PD IEC/TR 63051:2017

BSI Standards Publication

Documentation on design automation subjects — Mathematical algorithm hardware description languages for system level modeling and verification (HDLMath)

bsi.

National foreword

This Published Document is the UK implementation of IEC/TR 63051:2017.

The UK participation in its preparation was entrusted to Technical Committee EPL/501, Electronic Assembly Technology.

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

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

© The British Standards Institution 2017.
Published by BSI Standards Limited 2017

ISBN 978 0 580 93916 7
ICS 25.040.01; 35.240.50

Compliance with a British Standard cannot confer immunity from legal obligations.

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 31 January 2017.

Amendments/corrigenda issued since publication

Date	Text affected

IEC TR 63051

Edition 1.0 2017-01

TECHNICAL REPORT

colour inside

Documentation on design automation subjects – Mathematical algorithm hardware description languages for system level modeling and verification (HDLMath)

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.01; 35.240.50

ISBN 978-2-8322-3772-4

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

® Registered trademark of the International Electrotechnical Commission

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

DOCUMENTATION ON DESIGN AUTOMATION SUBJECTS – MATHEMATICAL ALGORITHM HARDWARE DESCRIPTION LANGUAGES FOR SYSTEM LEVEL MODELING AND VERIFICATION (HDLMath)

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) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees 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 Publication or any other IEC 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 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.

The main task of IEC technical committees is to prepare International Standards. However, a technical committee may propose the publication of a Technical Report when it has collected data of a different kind from that which is normally published as an International Standard, for example "state of the art".

IEC 63051, which is a Technical Report, has been prepared by IEC technical committee 91: Electronics assembly technology.

The text of this Technical Report is based on the following documents:

Enquiry draft	Report on voting
91/1349/DTR	91/1396/RVC

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

This document has been drafted in accordance with the ISO/IEC Directives, Part 2.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document will be

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

A bilingual version of this publication may be issued at a later date.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

Around the world, engineers in industries such as electronics and automobiles are developing many kinds of systems and products. However, these are developed based on conventional design processes and suffer from many design problems and long design times. Because the laws of nature can be expressed mathematically, mathematics is a good algorithmic method for the description and modeling of such systems. Mathematical modeling is also an important approach for both solving problems and visualizing the abstract concepts involved.

System LSI (Large Scale Integration) can be described at three levels of complexity as follows:

1) The the algorithmic level, which specifies only the algorithm used by the hardware for the problem solution;
2) the register transfer level, in which the registers are system elements and the data transfer between these registers is specified according to some rule;
3) the circuit level, where gates and flip-flops are replaced by the circuit elements such as transistors, diodes, resistors, etc.

For levels 2) and 3), VHDL (IEC 61691-1-1:2011 [1][1]) and SystemVerilog (IEC 62530:2011[2]) have already been standardized by the IEC and IEEE and have been in practical use for over twenty years.

For level 1), System C is able to describe hardware systems at the behavioral level.

The purpose of this document is to accelerate the standardization of a mathematical algorithm description language (HDLMath). HDLMath will be used to describe and verify the entire behavior of systems and/or products using mathematical algorithms of electronic systems. It is a higher level language than conventional HDL (Hardware Description Language) languages such as VHDL and SystemVerilog.

HDLMath and its design environment can support the design of many domains and applications as indicated in Table 1.

Table 1 – Examples of mathematics applications

Mathematics	Application examples
Complex numbers	Resistors, inductors, capacitors, power engineering, analysis of electric and magnetic fields, digital signal processing, image processing
Matrices and determinants	Electrical networks, computer graphics, image analysis
Laplace transforms	Circuits, power systems (generators), feedback loops
Statistics and probability	Failure rates for semiconductor devices, behavior of semiconductor materials, image analysis, data compression, digital communications techniques, error correction
Vector and trigonometry	Oscillating waves (circuits, signal processing), electric and magnetic fields, design of power generating equipment, radio frequency (RF) systems and antenna design
Differentiation and integration	Calculation of currents in a circuit, wave propagation, design of semiconductors, image analyses, design of firing circuits
Functions, polynomial, linear equations, logarithms, Euclidean geometry	Curve fitting, fuel cell design, traffic modeling, power analysis, stress analysis, determining the size and shape of parts, software design, computer graphics

[1] Numbers in square brackets refer to the Bibliography.

Recently, several HDLMath languages have already been used to design the mathematical algorithms in electronic systems. MATLAB/SIMULINK is one such popular design environment for the design and verification of various system behaviors. FinSimMath has been proposed and put to practical use by several groups to design and verify mathematical algorithms in ASIC (Application Specific Integrated Circuit) or FPGA (Field Programmable Gate Array). System C-AMS is mainly for analog circuit design and is an extension of the System C standardized by the IEEE and IEC. It is capable of describing mathematical algorithms using additional C-code extensions. IEC TR 62856:2013 [3] (BVDL, or Bird's-eye View of Design Languages) describes the features of existing design languages, as well as listing the requirements for enhancing design languages and for developing new ones.

Another purpose of this document is to add HDLMath to BVDL as a system modeling language. This document describes nine functional requirements for an HDLMath and compares current HDLMath languages from a design viewpoint. It is intended to accelerate the standardization of a mathematical algorithm design language and to establish a good system modeling environment in the world.

DOCUMENTATION ON DESIGN AUTOMATION SUBJECTS – MATHEMATICAL ALGORITHM HARDWARE DESCRIPTION LANGUAGES FOR SYSTEM LEVEL MODELING AND VERIFICATION (HDLMath)

1 Scope

A hardware description language provides a means to describe the behavior of a system precisely and concisely. This document describes the main functional requirements for an HDLMath language and compares existing HDLMath languages from the viewpoint of designers. It is intended to accelerate the standardization of a mathematical algorithm design language and to help establish a new and good system modeling and verification environment.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

There are no normative references in this document.

3 Terms and definitions

No terms and definitions are listed in this document.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

4 Definition and positioning of HDLMath

4.1 General

HDLMath is defined as a language for describing and verifying the behavior of an entire system or product using mathematical algorithms.

IEC TR 62856:2013 (BVDL) describes the features of existing design languages used in the design processes applied to the development of System-on-Chip (SoC) devices, which range from system level design, IP block creation and analog block design, to SoC design implementation and verification. HDLMath will cover system level design in the BVDL schema.

4.2 Current HDLMaths

Currently, there are three kinds of language for these design environments: HDLMath1, HDLMath2, and HDLMath3.

HDLMath1 is a kind of high-level language that has an interactive environment for numerical computation, visualization, and programming. It is able to analyze data, develop algorithms, and create models and applications using the language, tools, and built-in mathematical functions. It features the following:

a) a block diagram environment for multi-domain simulation and model-based design;

b) simulation, automatic code generation, and continuous test and verification of embedded systems.

HDLMath2 is motivated by the need for mathematical modeling within the Verilog language. Its features are as follows:

- no explicit conversion functions are necessary;
- support for runtime changes of formats, including the number of bits of the various fields;
- data in multi-dimensional arrays that are easy to access globally.

The language is designed to support a large number of mathematical system tasks, and provides access to information regarding the occurrence of overflows, underflows, maximum number of bits needed, cumulative error, etc.

HDLMath3 is a language mainly to support analog design. It allows networks of analog parts such as resistors, capacitors, etc., to be defined. The simulator extracts the differential equations corresponding to the network of analog parts and solves them based on initial conditions and using a timestep provided by the user. It is able to handle blocks that are modeled mathematically and written at the C/C++ level. However, the mathematical capabilities in math.h (a kind of C function library) are limited at the low level of C/C++ and not at the high levels found in HDLMath1 or HDLMath2.

4.3 Design abstraction level of HDLMath

Figure 1 shows the number of lines of code for several small examples written using HDLMath1 and HDLMath2. It also shows the length of the C-code generated automatically from an HDLMath1 description. The number of lines of C-code is several hundred times larger than that of the HDLMath descriptions. The figure indicates how HDLMath languages can be used to design at a higher level of design abstraction and hence how design productivity is higher than C level design.

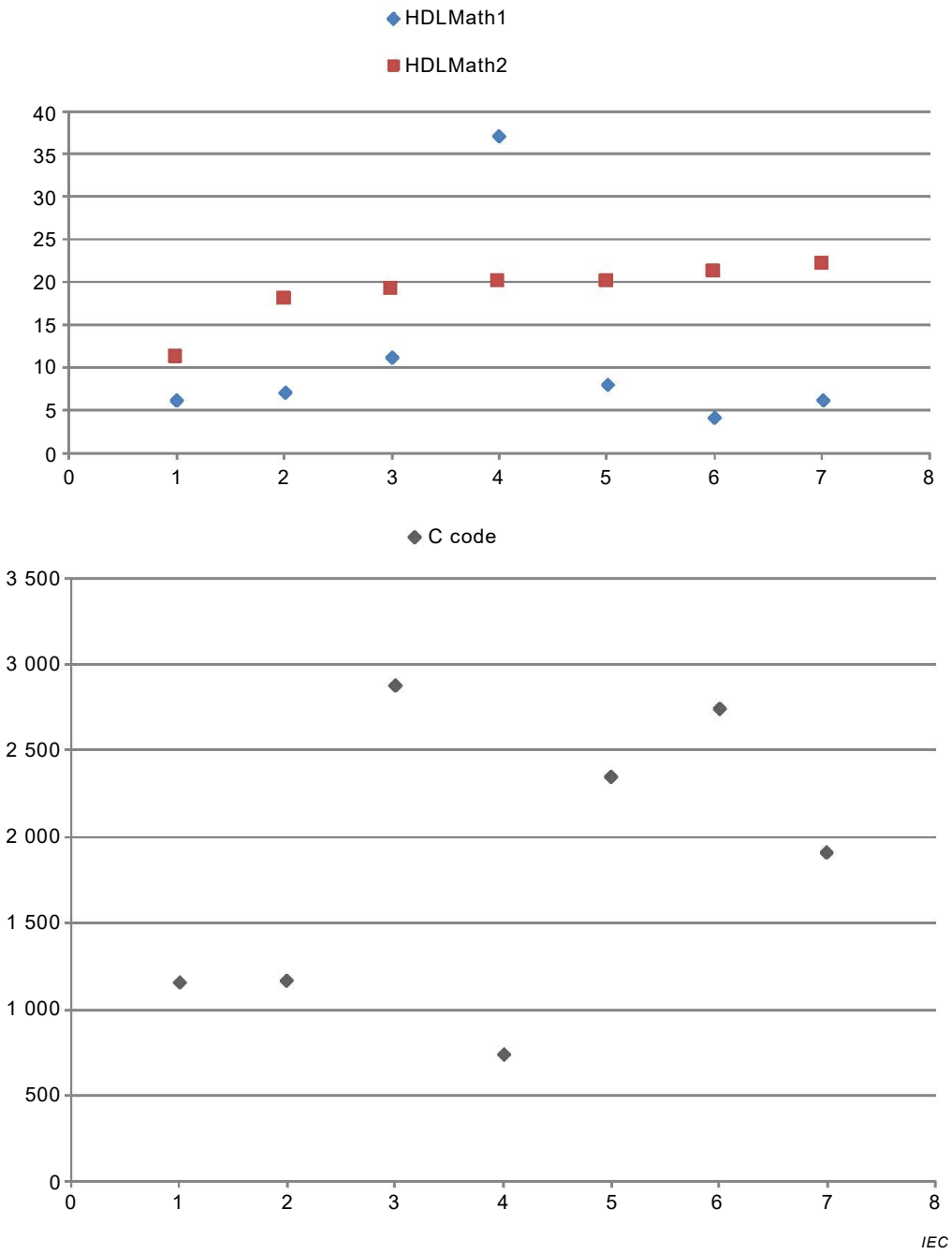


Figure 1 – Numbers of description lines

5 Functional requirements of HDLMath

5.1 General

When designing mathematical algorithms for system level modeling with an HDLMath, the HDLMath shall cover the following functional requirements in order to achieve utmost precision.

5.2 Mathematical expressions

The mathematical operators +, -, *, **, and / shall be applicable to any combination of the following operand and result formats: arbitrary-precision fixed-point, arbitrary-precision floating-point, integer, real, register, and constants. Trigonometric and hyperbolic (direct and inverse) functions shall be supported for any precision. Power, logarithm, and square root

operations are also needed. Figure 2 shows a part of a filter described using logarithm function in HDLMath1 and HDLMath2.

```
/**** HDLMath1 ****/
function ComputeGainC()
step = (srate/SIZE)/2;
l10 = log(10.0);
omega = 0;
for j = 1: SIZE
    Hr = vp*fact;
    for i = 1: 15
        t1 = vp*fact;
        t1 = t1*2*cos(omega*i*T);
        Hr = Hr+t1;
    end
    Hr = abs(Hr);
    y(0,j) = 20*log(Hr)/l10;
    omega = omega + step;
end
```

```
/**** HDLMath2 ****/
task ComputeGainC;
begin
 step = (srate/SIZE)/2;
 l10 = $VpLn(10.0);
 omega = 0;
 for (j = 1; (j < SIZE); j = j+1)
 begin
  vp = $VpCopyReg2Vp(mem[fir.Na]);
  Hr = vp*fact;
  for (i = 1; (i <= 15); i = i+1)
  begin
   vp = $VpCopyReg2Vp(mem[fir.Na-i]);
   t1 = vp*fact;
   t1 = t1*2*$VpCos(omega*i*T);
   Hr = Hr+t1;
  end
  Hr = $VpAbs(Hr);
  y[0][j] = 20*$VpLn(Hr)/l10;
  omega = omega + step;
 end
end
endtask
```

IEC

Figure 2 – Examples of mathematical expressions

5.3 Various kinds of precision computation

Data shall include scalar, complex numbers in Cartesian coordinates, and complex numbers in polar coordinates. The high level support shall be bit-accurate, i.e. the result of computations performed during simulation should match the results produced by the actual hardware. The formats (floating or fixed point) of high level data, as well as the number of bits in their respective fields, shall be modifiable during algorithm design. Modifying formats and their respective fields allows for a more efficient design space exploration. Table 2 shows the precision type of HDLMath1 and HDLMath2. HDLMath2 can handle data up to 1 Mbit in width and the precision type is defined using the “descriptor” construct.

Table 2 – Examples of precision type

	HDLMath1	HDLMath2
bit width	8, 16, 32, 64	defined using “descriptor” construct
sign	sign and unsigned	
floating point	single/double	
fixed point	using tool box	

5.4 Exception and error handling

To help optimize the implementation of mathematical algorithms, errors shall be minimized using only the necessary number of bits. Access shall also be provided to information regarding

the occurrence of overflow, underflow, maximum number of bits required, and cumulative error. Table 3 shows the overflow handling of HDLMath1 and HDLMath 2. HDLMath2 can define these using the “descriptor” construct.

Table 3 – Examples of overflow handling

	HDLMath1	HDLMath2
floating point	rounds to infinite value	defined using “descriptor” construct
integer	saturated with the maximum number	
fixed point	rounds to the specified number	

5.5 Multi-dimensional arrays

One- and two-dimensional arrays of any kind of data, including sparse arrays, and arithmetic and logical operations on any kind of legal combination of data shall be supported. This capability allows the implementation of all mathematical algorithms.

5.6 Mathematical functions

There shall be support for a large number of mathematical functions, such as differential equations, FFT (Fast Fourier Transform), DFT (Discrete Fourier Transform), finding eigenvalues and eigenvectors, norms and distances, finding roots of polynomials. Although all mathematical functionality can be written using the arithmetic operators, such an implementation would be slow. Figure 3 and Figure 4 show multi-dimensional arrays and mathematical functions in HDLMath1 and HDLMath2, respectively.

```
function ar_f_total = diff_math();
alpha = 1.65;
r_total_time = 0.005;
 . . . . . .
        ar_f_total(slice,j) = ar_f_spring(slice) +   ar_f_damper(slice) + ar_f_flux(slice);
        if (isZero)
            ar_f_total(slice,j) =    0;
        elseif (ar_f_total(slice,j)>0)
            ar_f_total(slice,j)= ar_f_total(slice,j);
        else
            ar_f_total(slice,j)   = 0;
        end
        if ((slice ~= 1) && (ar_f_total(slice,j) < 15))
            isZero = 1;_
. . . . . .
function dydx = vplode(t,x);
global coef;
dydx = [x(2);-(coef(2)*x(2)+coef(3)*x(1))/coef(1)];
. . . . . .
```

IEC

Figure 3 – Multi-dimensional arrays and mathematical functions in HDLMath1

```
module top;
parameter real alpha = 1.65;
parameter real r_total_time = 0.005;/* seconds */
. . . . . .
ar_f_flux[slice] = 0;
ar_f_total[j][slice] = ar_f_spring[slice] + ar_f_damper[slice] + ar_f_flux[slice];
ar_f_total[j][slice] = (isZero) ? 0: ((ar_f_total[j][slice]>0)?ar_f_total[j][slice]:0);
if ((slice != 0) && (ar_f_total[j][slice] < 15)
. . . . . .
$VpLODE(order, nrEq, h, nr_pts_per_ct_coef+1,x_ct, coef, Fe_ct, y_ct, ressymb);
end
. . . . . .
```

IEC

Figure 4 – Multi-dimensional arrays and mathematical functions in HDLMath2

5.7 Mixed numerical and symbolic computations

All the necessary processing should be performed in one execution. Users should not be burdened with passing data from a symbolic environment to a numeric environment. When performing symbolic simulation using strings, a string should be evaluated in the current context, as if it had been an expression in the numeric environment. Figure 5 shows mixed numerical and symbolic computations in HDLMath2, but HDLMath1 does not support this functional requirement.

```
/**** HDLMath1****/
syms x r;
symb1 = sin(r*x);
symb1_val = double(subs(symb1,[x r],[pi/6 pi/6]));
```

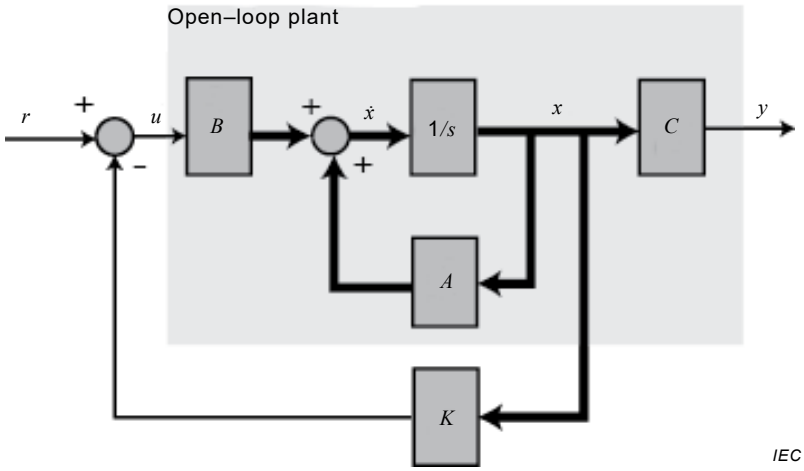
```
/**** HDLMath2****/
r = $Pi/6;
x = $Pi/6;
symbExpr1 = "$VpSin(r*x)";
$Eval(symbExpr1, val);
```

IEC

Figure 5 – Mixed numerical and symbolic computations in HDLMath1 and HDLMath2

5.8 Feedback process

One important aspect in control system design is to analyze the effects of feedback loops on the overall system. Figure 7 shows the feedback process in HDLMath1 and HDLMath2 for Figure 6.



IEC

Key

A, B, C, K	matrix values
$1/s$	Laplace transform of step function
r, u, x, y	variables

Figure 6 – Example of a feedback process

```
. . . . . .
poles = eig(A)
t = 0:0.01:2;
u = zeros(size(t));
x0 = [0.01 0 0];
sys = ss(A,B,C,0);
[y,t,x] = lsim(sys,u,t,x0);
plot(t,y)
xlabel('Time (sec)')
ylabel('Ball Position (m)')
. . . . . .
K = place(A,B,[p1 p2 p3]);
sys_cl = ss(A-B*K,B,C,0);
. . . . . .
```

```
. . . . . .
poles = $Eig(A);
$PrintM(poles, "%e");
/* The poles are:
poles[0].Re = -3.130495e+01, poles[0].Im = 0.0
poles[1].Re = -1.000000e+02, poles[1].Im = 0.0
poles[2].Re = 3.130495e+01, poles[2].Im = 0.0
*/
. . . . . .
K = $Place(A, B, poles);
BK = B*K;
AK = A-BK;
y = $LSim(AK, B, C, D, u, t0, dt, nr_samples, x);
. . . . . .
```

IEC

Figure 7 – Example of feedback process in HDLMath1 and HDLMath2

5.9 User-defined functions in C-code

Support for extending simulation functionality by having the capability to incorporate C code execution in the simulation in a standard manner is required to enhance performance.

The rationale behind this requirement is that many design teams have their own mathematical libraries and nothing else can work as well for them. In such cases, the designers can use their own libraries. Figure 8 is a user-defined C function call example in HDLMath1 and HDLMath2.

```
/****HDLMath1 ***/
static void yprime(. . . . . ){
  . . . . . .
      yp[1]=
2*y[3]+y[0]-mus*(y[0]+mu)/(r1*r1*r1)-mu*(y[0]-mus)/(r
2*r2*r2);
. . . . . .
    return;}
void mexFunction( i. . . . . ){
. . . . . .
    yprime(yp,t,y);
    return; }
```

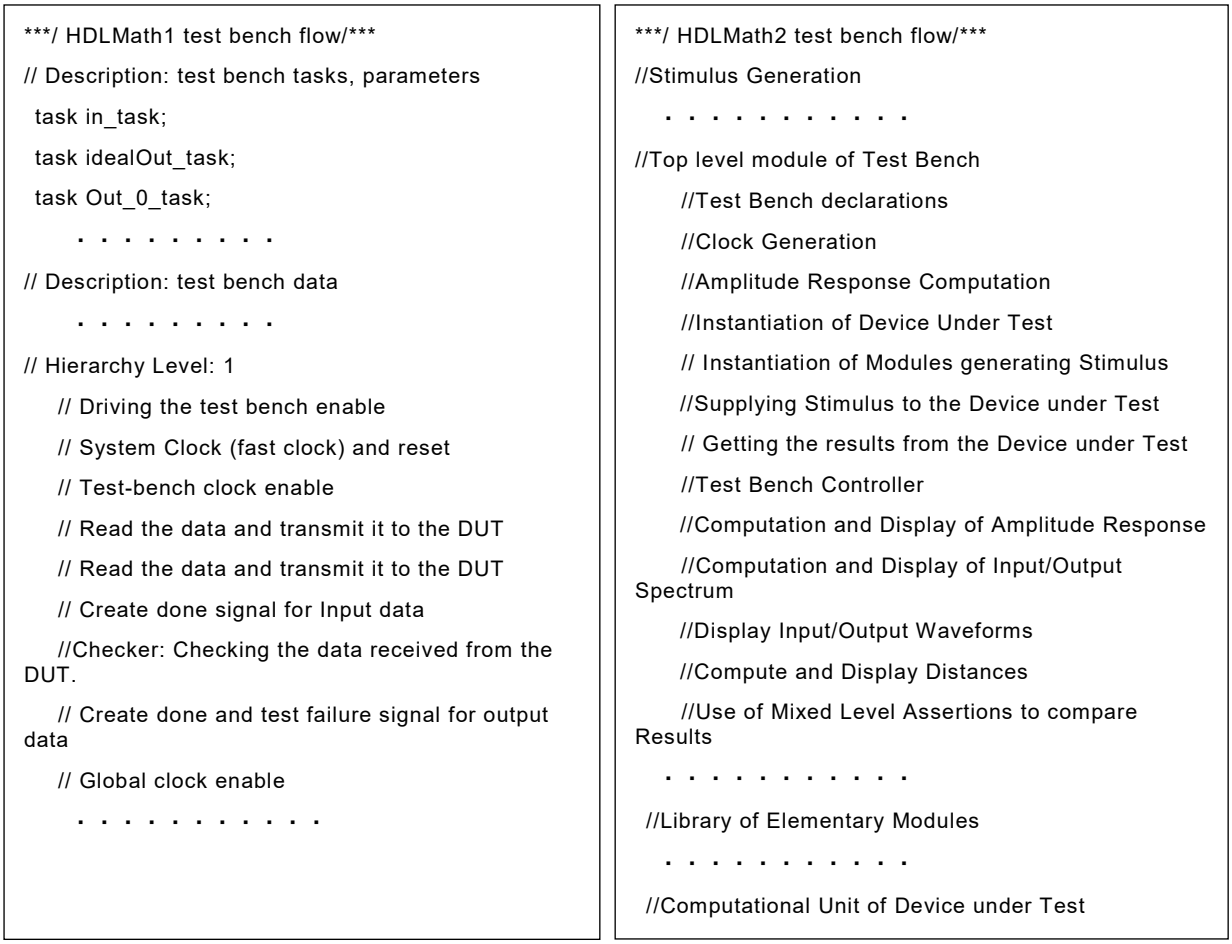
```
/****HDLMath2 ***/
long tf2ssc(long file, int line,
 . . . . . .
    int sz2SS, int st2SS, int end2SS, long SS);
 . . . . . .

M = tf2ssc(b, a);        /*call C code */


. . . . . .
```

IEC

Figure 8 – Examples of user-defined functions in C-code in HDLMath1 and HDLMath2

5.10 Verification environment

Test benches are an essential tool in the circuit design environment. They provide a virtual environment used to verify the correctness of the design or model. The test bench capability of an HDLMath language shall include four components: input, circuit, check functions, and output. Figure 9 shows the structure of test-bench descriptions in HDLMath1 and HDLMath2.



IEC

Figure 9 – Structure of test-bench description of HDLMath1 and HDLMath2

6 Comparison of current HDLMath languages

Table 4 shows a comparison between current HDLMath languages based on the functional requirements for an HDLMath.

These languages support most of the functional requirements, but they still require further descriptive capabilities for larger scale designs.

HDLMath1 has bit length limitations (maximum 64 bits) and several problems from a hardware design perspective such as limitations for functional requirements 2, 3, and 4.

HDLMath2 requires more functionality such as the ability to handle feedback processes without additional C coding.

HDLMath3 is able to describe mathematical algorithms, but it needs much C coding and associated debugging.

Table 4 – Comparison of current HDLMaths

Requirements	HDLMath1	HDLMath2	HDLMath3
1) Mathematical expressions	✓	✓	[a]
2) Various kinds of precision computation	✓[b]	✓	[a]
3) Exception and error handling	✓[c]	✓	[a]
4) Multi-dimensional arrays	✓[d]	✓	[a]
5) Mixed numerical and symbolic computations		✓	[a]
6) Mathematical functions	✓	✓	[a]
7) Feedback process	✓	[a]	[a]
8) User-defined function in C-code [a]	✓	✓	✓
9) Verification environment	✓	✓	✓

[a] This function is implemented using additional C coding.

[b] HDLMath1 has bit length limitations (maximum 64 bits).

[c] HDLMath1 does not support the setting of a flag that can be used by the hardware model such as the HDL generated by the HDL generator.

There is a structure in which an exception is returned to the calling function, but the user shall program the actual exception handling mechanism.

[d] The values are only of some small number of formats and sizes.

7 Conclusion

HDLMath is a language to describe and verify the behavior of entire systems and products using mathematical algorithms of electronic systems. HDLMath and its design environment shall support various kinds of design domains. When designing mathematical algorithms for system level modeling using HDLMath, the language shall cover the nine functional requirements to design system models precisely and concisely.

This document also describes small examples for each functional requirement and the status of current HDLMaths to accelerate mathematical algorithm description language (HDLMath) standardization. Current HDLMath languages have some of the functionalities required for mathematical algorithm design. However, they still need more extensive descriptive capabilities in order to handle larger scale design.

This document has focused on modeling and verification based on HDLMath because at this point in time synthesis based on HDLMath still requires further research and development.

Bibliography

[1] IEC 61691-1-1:2011, *Behavioural languages – Part 1-1: VHDL Language Reference Manual*

[2] IEC 62530:2011, *SystemVerilog – Unified Hardware Design, Specification, and Verification Language*

[3] IEC TR 62856:2013, *Documentation on design automation subjects – The Bird's-eye View of Design Languages (BVDL)*

_____

This page deliberately left blank

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Copyright in BSI publications

All the content in BSI publications, including British Standards, is the property of and copyrighted by BSI or some person or entity that owns copyright in the information used (such as the international standardization bodies) and has formally licensed such information to BSI for commercial publication and use.

Save for the provisions below, you may not transfer, share or disseminate any portion of the standard to any other person. You may not adapt, distribute, commercially exploit, or publicly display the standard or any portion thereof in any manner whatsoever without BSI's prior written consent.

Storing and using standards

Standards purchased in soft copy format:

- A British Standard purchased in soft copy format is licensed to a sole named user for personal or internal company use only.
- The standard may be stored on more than 1 device provided that it is accessible by the sole named user only and that only 1 copy is accessed at any one time.
- A single paper copy may be printed for personal or internal company use only.

Standards purchased in hard copy format:

- A British Standard purchased in hard copy format is for personal or internal company use only.
- It may not be further reproduced – in any format – to create an additional copy. This includes scanning of the document.

If you need more than 1 copy of the document, or if you wish to share the document on an internal network, you can save money by choosing a subscription product (see 'Subscriptions').

Reproducing extracts

For permission to reproduce content from BSI publications contact the BSI Copyright & Licensing team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email subscriptions@bsigroup.com.

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Useful Contacts

Customer Services
Tel: +44 345 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 345 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

bsi.