Algorithmic C (AC) Math Library

Piece-wise Linear Square Root ac_sqrt_pwl()

May 2018

Mentor Graphics Corporation

8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777. Telephone: 503.685.7000

Toll-Free Telephone: 800.592.2210 Website: www.mentor.com

SupportNet: supportnet.mentor.com/

Introduction

Function ac_sqrt_pwl is piecewise linear implementation of a square root function, optimized to provide high performance with quick results. This function is implemented as an overloaded function for ac_fixed, ac_float, ac_complex of ac_fixed and takes positive values as inputs. If negative input is provided to it, it will throw a compilation error. It provides good accuracy. It is observed that the pwl implementation is faster than other algorithms and consumes significantly smaller area making it useful as a basic building block in high speed IP blocks.

The Piece-wise Sqrt Toolkit

The ac_sqrt_pwl toolkit is accessed in Catapult by picking Examples->Math-> AC Square Root Piece-wise Linear. Three example scripts are provided in the Example Project section:

- Run fixed-point example(ac_fixed): Uses ac_sqrt_pwl function with ac_fixed data type for input and output arguments.
- Run floating-point example (ac_float): Uses ac_sqrt_pwl function with ac_float data type for input and output arguments.
- Run complex fixed-point example (ac_complex<ac_fixed>): Uses ac_sqrt_pwl function with complex ac_fixed data type for input and output arguments.

This document is viewed by selecting "AC Piece-wise Linear Square Root Function" in the Documentation section.

Invoke Catapult in an empty directory and select "Run fixed point example(ac_fixed)" script. Then click on "Export Files". The following files are observed:

- ReadMe.txt: Describes the toolkit briefly.
- ac_sqrt_pwl.pdf: This document.
- ac_sqrt_pwl_tb.h: Header file which defines input and output data types. Also declares the testbench function "project".
- ac_sqrt_pwl_tb.cpp: C++ file which implements the function "project" which calls the ac_sqrt_pwl function. Additionally implements a testbench which exercises ac_sqrt_pwl across a programmable range of inputs. Accuracy is tracked and reported.
- ac_sqrt_pwl_tb_ac_fixed.tcl: A Catapult TCL script which configures synthesis and SCVerify to exercise ac_sqrt_pwl using ac_fixed data type for input and output parameters.
- ac_sqrt_pwl_tb_ac_float.tcl: A Catapult TCL script which configures synthesis and SCVerify to exercise ac_sqrt_pwl using ac_float data type for input and output parameters.
- ac_sqrt_pwl_tb_ac_complex_ac_fixed.tcl: A Catapult TCL script which configures synthesis and SCVerify to exercise ac_sqrt_pwl using ac_complex<ac_fixed> data type for input and output parameters.
- ac_sqrt.mdl: A Simulink model which is used to exercise ac_sqrt_pwl in Simulink to compare accuracy with Simulink modeling techniques.

The ac_sqrt_pwl_tb.h header file defines data types for input and output arguments to ac_sqrt_pwl. These definitions are as follows:

```
//Declaration of ac_fixed datatypes
#if defined(TEST_SQRT_AC_FIXED)
typedef ac_fixed<32,16, false, AC_RND, AC_SAT> input_type;
typedef ac_fixed<21,8, false, AC_RND, AC_SAT> output_type;
#else

//Declaration of ac_float datatypes
#if defined(TEST_SQRT_AC_FLOAT)
typedef ac_float<9, 3, 1, AC_RND> input_type;
typedef ac_float<32, 16, 4, AC_RND> output_type;
//typedef ac_float<5, 1, 1, AC_RND> output_type;
#else

//Declaration of ac_complex <ac_fixed> datatypes
#if defined(TEST_SQRT_AC_COMPLEX_AC_FIXED)
typedef ac_complex<ac_fixed<32, 12, true, AC_RND, AC_SAT> >
input_type;
typedef ac_complex<ac_fixed<32, 16, true, AC_RND, AC_SAT> >
output_type;
#else
#error Must select a datatype
#endif
#endif
#endif
```

As seen, exactly one of the compile-time macros `(TEST_SQRT_AC_FIXED, TEST_SQRT_AC_FLOAT, TEST_SQRT_AC_COMPLEX_AC_FIXED)` is selected for each testcase. An example of how this is done is seen in ac_sqrt_pwl_tb_ac_fixed.tcl:

```
options set Input/CompilerFlags {-DTEST_SQRT_AC_FIXED}
solution options set Input/CompilerFlags {-DTEST_SQRT_AC_FIXED}
```

For more information on Catapult datatypes:

`$MGC_HOME/shared/pdfdocs/ac_datatypes_ref.pdf`

The ac_sqrt_pwl_tb.cpp C++ file implements a wrapper function - "project" - which calls the ac_sqrt_pwl function. This function is also defined to be the top for Catapult synthesis, and for SCVerify usage:

```
#include <mc_scverify.h>
//defining top design and using CCS_BLOCK to declare the design
#pragma hls_design top
void CCS_BLOCK(project)(
    input_type &input,
    output_type &output
)
{
    ac_sqrt_pwl(input, output); //can be replace by, output =
ac_sqrt_pwl <output_type> (input);
}
```

Running And Analyzing The ac_fixed Example

After exporting files as discussed above, in Catapult, type "source ac_sqrt_pwl_tb_ac_fixed.tcl". This script uses ac_fixed data type for input and output parameters using the following parameters and widths:

```
typedef ac_fixed<32,16, false, AC_RND, AC_SAT> input_type;
typedef ac_fixed<32,16, false, AC_RND, AC_SAT> output_type;
```

As the script runs, the following groups of messages are seen in Catapult output:

- Synthesis output: As mentioned, the "project" function is the top of Catapult design, and it is synthesized. This function simply calls the ac_sqrt_pwl function, which is where all the synthesis actions are performed.
- SCVerify testbench compilation and run. This is discussed later in this document.
- Simulink S-Function compilation: This script uses the Matlab MEX compiler to generate an S-Function suitable for simulation in Simulink. More discussion of this usage later.

SCVerify Behavior and Output

In the ac_sqrt_pwl_tb.cpp file, a SCVerify C++ testbench is included. This testbench is implemented in CCS_MAIN, and "sweeps" from some minimum value to a maximum value, incrementing by some step size. At each step, the unix sqrt function is called, and compared to ac_sqrt_pwl output. A percentage error is calculated, and if this error percentage is over some value, an error is flagged.

If input and output is complex, a special sqrt function is implemented which computes the expected value.

For this example, we sweep from 0 to 500 using one "Quantum" as step size. A Quantum is based on the fixed point fraction size of the input variable. A Quantum is the value represented by the smallest precision possible; for this example this value is 1/(2^16) = 1/65536 = 0.00001525878. An error message is printed if the difference between exact and approximation is greater than 1%.

These simulation parameters are set by this line in the ac_sqrt_pwl_tb_ac_fixed.tcl script:

```
flow package option set /SCVerify/INVOKE_ARGS {0 500 - 1}
```

The first two arguments are min and max values to be applied. Third parameter is increment size where '-' means to use a Quantum from input specification, and fourth argument is allowed difference threshold as a percentage.

In the case of complex data types, expected square root is computed using a formula which produces magnitude and angle values; In this case, the threshold is used when comparing magnitudes. A 5th parameter is used to provide a separate threshold for angle.

Numeric difference between the exact sqrt value and approximated sqrt value is calculated by the equation:

$$d(x,y) = \frac{abs(x-y)}{\max\left(abs(x), abs(y)\right)} \times 100$$

Note that an odd thing happens when either value is exactly 0, as this example can see because sqrt(0) is expected to be 0. If either value is 0, and the other is any non-zero value, then the above expression will compute 100% error differential. If both values are 0, then division by 0 is attempted which is also bad. Because of these, the error computation special-cases these scenarios to avoid pessimism. If both values are zero, then zero error is used. If only one value is zero, then the error is an extrapolation based on history of computations of the previous two datapoints - else 0 is used.

At the end of the catapult output log, these lines are printed, which means the test passed with maximum error being 0.768%:

```
# ============================================
# Simulating design
# cd ../..;
./Catapult/project.v1/scverify/orig_cxx_osci/scverify_top 0 500 -
1.0
# ============ sqrt_pwl test ==================
# lower_limit    = 0
# upper_limit    = 500
# step           = 1.52588e-05
# allowed_error  = 1
#
# Testbench finished
# max_error =0.194932  with sample =6.10352e-05
expected=0.0078125  actual=0.00782776
# ===================================================
```

Maximum error is seen with input value 6.1035e-05 which is exactly 4 Quantums from 0.

The testbench implemented in ac_sqrt_pwl_tb.cpp can optionally produce a data file which logs each data point in a format to be read into Excel. This file is plot_values.csv and is generated by setting this compile-time macro:

```
-DPLOT_SQRT_FILE_WRITE
```

Note that this file can get very large since one line is printed for each simulation point.

Simulink Simulation and Accuracy Visualization

As discussed above, default accuracy for the ac_sqrt_pwl toolkit is very nice. However, if different parameters are used, accuracy is likely to be affected. A Simulink setup, or harness, is provided to allow the user to modify ac_sqrt_pwl parameters, then measure accuracy. This section shows how to use Simulink to measure accuracy for the default toolkit setup, as well as how to modify parameters and measure the impact on accuracy. In order to do either of these, it is required that the user have Simulink licensing and installation of R2017b or later. The MATLABROOT environment variable should be set to that installation.

Default Toolkit Model Measurements

In Catapult, after source-ing ac_sqrt_pwl_tb_ac_fixed.tcl, issue this command:

```
flow run /Matlab/launch_matlab ac_sqrt.mdl
```

After some time, Matlab/Simulink will be seen. This command invokes matlab in the ./simulink directory, then opens the ac_sqrt.mdl system model. A pre-configured Simulink model is opened which simulates the ac_sqrt_pwl function for 100 seconds of time, using a simple ramp function with step size .01 seconds. The ramp is configured to sweep from 0.0 to 500 over the 100 seconds. The ac_sqrt_pwl function is embedded in the S-Function.

The Simulink schematic looks something like Figure 1:

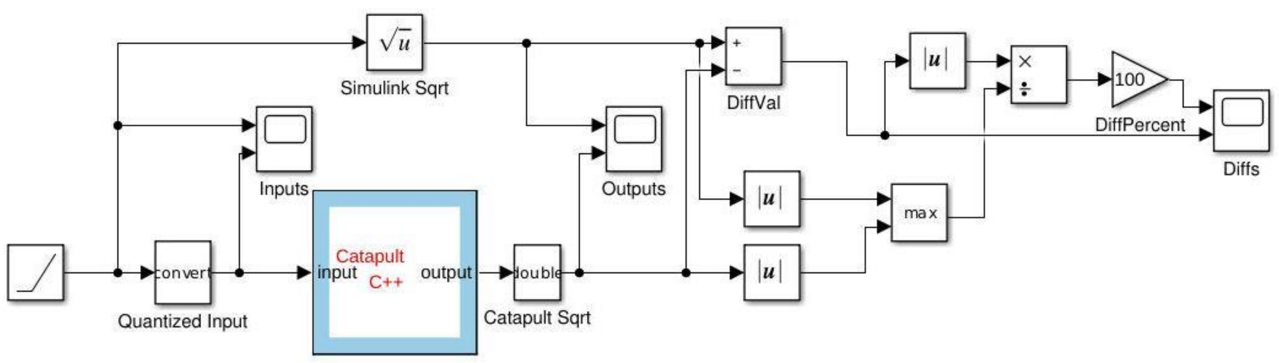


Figure 1: Simulink Schematic for ac_sqrt_pwl Measurements

It is seen in Figure 1 that the Simulink ramp source drives both the catapult S-Function for ac_sqrt_pwl and the Simulink square root math function. Scopes are used to view ramp input ('Inputs'), square root function outputs ('Outputs'), and differences between the two models ('Diffs').

The percent difference between two model outputs is computed by matlab at each timestep as:

$$d(x,y) = \frac{abs(x-y)}{\max(abs(x),abs(y))} \times 100$$

Simulink is taken as "x", and the catapult approximated ac_sqrt_pwl function is "y". Difference is plotted as a percentage, as well as raw value.

In Simulink, choose Simulation->Run. The observed waveforms will look something like Figure 2:

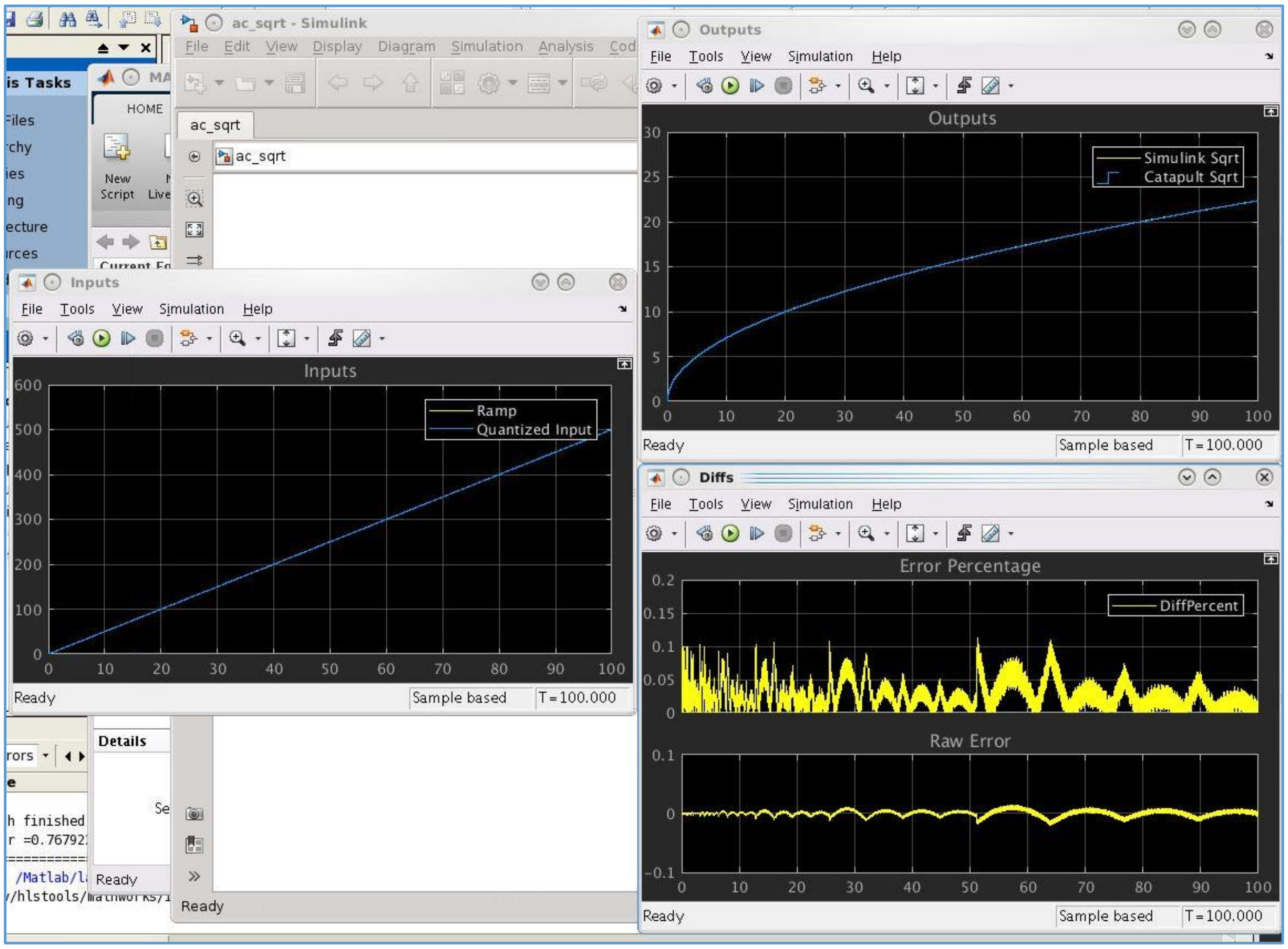
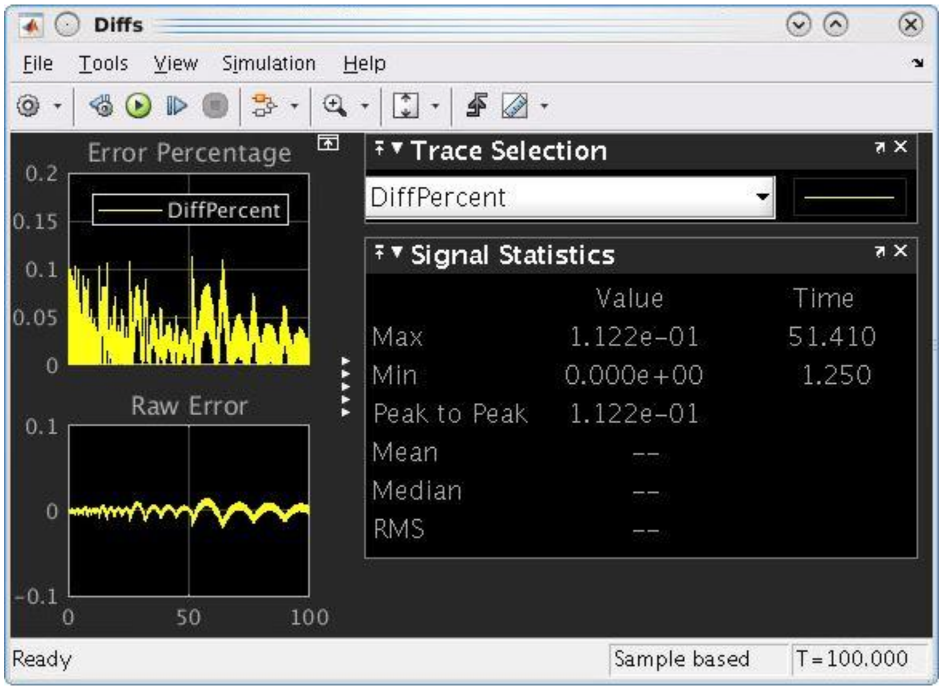


Figure 2: Waveforms After ac_sqrt_pwl Simulation

It is seen that "Error Percentage" looks to vary between 0% to 0.15%. Simulink can provide more accurate measurements. In the Diffs scope, choose Tools->Measurements->Signal Statistics. The measured data is shown in Figure 3:



	Value	Time
Max	1.122e-01	51.410
Min	0.000e+00	1.250
Peak to Peak	1.122e-01	
Mean	--	
Median	--	
RMS	--	

Figure 3: Difference Measurements for ac_sqrt_pwl vs Simulink sqrt model

The default toolkit usage has maximum error percentage 0.112% at simulation time 51.4s. This compares to SCVerify maximum 0.195%.

Modified Model and Accuracy Measures

Now assume that different parameters are desired for QoR reasons, and accuracy needs to be verified.

We will experiment with reduced accuracy by reducing output data precision and range. As described earlier, these parameters are specified in the ac_sqrt_pwl_tb.h header file. We will reduce output to 5 integer bits and 8 fraction bits; the earlier Simulink run used 16 fraction bits.

Now edit the ac_sqrt_pwl_tb.h header file and change output type from this:

```
typedef ac_fixed<32,16, false, AC_RND, AC_SAT> output_type;
```

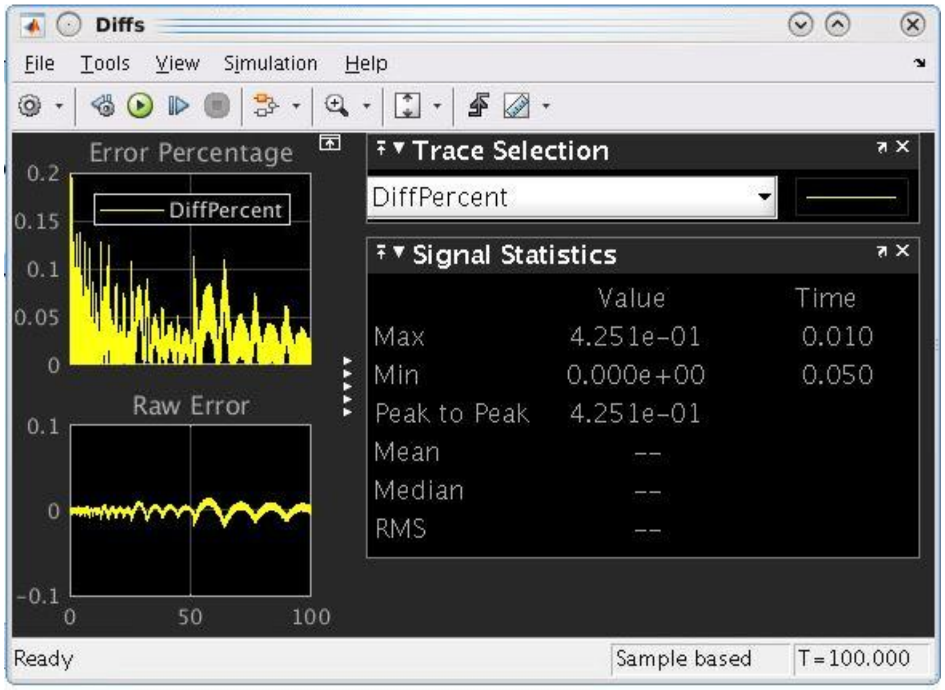
to this:

```
typedef ac_fixed<13, 5, false, AC_RND, AC_SAT> output_type;
```

Exit Catapult and Simulink, then rebuild the project:

```
  catapult -shell -file ac_sqrt_pwl_tb_ac_fixed.tcl
```

Then cd into the simulink directory, invoke matlab, and specify "open('ac_sqrt.mdl')". Re-run the simulation and observe the scope as shown in Figure 4:



```
              Value        Time
Max           4.251e-01    0.010
Min           0.000e+00    0.050
Peak to Peak  4.251e-01
Mean          --
Median        --
RMS           --
```

Figure 4: Difference Measurements for ac_sqrt_pwl after accuracy changes

We see that reduction of output fraction width by 50% leads to an increase of maximum error to 0.425%.