# A ARTIFACT APPENDIX

## A.1 Abstract

Our Cordic implementation using posit [38] and the artifact [32] is open source and publicly available. We also provide our implementation and the artifact in an archival link. The artifact contains the source code and scripts to automatically run experiments and reproduce our results. To ease installation effort, a prebuilt docker image containing the required software and the artifact is also available.

## A.2 Artifact check-list (meta-information)

- **Algorithm:** Cordic Algorithm.
- **Program:** C++, Python3, and SoftPosit
- **Compilation:** g++
- **Run-time environment:** Experiments were performed on ubuntu 18.04 and confirmed to work on macOS Catalina.
- **Hardware:** Modern machines with at least 2.3GHz processors and 8GB memory should be sufficient.
- **Metrics:** The artifact contains the expected results.
- **Experiments:** Download and run docker image, run the test scripts, and observe the results.
- **How much time is needed to prepare workflow (approximately)?:** Preparation should take less than 30 minutes
- **How much time is needed to complete experiments (approximately)?:** All experiments may take more than 6 days depending on parallelization option. Therefore, we provide parallelization option as well as a shorter version of the experiments which will take roughly 1.5 hours at most.
- **Publicly available?:** Yes.

## A.3 Description

*A.3.1 How to access.* The artifact can be downloaded from the archive at http://doi.org/10.5281/zenodo.3774064 or use the prebuilt docker image.

*A.3.2 Software dependencies.* Our implementation is written in C++ and uses SoftPosit library. The experiment scripts are written in Python3 and uses numpy and matplotlib. All softwares are installed in the docker image.

## A.4 Installation

*A.4.1 Using docker image.* Install Docker by going to https://docs.docker.com/get-docker/ and selecting the installation file for the corresponding OS and follow the instructions. Then, pull the docker image and run it.

```
$ docker run -it jpl169/cordicwithposit
```

*A.4.2 Manual installation with Ubuntu 18.04.* To evaluate the artifact without using Docker, install required packages:

```
$ sudo apt-get update
$ sudo apt-get install -yq --no-install-recommends apt-utils
$ sudo apt-get install -yq build-essential python3 python3-pip \
libgmp3-dev libmpfr-dev git
$ python3 -m pip install numpy matplotlib
```

Next, download and build the SoftPosit library:

```
$ git clone https://gitlab.com/cerlane/SoftPosit.git
$ cd SoftPosit/build/Linux-x86_64-GCC/
$ make
$ cd ../../..
```

Finally, untar the artifact and build the code:

```
$ export SOFTPOSITPATH=<path to SoftPosit directory>
$ tar -xvf CordicWithPosit.tar.gz
$ cd CordicWithPosit && make
```

## A.5 Experiment workflow

This artifact provides scripts to automatically conduct experiments described in Section 5.

*Graph Generation.* This experiment creates three graphs presented in Section 5. To run the experiment, use the command:

```
$ python3 runGraphGeneration.py
```

This script generates three graphs, `sin.pdf`, `cos.pdf`, and `atan.pdf` in `graph` directory, which can be compared against the reference graph in `expected` directory. To copy pdf files (for example sin.pdf) from docker container to the host machine, use the command

```
$ exit
$ docker cp <container id>:/home/CordicWithPosit/graph/sin.pdf .
$ docker start <container id> && docker attach <container id>
```

*Accuracy Experiment (Full).* The full accuracy evaluation can be run using the command:

```
$ python3 runAccAnalysis.py <optional: # of parallelization>
$ python3 GenerateTableForAcc.py
```

By default, the first python script without parallelization argument runs 4 experiments in parallel. You can use the argument to specify how many experiments to run in parallel. The script runs a total of 10 individual experiments which can take up to 20 hours each. Without parallelization, these experiments can take up to 6 days. This experiment automatically compares the output (`table/table2table3.txt`) to the reference result (`expected/table2table3.txt`) and checks whether they are the same.

*Accuracy Experiment (Fast).* Instead of the full accuracy evaluation script above, you can run a simplified evaluation script that uses $0.01\times$ of the total input space. To run this experiment, use the command:

```
$ python3 runSimplifiedAccAnalysis.py <optional: # of parallelization>
$ python3 GenerateTableForSimplifiedAcc.py
```

This experiment should take 1 to 1.5 hours even without parallization. The script automatically compares the output (`table/simpleTable2table3.txt`) to the expected result (`expected/simpleTable2table3.txt`) and checks whether they are exactly the same or not in the terminal.

## A.6 Evaluation and expected result

The generated graphs should be compared against the the graphs found in `expected` directory with the same file name. The accuracy experiment automatically compares the result against the expected result.

## A.7 Experiment customization

Our Cordic implementation is built as a static library which can be found in `lib/lib_cordic.a` and the header file can be found in `include/cordic.h`. We have provided an example program in the `example` directory. Use the following command to test the example:

```
$ cd example && make && ./example
```