A fast implementation of Schmitt Trigger in MATLAB and GNU Octave

Vasile Daria-Andreea, Petrița Teodor

Faculty of Electronics, Telecommunication and Information Technology, Timișoara, România

INTRODUCTION

The scientific activity of analysis, modeling and simulation involves large volume of computation and data processed using software routines. This is true for signal processing also; as the size of the signal increases, the more difficult is to process it.

The software version for Schmitt Trigger has no implementation in MATLAB's Communications Toolbox and it has an implementation in GNU Octave in signal package that implements a multi signal Schmitt trigger with levels with inconstant results in our trials. The only implementation in MATLAB is in the one of Simscape™ Library, which is a part of the Simulink® environment. The solution to oscillation or multiple clocking problems is to use a Schmitt trigger type device to translate the slow or noisy edges into something faster that will meet the input rise and fall specifications of the following device. An ideal Schmitt trigger input does not rise and fall time limitations.

There is implemented a compiled function, which solves the problem of the implementation and the processing speed.

METHODS

Opposed to ordinary comparator, which is the equivalent of the comparator operator in software (e. g. x>threshold or x<threshold, depending on the implementation)

The switching of the output at different thresholds depending on the past state is called hysteresis. Our implementation takes into consideration not the two thresholds, but their average and their difference, called hysteresis gap. The middle threshold is at the half distance of the hysteresis gap between the upper and the lower threshold. In our take, the thresholds are:

low_threshold=middle_treshold-hysteresis_gap/2

high_threshold=middle_treshold+hysteresis_gap/2

With the defined, the function call is the same for MATLAB and GNU Octave:

output_signal_vector=trigger_smith,input_signal_vector, middle_treshold,hysteresis_gap;

For e.g trigger_smith(input_signal_vector, 0, 1) will set the thresholds to −0.5 to +0.5. The iterative loop excludes input values tests since to be as fast as possible. The output values can be either 0 or 1, and the initial value is set to 0 by default.

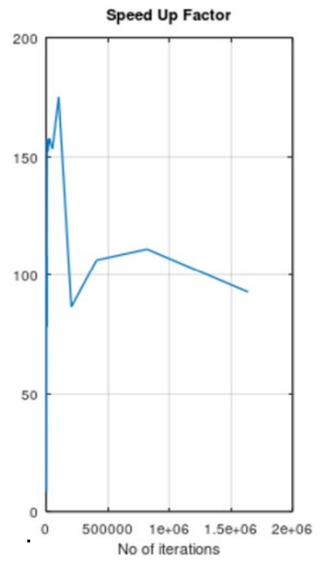

Figure 1: Performance of for loops algorithm MATLAB equivalent vs. for loops in C++ function

The compiled vectored functions in both suites are exhibiting a superior speed than the own language iterative implementation. Since not every function can be implemented in vectored mode, an iterative loop may not be a proper solution for large signals (or other vector/matrix processing) due to long times involved. The results are consistent and improve the speed until 100x times.

To assess the suitability of the both suites we searched for fast compilation and execution of the code and also check the compatibility of the programming languages.

The method is called a MEX-files and allows to use the software in C/C++ proper withinside the MATLAB user interface using classes and methos by mex libraries. GNU Octave can dynamically load and execute functions written in C/C++ using oct libraries. MEX – files and OCT-files are pieces of C/C++ code that have been compiled with the MATLAB/ GNU Octave API into a dynamically loadable object. For MATLAB suite the function is compiled with the command mex -setup C++ and after mex trigger_smith.cpp commands. For GNU Octave the function is complied with mkoctfile trigger_smith.cpp. The both command create new files with the extension .mex for MATLAB, respectively .oct for GNU Octave file that contain the code recognized by the both suites for calling them.

RESULTS

Our function was tested on various computer configurations, reference machine: Processor: Intel(R) Core (TM) M5Y71 CPU @ 1.20GHz 1.40 GHz Memory: 4,00 GB, with a 10^9 samples signal (no of iterations), we obtained a 5x average speed increment in MATLAB and a 2000x speed increment in GNU Octave. The results are showed speeds in both suites for our implementation in the figures below.

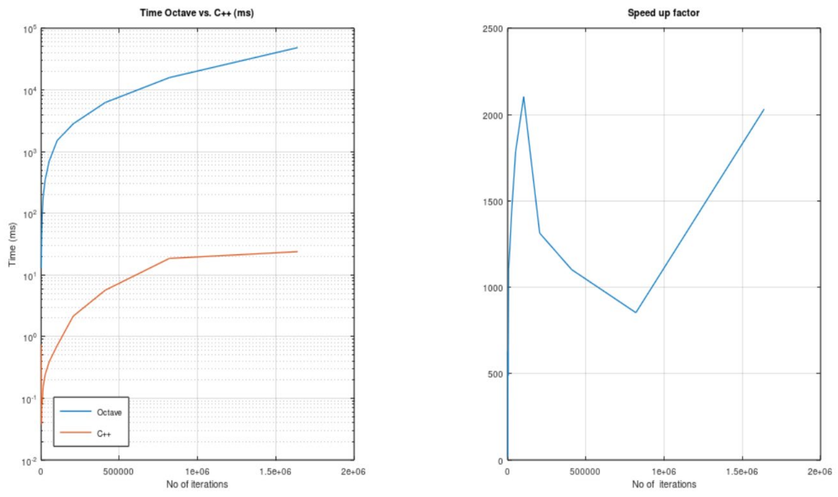

Figure 2: left: loop time for various number of iterations, GNU Octave vs C++ ; right: Speed up Factor C++ vs GNU Octave script

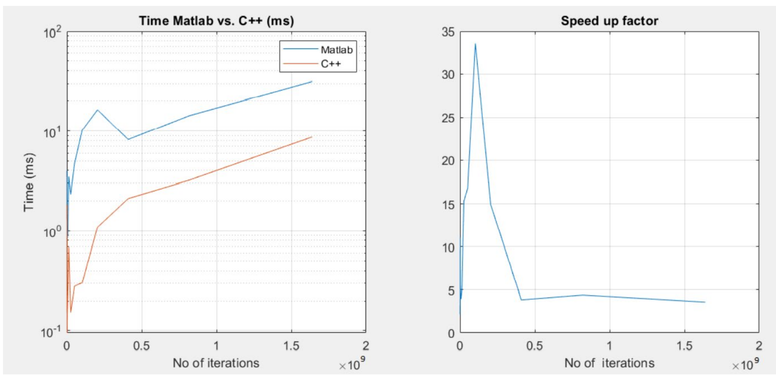

Figure 3: left: Loop time for various number of iterations, MATLAB vs C++; right: Speed up Factor C++ vs MATLAB script:

Since there is a Schmitt trigger function in GNU Octave package, we tested our function against the existing one and compare the results. We tested a normal random signal of unitary rms value, generated with randn function against the two triggers. For an average of 40 times the speed increases as it can be observed below.

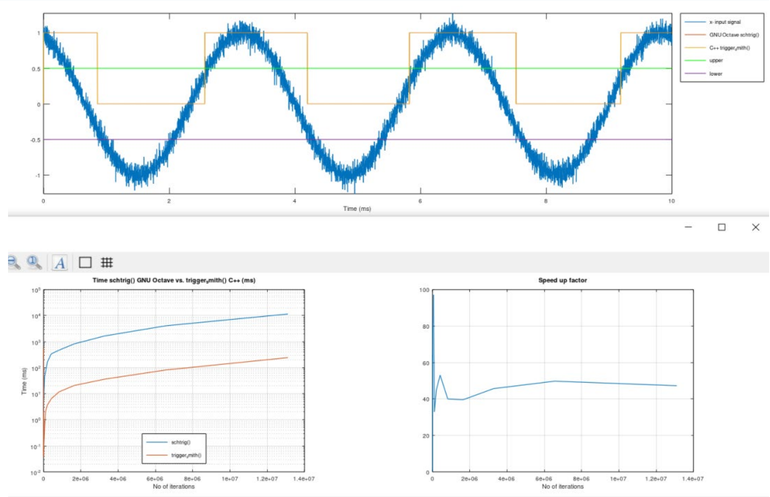

Figure 4. Performance between schtrig() and trigger_smith() functions from speed point of view

CONCLUSIONS

- **decrease significantly the signal processing time and brings improvements, being a fast tool for making calculations in which the waiting time is considerably reduced**
- **application that it can be used is to measure the frequency of a signal, or to count the pulses of a signal**
- **develop some fast scripts for projects like Microwave Doppler Radar for the frequency counter and foot press detector for an ankle torque measurement device**
- **solved in an efficiently the missing function, documenting at the same time the steps needed to implement other useful functions**

REFERENCES

1. „The Math Works, Inc. MATLAB. Version 2020a, The Math Works, Inc., 2020. Computer Software." https://ch.mathworks.com/ (access date: 12 June 2022).
2. J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring, „GNU Octave version 5.2.0 manual: a high-level interactive language for numerical computations.", 2019. https://www.gnu.org/software/octave/index (access date: 14 May 2022).
3. „C++ vs Octave | What are the differences?" https://stackshare.io/stackups/cplusplus-vs-octave (access date: 19 May 2022)
4. Andrews, T., 2012. Computation Time Comparison Between MARLAB and C++ Using Launch Windows. Digital commons.calpoly.edu. Available at:<http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1080&context=aerosp> (access date 25 May 2022)
5. "Create a C++ MEX Source File - MATLAB & Simulink - MathWorks Switzerland". https://ch.mathworks.com/help/matlab/matlab_external/c-mex-source-file.html (access date: 23 May 2022).
6. GNU Octave: Getting Started with Oct-Files". https://octave.org/doc/v4.2.0/Getting-Started-withOct_002dFiles.html (access date: 18 May 2022).