Out: September 9th, 2016
Due: September 23rd, 2016

Goal

To become familiar with using Questa to compile a testbench, run a simulation, and analyze results. To learn how to use SystemVerilog to test your ROM and RAM designs.

Procedure

1. Log into a lab machine.
2. Make sure that you are working on your cabinet drive. Make a CME_435 subdirectory and copy ROM.zip (included in this lab package) to this directory.
3. Unzip the files creating a directory called ROM in your "cabinet" drive directory structure under the CME_435 directory. If you have both a "folder" directory and a "cabinet" directory, work in the "cabinet" directory. If you only have a "folder" directory, work in the "folder" directory. Normally you would be using your own Program_Memory_ROM.v that you created in Assignment #1, but because Questa was unavailable until recently a ROM file has been provided for you.
4. Questa should be accessible from the College of Engineering Computer Center Remote Application website (https://www.engr.usask.ca/engr-apps). You will need to provide your NSID and password in order to use the website. Note that you will have to include the usask domain with your NSID (i.e., usask\abc123). When the virtual desktop comes up double-click the Questasim shortcut to start the program. When asked, click "close" on the Jumpstart Welcome screen.
5. Change your working directory within Questa to your ROM directory. You can use Windows command-line commands pwd (see where you are), dir (view the contents of the current directory), and cd <path> (change directory). You can also browse folders in the File->Change Directory dialog.
6. As a general note, the Questa GUI truly is a front end to the simulation core. That is, the GUI menus all have a command-line equivalent. We will typically rely on the command line more frequently, as it's often quicker and easier to use. Likewise, managing .mpf project files within the GUI can be quite cumbersome, so we'll frequently make use of .do files (i.e. Questa script files) to perform compilations and launch simulations instead of managing project files in an .mpf.
7. At the command prompt, enter:

   ```
   QuestaSim> vlib work
   ```

   This creates a top-level working directory for vlog to compile into.
8. Now enter:

   ```
   QuestaSim> do compile_all.do
   ```

This will compile all of the source files in the testbench, including the DUT. Take a quick look at the .do file; you can see that it's a simple way to manage the files to be compiled. There should not be any errors or warnings with this operation.

9. Now launch the simulation:

   ```
   VSIM> do run_test.do
   ```

   This additionally compiles the test.sv file, and then launches the simulator on the top-level design unit (work.top). Compiling the SystemVerilog test file separately from the rest of the testbench + DUT environment is standard practice. Eventually we will have testbenches with multiple different test files, and we can only compile and run one at a time.

10. Create a wave.do script:

    - We will create this script to automatically set up a signal layout in the Wave results window. Once we have all of the DUT signals arranged, we can simply run the script as part of running a test in order to set up our waveform view rather than having to re-do the layout every time.
    - If the Wave window is not currently visible, enter:

      ```
      VSIM> view wave
      ```
    - If the Objects window is not currently visible:

      ```
      VSIM> view objects
      ```
    - In the sim window (mid-left portion of the screen), expand the hierarchy under 'top', then under 'env', and left-click on 'DUT'. A list of signals familiar to you should show up in the Objects window. These are the signals that we want to watch during the simulation.
    - Drag and drop all of the signals from the Objects window to the Wave window. (You can Shift+click and CTRL+click to select and drag multiple signals). The signals can be reordered within the Wave window with drag & drop as well.
    - Dividers can also be inserted between groups of signals. Right-click in the leftmost grey space of the wave window and select Insert Divider. Label it as "Environment Signals". Drag the 'clk' signal under this divider.
    - Add additional dividers and drag the appropriate signals under them to distinguish between inputs, outputs, test, and internal signals. Additional dividers can improve readability and debugging efficiency.
    - Some signals are better represented as binary, hexadecimal, decimal, unsigned, etc. Also, some signals may be better interpreted by an analog-step than a literal interpretation of their value. Right-click on a signal in the wave window and select Properties to access these options. When viewing a signal as analog-step (superior to analog-interpolated) it is necessary to find an appropriate height, offset, and scale. Experiment with these parameters until you are comfortable with the process. Each time you add a new signal to the wave window you should save and rerun an initial short simulation to get a visualization of how the simulation is drawing the signal.

- Save the Wave window layout as wave.do. Click anywhere in the Wave window to give it focus, then click File->Save Format… and OK. Use the default name wave.do.
- Now you can have your waveform layout automatically appear every time you launch a simulation. Edit test.do in any text editor, and uncomment the "do wave.do" line by removing the '#' comment marker from the beginning of the line.
- Close the Objects window to make more room for the Wave window.

11. Exit the simulation mode:

    VSIM> quit –sim

    Note the '- sim' switch. If you simply enter 'quit', Questa will prompt you to close Questa down entirely. Usually this isn't what you want to do.

12. Re-launch the simulation:

    QuestaSim> do test.do

    Note that your waveform layout now appears automatically.

13. Run the simulation. You can issue the 'run –all' command if your test program includes a $stop command to stop the simulation at a certain point rather than having it run forever, or you can specify the runtime explicitly (e.g. VSIM> run 200us). The supplied test does have a built in $stop command, so:

    VSIM> run –all

    NOTE: Answering YES to "Are you sure you want to finish?" will terminate the Questa software. In most case you do NOT want to do this…

14. Check to make sure everything is working properly by zooming in the waveform window. Right click in the black region where the waveforms are displayed and select the appropriate command (i.e. Zoom Full). The lab package includes a PDF of the simulation waveforms (QuestaSim_ROM.pdf) that you can compare against.

15. When you are satisfied, export the waveform to a BMP image: Make sure the Wave window has focus, then File->Export->Image. Enter quit –sim to end the simulation.

16. Now it is time to test out the RAM module. Extract the RAM folder from RAM.zip into your CME_435 directory, then repeat steps 7-13 to compile the testbench, set up a waveform layout, etc.

17. In the RAM testbench, test.sv is incomplete. It will be up to you to implement the READ and WRITE tests (see the TO DO comments in the file). These tests should be set up to reproduce the stimulus (and results) shown in QuestaSim_RAM_Read.pdf and QuestaSim_RAM_Write.pdf, respectively.

    Even though you have little knowledge of SystemVerilog at this point, it should not be too difficult to complete this task if you use the "initial" block from test.sv in the ROM testbench as a guide or template. All the constructs that you need are shown in the

ROM testbench's test.sv (do note that the RAM driver is invoked as env.input_drvr.RAM_signals(address, data, wren)).

18. Run the two simulations. Switch between the READ and WRITE tests by setting the test_type variable in test.sv, then re-running run_test.do.
19. When you are satisfied with your implementation of the READ and WRITE tests, export a waveform for each run to a BMP image.

Deliverable

Completed test.sv from the **RAM** testbench, and the saved waveform outputs from the ROM test, the RAM read test, and the RAM write test.

Package the required files in a zip archive that includes your name in the filename, and email to the course lab instructor Chandler Janzen (chandler.janzen@usask.ca) by midnight on September 23rd.

Some Notes and Gotchas

ALWAYS work in a subfolder of your cabinet folder.

It is possible to have a compile that works successfully but the simulation load does NOT load successfully. Make sure you check out the load messages. Warnings show up in blue. Errors show up in red.

If you change a source file make sure you save it before doing another compile. Sounds obvious but you WILL forget to do it at least once.

Questa has command history capabilities (for everything but compiles). After having used a command it will be available in the command history. Activate (by clicking) in the Transcript window and then use the up-arrow to select the command to run.

Questa does not know about simulation time expressed as seconds (s). Microseconds (us), nanoseconds (ns), etc. are OK but NOT seconds.

If you change the signals in the wave window make sure you do a save (to the wave.do file) to have the same setting the next time you simulate your design.

Do NOT forget the "initial begin" line in your test program. This is noted in the example test program.

Always pay attention to your current working directory within Questa when switching between projects.