kvcordic user manual

Title	kvcordic (Multi-function, universal, fixed-point CORDIC)
Author	Nikolaos Kavvadias (C) 2010-2022
Contact	nikos@nkavvadias.com
Website	http://www.nkavvadias.com
Release Date	20 February 2022
Version	1.1.0
Rev. history	
v1.1.0	2022-02-20 Synthesize cordic.c with Vitis HLS.
v1.0.1	2014-09-26 Updated file organization for Github; changed README to README.rst; CORDIC-EULA.txt to LICENSE.
v1.0.0	2013-02-22 Updated architecture. A single cycle is now needed per iteration. Documentation updated to RestructuredText. Logic synthesis scripts updated for Xilinx ISE/XST 14.6.
v0.0.2	2010-11-08 A universal CORDIC algorithm is now specified. The new version uses the full CORDIC interface (X,Y,Z) for input and output. Q2.14 (16-bit) signed fixed-point arithmetic is emulated through the use of integers.
v0.0.1	2010-11-04 The CORDIC-EULA.txt has been added.
v0.0.0	2010-11-01 Initial release.

1. Introduction

`kvcordic` is a collection of files comprising an implementation of a universal CORDIC algorithm (rotation/vectoring direction, circular/linear/ hyperbolic mode) high-level synthesis benchmark by Nikolaos Kavvadias. All design files except `cordic.c`, `cordic.nac`, and `cordic_test_data.txt` have been automatically generated. The original `cordic.vhd` has been optimized via (manual) operation chaining. `operpack.vhd`, `std_logic_textio.vhd` are simulation/synthesis library files, copyrighted by their respective authors.

IMPORTANT: Please go through the license agreement (`LICENSE`) to ensure proper use of the CORDIC IP CORE.

2. File listing

The CORDIC IP core distribution includes the following files:

/kvcordic	Top-level directory
AUTHORS	List of authors.
LICENSE	End-user license agreement for using `kvcordic`.
README.rst	This file.
README.html	HTML version of README.
README.pdf	PDF version of README.
VERSION	Current version of the CORDIC IP cores.
rst2docs.sh	Bash script for generating the HTML and PDF versions.
/bench/vhdl	Benchmarks VHDL directory
cordic_tb.vhd	Automatically-generated VHDL testbench file.
/doc	Documentation directory
/rtl/vhdl	RTL source code directory for the IP core
cordic.vhd	Automatically-generated VHDL design file (hand-optimized for operation chaining).
cordic_cdt_pkg.vhd	Package containing declarations.
/sim/rtl_sim	RTL simulation files directory
/sim/rtl_sim/bin	RTL simulation scripts directory
cordic.mk	Unix/Cygwin makefile for running a GHDL simulation.
/sim/rtl_sim/out	Dumps and other useful output from RTL simulation
cordic_alg_test_res-ults.txt	Check this file for output.
/sim/rtl_sim/run	Files for running RTL simulations
cordic.sh	Unix/Cygwin bash shell script for running a GHDL sim-ulation.
cordic_test_data.txt	Reference vectors.
/sim/rtl_sim/vhdl	VHDL source files used for running RTL simulations
operpack.vhd	Reduced version of Nikolaos Kavvadias' operator li-brary.
std_logic_textio.vhd	Modified version of a testbench-related package.
/sw	Software utilities
cordic.c	Reference C implementation for test vector generation.
cordic.dot	CDFG of the cordic procedure as a Graphviz file.
cordic.dot.png	PNG image for the above.
cordic.nac	The NAC description of the CORDIC application.
cordic-flp.txt	Comparison of fixed-point to floating-point CORDIC (using calls to the `math` C library) results.
/syn/vitis	Synthesis files for use with Xilinx Vitis

/syn/vitis/bin	Synthesis scripts directory
vhls.tcl	Pass Tcl directives to Vitis HLS.
/syn/vitis/run	Files for running synthesis
vsyn.sh	Bash shell script for synthesizing `cordic()` with Vitis HLS.
/syn/xise	Synthesis files for use with Xilinx ISE
/syn/xise/bin	Synthesis scripts directory
xst.mk	Standard Makefile for command-line usage of ISE.
/syn/xise/log	Generated log files from the synthesis process
cordic-xst14.6.txt	Synthesis report from Xilinx ISE (XST) 14.6.
/syn/xise/run	Files for running synthesis
syn.sh	Bash shell script for synthesizing `kvcordic` architectures with ISE.

3. Usage

1. Run the shell script from a Unix/Linux/Cygwin command line.

```
$ ./cordic.sh
```

After this process, the `cordic_alg_test_results.txt` file is generated containing simulation results. The GHDL simulation will also generate a VCD (waveform) file that can be opened with GTKwave:

```
$ gtkwave cordic_fsmd.vcd
```

2. Create, build and run a Modelsim project with the following files (in this order):

```
operpack.vhd
std_logic_textio.vhd
cordic_cdt_pkg.vhd
ram.vhd
cordic.vhd
cordic_tb.vhd
```

4. Synthesis

The CORDIC IP cores distribution includes scripts for logic synthesis automation supporting Xilinx ISE. The corresponding synthesis script can be edited in order to specify the following for adapting to the user's setup:

- `XDIR`: the path to the `/bin` subdirectory of the Xilinx ISE/XST installation where the `xst.exe` executable is placed
- `arch`: specific FPGA architecture (device family) to be used for synthesis
- `part`: specific FPGA part (device) to be used for synthesis

4.1. Running the synthesis script

For running the Xilinx ISE synthesis tool, change directory to the `/syn/xise/run` subdirectory from the top-level directory of CORDIC:

```
$ ./syn/xise/run
```

and execute the corresponding script (for synthesizing `hwlu`):

```
$ ./syn.sh
```

The synthesis procedure invokes several Xilinx ISE command-line tools for logic synthesis as described in the corresponding Makefile, found in the the `/syn/xise/bin` subdirectory.

Typically, this process includes the following:

- Generation of the `*.xst` synthesis script file.
- Generation of the `*.ngc` gate-level netlist file in NGC format.
- Building the corresponding `*.ngd` file.
- Performing mapping using `map` which generates the corresponding `*.ncd` file.
- Place-and-routing using `par` which updates the corresponding `*.ncd` file.
- Tracing critical paths using `trce` for reoptimizing the `*.ncd` file.
- Bitstream generation (`*.bit`) using `bitgen`, however with unused pins.

Finally, the `cordic.bit` bitstream file is produced.

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.

- GHDL simulator (http://ghdl.free.fr) [optional if you use Modelsim]. Provides the `ghdl` executable (has several Windows versions, with 0.29.1 and 0.31 being the latest). It also installs GTKwave on Windows. Note that the latest version (0.31) from http://sourceforge.net/project/ghdl-updates/ does not include GTK-wave.

- Xilinx ISE (free ISE webpack is available from the Xilinx website): http://www.xilinx.com) The 14.6 version on Windows 7/64-bit is known to work.