEX are provided:

MATH_CBASE_1	MATH_CBASE_J	MATH_CZERO

The following type conversion functions for COMPLEX and COMPLEX_POLAR are provided:

CMPLX(X, Y)	POLAR_TO_COMPLEX(Z)
COMPLEX_TO_POLAR(Z)	GET_PRINCIPAL_VALUE(X)

The following overloaded relational functions for type COMPLEX_POLAR are provided:

"="(L, R)	"/="(L, R)

The following functions for type COMPLEX and COMPLEX_POLAR are provided:

"ABS"(Z)	EXP(Z)	SIN(Z)
ARG(Z)	LOG(Z)	COS(Z)
	LOG2(Z)	
"-"(Z)	LOG10(Z)	SINH(Z)
CONJ(Z)	LOG(Z, BASE)	COSH(Z)
SQRT(Z)		

The following arithmetic functions for type COMPLEX and COMPLEX_POLAR are provided:

"+"	"*"
"-"	"/"

1.3 Conformance with this standard

The following conformance rules shall apply as they pertain to the use and implementation of this standard:

a) The package declarations may be modified to include additional data required by tools, but modifications shall in no way change the external interfaces or simulation behavior of the description. It is permissible to add comments and/or attributes to the package declarations, but not to change or delete any original lines of the approved package declaration.

b) The standard mathematical definition and conventional meaning of the mathematical functions that are part of this standard, together with the MATH_REAL and MATH_COMPLEX package declarations, represent the formal semantics of the implementation of the MATH_REAL and MATH_COMPLEX packages. An implementation is provided as a guideline in annex B. Implementors of these packages may choose to simply compile the package bodies provided in annex B, or they may choose to implement the package bodies in the most efficient form available to them. Implementations should conform to the semantics and minimum precision required by this standard.

c) The MATH_REAL package shall be built on top of the standard data type and precision requirements for floating point operations defined in IEEE Std 1076-1993 (STD.STANDARD).

d) The minimum precision required is that of IEEE Std 1076-1993. Because of this reason and the fact that the functions are implemented on digital computers with only finite precision, the functions and constants in this set of packages can, at best, only approximate the corresponding mathematically defined functions and constants. An implementation is allowed to provide a higher precision than the minimum required.

e) For some functions, the implementation shall deliver “prescribed results” for certain special arguments, as defined in the comments for the functions in the function declaration. The purpose is to strengthen the accuracy requirements at special argument values. Prescribed results take precedence over maximum relative error requirements.

f) The semantics of the standard require that all the functions in the packages detect and report invalid parameters (out of valid domain) through an assert statement. The domain of valid values is indicated in the MATH_REAL and MATH_COMPLEX package declarations. The default value of the severity level shall be Error.

g) The semantics of the standard do not require detection of overflow or underflow. Therefore, detection of underflow/overflow is optional and implementation dependent.

h) If an implementation chooses to provide any extensions beyond the minimum requirements of this standard (e.g., precision, overflow handling), then it shall document its behavior accordingly.

i) The MATH_REAL and MATH_COMPLEX packages shall be compiled into a library symbolically named IEEE.

2. References

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

IEEE Std 754-1985 (Reaff 1990), IEEE Standard for Binary Floating-Point Arithmetic (ANSI).[1]

IEEE Std 1076-1993, IEEE Standard VHDL Language Reference Manual (ANSI).

[1] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P. O. Box 1331, Piscataway, NJ 08855-1331, USA.

3. Package declarations

The declaration of each function includes the following information: description of the mathematical definition of the function; values to be returned by the function for special arguments; valid domain of values for the input arguments; error conditions; range of values into which the function maps the values in its domain; and notes on special accuracy situations, reachable values, usable domains, or algorithms to be used by an implementation.

3.1 MATH_REAL

```
------------------------------------------------------------------------
--
-- Copyright 1996 by IEEE. All rights reserved.
--
-- This source file is an essential part of IEEE Std 1076.2-1996, IEEE Standard
-- VHDL Mathematical Packages. This source file may not be copied, sold, or
-- included with software that is sold without written permission from the IEEE
-- Standards Department. This source file may be used to implement this standard
-- and may be distributed in compiled form in any manner so long as the
-- compiled form does not allow direct decompilation of the original source file.
-- This source file may be copied for individual use between licensed users.
-- This source file is provided on an AS IS basis. The IEEE disclaims ANY
-- WARRANTY EXPRESS OR IMPLIED INCLUDING ANY WARRANTY OF MERCHANTABILITY
-- AND FITNESS FOR USE FOR A PARTICULAR PURPOSE. The user of the source
-- file shall indemnify and hold IEEE harmless from any damages or liability
-- arising out of the use thereof.
--
-- Title:       Standard VHDL Mathematical Packages (IEEE Std 1076.2-1996,
--              MATH_REAL)
--
-- Library:     This package shall be compiled into a library
--              symbolically named IEEE.
--
-- Developers:  IEEE DASC VHDL Mathematical Packages Working Group
--
-- Purpose:     This package defines a standard for designers to use in
--              describing VHDL models that make use of common REAL constants
--              and common REAL elementary mathematical functions.
--
-- Limitation:  The values generated by the functions in this package may
--              vary from platform to platform, and the precision of results
--              is only guaranteed to be the minimum required by IEEE Std 1076-
--              1993.
--
-- Notes:
--              No declarations or definitions shall be included in, or
--              excluded from, this package.
--              The "package declaration" defines the types, subtypes, and
--              declarations of MATH_REAL.
--              The standard mathematical definition and conventional meaning
--              of the mathematical functions that are part of this standard
--              represent the formal semantics of the implementation of the
--              MATH_REAL package declaration.  The purpose of the MATH_REAL
--              package body is to provide a guideline for implementations to
--              verify their implementation of MATH_REAL.  Tool developers may
--              choose to implement the package body in the most efficient
--              manner available to them.
```

```
--
-- -----------------------------------------------------------------------------
-- Version    : 1.5
-- Date       : 24 July 1996
-- -----------------------------------------------------------------------------

package MATH_REAL is
    constant CopyRightNotice: STRING
      := "Copyright 1996 IEEE. All rights reserved.";

    --
    -- Constant Definitions
    --
    constant  MATH_E : REAL := 2.71828_18284_59045_23536;
                                                        -- Value of e
    constant  MATH_1_OVER_E : REAL := 0.36787_94411_71442_32160;
                                                        -- Value of 1/e
    constant  MATH_PI : REAL := 3.14159_26535_89793_23846;
                                                        -- Value of pi
    constant  MATH_2_PI : REAL := 6.28318_53071_79586_47693;
                                                        -- Value of 2*pi
    constant  MATH_1_OVER_PI : REAL := 0.31830_98861_83790_67154;
                                                        -- Value of 1/pi
    constant  MATH_PI_OVER_2 : REAL := 1.57079_63267_94896_61923;
                                                        -- Value of pi/2
    constant  MATH_PI_OVER_3 : REAL := 1.04719_75511_96597_74615;
                                                        -- Value of pi/3
    constant  MATH_PI_OVER_4 : REAL := 0.78539_81633_97448_30962;
                                                        -- Value of pi/4
    constant  MATH_3_PI_OVER_2 : REAL := 4.71238_89803_84689_85769;
                                                        -- Value 3*pi/2
    constant  MATH_LOG_OF_2 : REAL := 0.69314_71805_59945_30942;
                                                        -- Natural log of 2
    constant  MATH_LOG_OF_10 : REAL := 2.30258_50929_94045_68402;
                                                        -- Natural log of 10
    constant  MATH_LOG2_OF_E : REAL := 1.44269_50408_88963_4074;
                                                        -- Log base 2 of e
    constant  MATH_LOG10_OF_E: REAL := 0.43429_44819_03251_82765;
                                                        -- Log base 10 of e
    constant  MATH_SQRT_2: REAL := 1.41421_35623_73095_04880;
                                                        -- square root of 2
    constant  MATH_1_OVER_SQRT_2: REAL := 0.70710_67811_86547_52440;
                                                        -- square root of 1/2
    constant  MATH_SQRT_PI: REAL := 1.77245_38509_05516_02730;
                                                        -- square root of pi
    constant  MATH_DEG_TO_RAD: REAL := 0.01745_32925_19943_29577;
                                 -- Conversion factor from degree to radian
    constant  MATH_RAD_TO_DEG: REAL := 57.29577_95130_82320_87680;
                                 -- Conversion factor from radian to degree

    --
    -- Function Declarations
    --
    function SIGN (X: in REAL ) return REAL;
        -- Purpose:
        --         Returns 1.0 if X > 0.0; 0.0 if X = 0.0; -1.0 if X < 0.0
        -- Special values:
        --         None
        -- Domain:
```

```
    --          X in REAL
    -- Error conditions:
    --          None
    -- Range:
    --          ABS(SIGN(X)) <= 1.0
    -- Notes:
    --          None

function CEIL (X : in REAL ) return REAL;
    -- Purpose:
    --          Returns smallest INTEGER value (as REAL) not less than X
    -- Special values:
    --          None
    -- Domain:
    --          X in REAL
    -- Error conditions:
    --          None
    -- Range:
    --          CEIL(X) is mathematically unbounded
    -- Notes:
    --          a) Implementations have to support at least the domain
    --                 ABS(X) < REAL(INTEGER'HIGH)

function FLOOR (X : in REAL ) return REAL;
    -- Purpose:
    --          Returns largest INTEGER value (as REAL) not greater than X
    -- Special values:
    --          FLOOR(0.0) = 0.0
    -- Domain:
    --          X in REAL
    -- Error conditions:
    --          None
    -- Range:
    --          FLOOR(X) is mathematically unbounded
    -- Notes:
    --          a) Implementations have to support at least the domain
    --                 ABS(X) < REAL(INTEGER'HIGH)

function ROUND (X : in REAL ) return REAL;
    -- Purpose:
    --          Rounds X to the nearest integer value (as real). If X is
    --          halfway between two integers, rounding is away from 0.0
    -- Special values:
    --          ROUND(0.0) = 0.0
    -- Domain:
    --          X in REAL
    -- Error conditions:
    --          None
    -- Range:
    --          ROUND(X) is mathematically unbounded
    -- Notes:
    --          a) Implementations have to support at least the domain
    --                 ABS(X) < REAL(INTEGER'HIGH)

function TRUNC (X : in REAL ) return REAL;
    -- Purpose:
    --          Truncates X towards 0.0 and returns truncated value
    -- Special values:
    --          TRUNC(0.0) = 0.0
```