CprE 381 Toolflow Manual

Testing Framework

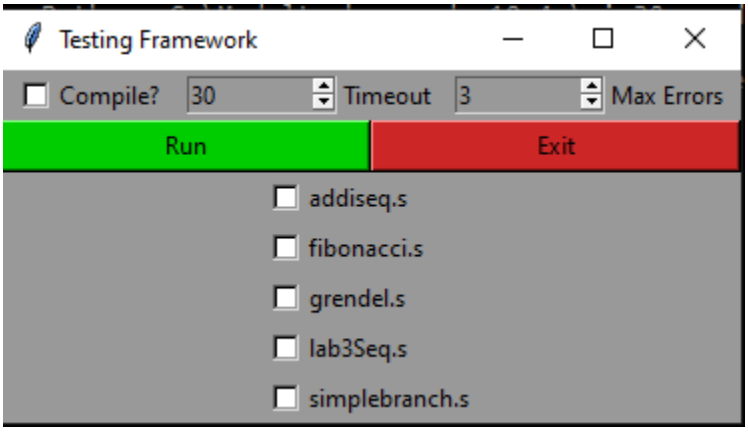
The goal of the test framework is to allow students to compare the output of their processor with that of MARS. It allows for compilation of a processor as well as the execution of various assembly programs.

Getting Started

Opening the test framework, you should see the following files present.

internal	2/22/2021 3:22 PM	File folder	
mips	2/22/2021 10:48 AM	File folder	
src	2/22/2021 2:23 PM	File folder	
temp	1/27/2021 9:04 PM	File folder	
.gitignore	1/20/2021 8:20 PM	Text Document	1 KB
config.txt	2/22/2021 2:06 PM	TXT File	1 KB
CprE 381 Toolflow manual.pdf	1/20/2021 8:20 PM	Chrome HTML Do...	205 KB
Mars_CPRE381_SIMD_v1.jar	1/20/2021 8:20 PM	Executable Jar File	11,445 KB
modelsim_framework.do	1/20/2021 8:20 PM	DO File	1 KB
SynthesisFramework.exe	2/22/2021 2:03 PM	Application	6,068 KB
TestingFramework.exe	2/22/2021 1:58 PM	Application	6,068 KB
version.txt	2/22/2021 2:37 PM	TXT File	1 KB

1. Copy all the source files from your processor into the src folder. There should already be a file named tb.vhd, you do not need to edit this file, and it should not be removed.
2. Double-click on TestingFramework.exe



3. You will be presented with a simple GUI that shows a few options, as well as all the assembly files stored in the mips folder.
4. Selecting the checkboxes of the assembly files selects the programs that the framework will attempt to run printing the output to the command line.

Options

There are several options that the testing framework utilizes to help debug and simplify the process.

- Compile
  - This option compiles all the files in the src folder and allows for quick modifications.
- Timeout
  - This option specifies how long in seconds the program should run before stopping in the case of an infinite loop.
- Max Errors
  - This options specifies the max number of mismatches between Mars and your processor that are accepted before the simulation is stopped.

Troubleshooting and FAQs

Q: I defined some types in a separate file, so now the program won't compile unless that file is compiled first. How do I fix this?

A: Then copy the entire contents of the VHDL file with the types onto the top the file with the error. Delete the types file:

```
use ieee.std_logic_1164.all;

package mytypes is
    type vector_32_array is array (0 to 31) of std_logic_vector(31 downto 0);
end mytypes;

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all
```

Q: My processor had an issue. How can I see the waveform to debug the issue?

A: There is a file "temp/vsim.wlf." This is a ModelSim waveform file. You can open it in ModelSim with file->open and setting "Files of Type" at the bottom of the window to "Log Files (*.wlf)"

Note: If you run multiple files at a time, the .wlf file will be from the last simulated assembly file.

Q: There is a specific signal I would like to look at in the waveform file that is not already there. How do I add it?

A: By default, the framework adds only the top-level signals. If you need to add additional signals, you can copy the commands to add the waves to the top of "modelsim_framework.do"

Synthesis Framework

The goal of the Synthesis Framework is for students to be able to get timing data from mapping their processors onto real hardware. This is beneficial for analyzing critical paths and ways to improve performance.

Getting Started

Before running the Synthesis Framework, please ensure that your processor works correctly in the Test Framework. The Synthesis Framework uses the VHDL files in "src." If the Test framework runs correctly, the simulation framework probably will also.

There are no options or settings in the simulation framework; you should be able to run it by double clicking "SynthesisFramework.exe". For a single cycle processor, the simulation should take around an hour to run on the lab computers and 2.5 hours to run on VDI. Please plan your time accordingly. Once the framework is complete, the results should open in Notepad, but are also in "temp/timing.txt."

Troubleshooting and FAQs

Q: My processor compiles correctly using ModelSim, but does not compile correctly here. Why is this?

A: This uses Quartus to compile VHDL instead of ModelSim, and thus has slightly different rules. The compiler errors are generally readable should point you to what the issue is.

Portability

This framework is supported for VDI on the physical computers as well as the virtual computers.

Additionally, there is a "config.txt" file that allows you to modify the file paths of the major components for custom installs.

For Installation on personal computes the following requirements must be met:

- Window 64 or 32 bit.
- Python 3.7.0 installed.
- Modalism PE Student edition installed.
- Quartus Prime Student edition installed.