

elemapprox user manual

Title	elemapprox (Elementary functions approximation in ANSI C and Verilog HDL)
Author	Nikolaos Kavvadias (C) 2013, 2014
Contact	nikos@nkavvadias.com
Website	http://www.nkavvadias.com
Release Date	16 September 2014
Version	1.0.0
Rev. history	
v1.0.0	2014-09-16 Initial release. This is an expanded version built upon Mark G. Arnold's Verilog Transcendental Functions paper.

1. Introduction

`elemapprox` is an ANSI C code and Verilog HDL collection of modules that provide the capability of evaluating and plotting transcendental functions by evaluating them in single precision. The original work supports plotting in ASCII of a subset of the functions; this version provides a more complete list of functions in addition to bitmap plotting for the transcendental functions as PBM image files.

`elemapprox` has been conceived as an extension of Prof. Mark G. Arnold's work as published in HDLCON 2001. Most functions have been prefixed with the letter `k` in order to avoid function name classes in both ANSI C and Verilog HDL implementations.

The transcendental functions supported include most elementary functions (hence the name `elemapprox`) and the list is as follows:

Function	Description
<code>kfabs</code>	Floating-point absolute value (helper function).
<code>ksin</code>	Sine.
<code>kcos</code>	Cosine.
<code>ktan</code>	Tangent.
<code>kcot</code>	Cotangent.
<code>ksec</code>	Secant.
<code>kcsc</code>	Cosecant.
<code>rootof2</code>	Calculate root-of-2.

kexp	Exponential.
klog	Natural logarithm.
kpow	Powering function.
ksqrt	Square root.
khypot	Hypotenuse.
katan	Arc tangent.
katan2	Two-argument (x/y) arc tangent.
kasin	Arc sine.
kacos	Arc cosine.
kacot	Arc cotangent.
kasec	Arc secant.
kcsc	Arc cosecant.
ksinh	Hyperbolic sine.
kcosh	Hyperbolic cosine.
ktanh	Hyperbolic tangent.
kcoth	Hyperbolic cotangent.
ksech	Hyperbolic secant.
kcsch	Hyperbolic cosecant.
kasinh	Inverse hyperbolic sine.
kacosh	Inverse hyperbolic cosine.
katanh	Inverse hyperbolic tangent.
kacoth	Inverse hyperbolic cotangent.
kasech	Inverse hyperbolic secant.
kacsch	Inverse hyperbolic cosecant.

The reference paper and the corresponding presentation are available from the web at the following links:

- http://www.cse.lehigh.edu/~caar/marnold/papers/sanjose_hdlcon.doc
- <http://www.cse.lehigh.edu/~caar/marnold/presentations/freeddy.ppt>

2. File listing

The `elemapprox` library and modules include the following files:

/elemapprox	Top-level directory
AUTHORS	List of authors.
LICENSE	License agreement (Modified BSD license).
README.rst	This file.
README.html	HTML version of README.
README.pdf	PDF version of README.
rst2docs.sh	Bash script for generating the HTML and PDF versions.

VERSION	Current version.
/ansic	ANSI C implementation
clean-math-ansic.sh	Bash script for cleaning up the generated executables.
elemapprox.c	C code for the function approximations.
elemapprox.h	C header file for the above. Defines certain mathematical constants and declares function prototypes.
funcplot.c	Reference code for creating the plot data for the functions.
funcplot.h	C header file for the above.
graph.c	Collection of ASCII and PBM graphing functions.
graph.h	C header file for the above.
Makefile	GNU Makefile for building <code>testfunc.exe</code> .
plot-ansic-ascii.sh	Bash script for plotting the elementary functions as ASCII graphs using <code>testfunc.exe</code> .
plot-ansic-pbm.sh	Bash script for plotting the elementary functions as PBM images using <code>testfunc.exe</code> .
testfunc.c	Application code for the elementary function Options include PBM or ASCII image generation and function selection.
test<func>.pbm	Generated PBM image data for the function <func>.
test<func>.txt	Generated ASCII graph data for the function <func>.
test<func>-ascii.txt	Concatenation of the generated ASCII graph data for all supported functions.
/verilog	Verilog HDL implementation
clean-math-verilog.sh	Bash script for cleaning up the generated interpreted intermediate code (for Icarus Verilog).
constants.v	Certain mathematical constants.
elemapprox.v	Verilog HDL code for the function approximations.
elemapproxpp.v	Preprocessed version of the above, directly including the mathematical constants from <code>constants.v</code> and expanding all macro-definitions.
funcplot.v	Reference code for creating the plot data for the functions.
graph.v	Collection of ASCII and PBM graphing tasks.
Makefile	GNU Makefile for building <code>testfunc.exe</code> .
plot-verilog-ascii.sh	Bash script for plotting the elementary functions as ASCII graphs using <code>testfunc.v</code> . The script Icarus Verilog' VVP interpreter which is capable of parsing command-line options.
plot-verilog-pbm.sh	Bash script for plotting the elementary functions as PBM images using <code>testfunc.v</code> .

testfunc.v	Application code for the elementary function Options include PBM or ASCII image generation and function selection.
test<func>.pbm	Generated PBM image data for the function <func>.
test<func>.txt	Generated ASCII graph data for the function <func>.
test<func>-ascii.txt	Concatenation of the generated ASCII graph data for all supported functions.
/refs	Reference documentation
sanjose_hdlcon.doc	MS Word document for the manuscript: M. G. Arnold, C. Walter and F. Engineer, "Verilog Transcendental Functions for Numerical Testbenches," Proceedings of the Tenth International HDL conference, Santa Clara, California, March 1, 2001.
freddy.ppt	MS PowerPoint presentation of the above work.

3. Usage

Both the ANSI C and Verilog HDL versions can be used for generating graph data and depicting any of the supported transcendental functions via two similar scripts.

3.1. ANSI C

1. Run the following shell script from a Unix/Linux/Cygwin command line in order to generate an ASCII graph for each function.

```
cd ansic
$ ./plot-ansic-ascii.sh
```

All generated data are also concatenated to `testfunc-ascii.txt`.

2. Run the following shell script from a Unix/Linux/Cygwin command line in order to generate an ASCII graph for each function.

```
$ ./plot-ansic-pbm.sh
```

All generated data are saved in the form of PBM (monochrome bitmap) image files. Such files can be visualized using e.g. the public domain `Image viewer`: <http://www.nyam.pe.kr/>

3.2. Verilog HDL

1. Run the following shell script from a Unix/Linux/Cygwin command line in order to generate an ASCII graph for each function.

```
cd verilog
$ ./plot-verilog-ascii.sh
```

All generated data are also concatenated to `testfunc-ascii.txt`.

2. Run the following shell script from a Unix/Linux/Cygwin command line in order to generate an ASCII graph for each function.

```
$ ./plot-verilog-pbm.sh
```

All generated data are saved in the form of PBM (monochrome bitmap) image files.

4. Synthesis

The implementation code (either ANSI C or Verilog HDL) for the transcendental functions has not been tested for high-level or RTL synthesis.

5. Prerequisites

- Standard UNIX-based tools (tested with gcc-4.6.2 on MinGW/x86) [optional if you use Modelsim].
 - make
 - bash (shell)

For this reason, MinGW (<http://www.mingw.org>) or Cygwin (<http://sources.redhat.com/cygwin>) are suggested, since POSIX emulation environments of sufficient completeness.

- Icarus Verilog simulator (<http://iverilog.icarus.com/>). The Windows version can be downloaded from: <http://bleyer.org/icarus/>
- Alternatively, a commercial simulator like Mentor Modelsim (<http://www.mentor.com>) can be used (however this has not been tested).