CME341 Preamble for a Remotely Written Final Exam

About a Remote Delivery Final Exam

The final exam will set up as an assignment on Canvas called "Final Exam". The assignment will hidden from the student until the scheduled start time for the exam. At that time the students will be able to download the exam as well as the files needed for the exam.

The worth of the questions on the exam are not defined *a priori*. Worth is only assigned to questions that have been answered correctly. For purposes of grading, each part of a multipart question is considered to be a separate question. The worths assigned to correctly answered are scaled as follows: The first question (or a part in a multipart question) answered correctly is worth 10. The second, third, fourth and fifth question answered correctly are worth 9, 8, 7 and 6, respectively. The sixth and subsequent questions answered correctly are worth 5.

To make it absolutely clear, the formula for the worth of the n^{th} question answered correctly is

$$\begin{cases} 11-n; & 1 \le n \le 5 \\ 5; & 6 \le n \end{cases}$$

The accumulated worth of answering K questions correctly is

$$\begin{cases} \frac{K(21-K)}{2}; & 1 \le K \le 5 \\ 15+5K; & 6 \le K \end{cases}$$

Please note:

1. For purposes of grading and only for purposes of grading, each part within a numbered question, e.g. 1a), 1b), ..., is considered to be a separate question.
2. The ROM for each numbered question is to be initialized with the .hex file specified in the question.
3. Each numbered question is stand alone. However, if a numbered question has multiple parts, for example 1a), 1b) and 1c), then
   (a) Program memory (i.e. the ROM) is initialized with the same .hex file for all parts.
   (b) Each part builds on the previous part. This means modules modified to answer part a) are further modified to answer part b).

Pre-Exam Preparation

Prior the entering the exam place copies of your prototypes for the microprocessor in a separate folder, which will be referred to as `prototypes_final_exam`.

Modify the prototypes that were copied to folder `prototypes_final_exam` and prepare Quartus and Modelsim-Altera projects in accordance with the instructions below.

1. Modify the prototypes of the program sequencer, instruction decoder, computational unit and the top level micro processor as follows:

   Program Sequencer:

   Make the program counter an output called `pc` so that it can be used in the test bench as part of the scrambler.

   Make a new 8-bit output called `from_PS`. This output is intended to convey signals that result from changes made to the program sequencer to the test bench. `from_PS` is wired into the testbench scrambler and affects the answer code. On specific questions you will be asked to connect a signal that was created in the program sequencer to `from_PS` to get that signal to the testbench.

   Upon entering the exam `from_PS` is to be set to `8'H0`, but sometime before the exam the connection path from the program sequencer to the testbench should be tested by connecting the program counter to `from_PS`, i.e. `from_PS` $= pc$.

   Instruction Decoder:

   Make the instruction register an output called `ir` so that it can be used in the test bench as part of the scrambler.

   Make a new 8-bit output called `from_ID`. This output is intended to convey signals that result from changes made to the instruction decoder. Upon entering the exam `from_ID` must be made equal to `8'H00`. However, sometime prior to the exam the connection path from the instruction decoder to the testbench should be tested by setting `from_ID` to `reg_enables[7:0]`, where the bits in `reg_enables[7:0]` starting from the least significant are the enables for data registers x_0, x_1, y_0, y_1, r, m, i and dm.

   Also make the entire 9 bits of `register_enables` an output of the instruction decoder. The register enables are connect to the testbench so they can be displayed and used for debugging circuits built in the exam. The register enables are not connected to the scrambler and do not affect the accumulator output.

   Also decode the no-operations `NOPC8, NOPCF, NOPD8` and `NOPDF` and make them outputs of the instruction decoder. Having these signals run through the microprocessor to the testbench is extremely helpful in the exam, both for building and debugging the circuits. These no-operation signals `are wired into the scrambler in the test bench and affect the accumulator output.`

   Computational Unit:

   Make all the data registers outputs.

Make a new 8-bit output called `from_CU`. This output is intended to convey signals that result from changes made to the computational unit. `from_CU` is wired into the testbench scrambler and affects the answer code. On specific questions you will be asked to connect a signal that was created in the computational unit to `from_CU` to get that signal to the testbench.

Upon entering the exam `from_CU` must be made equal `8'H00`. Again, some time prior to the exam, to test the connection path from the computational unit to the testbench `from_CU` should be set to `from_CU` = $\{x_1, x_0\}$.

Microprocessor:

In addition to the original outputs make the following signals outputs of the microprocessor:

(a) `pm_data`

(b) `pc`, `from_PS` and `pm_address`

(c) `ir`, `from_ID`, `register_enables`, `NOPC8, NOPCF, NOPD8` and `NOPDF`.

(d) `from_CU`, all of the microprocessor's 4-bit registers (Note that `o_reg` is already an output.) and the zero flag.

2. After making the changes as described above make sure that `from_PS` = pc, `from_ID` = `reg_enables`$[7:0]$ and `from_CU` = $\{x_1, x_0\}$.

3. Make a Quartus project for the micro. This project will be used in the final exam. Use A Cycloneive FPGA. Initialize the ROM with file `final_program_preamble.hex` (downloaded from the class website).

   Compile and remove all errors.

   It is pointed out that the first time this project is compiled, the compiler will generate a warning indicating the .hex file does not fill the entire memory. That warning is correct, but can be ignored. The warning will only appear the first time the project is compiled.

4. Copy all the .v, .do and .lst files from the class website folder `preamble_files_for_final_exam` to the directory where Quartus places the .vo file for the microprocessor.

   Then set up a Modelsim-Altera project called `final_exam_testbench` in this directory. Include `final_exam_testbench.v` and the .vo file for the microprocessor in the project.

5. Compile `final_exam_testbench` and the .vo file for the microprocessor in the project.

6. Load the simulation using libraries "cycloneive_ver", "altera_mf_ver" and "altera_ver".

7. Execute "do final_preamble_wave.do" in the transcript window to set up the wave window.

8. Run the simulation.

9. After running the simulation observe the transcript window. There will be three one-line messages. The first two will list the value of `accumulator_output` for two successive uses of seed `8'HAA`. The values of `accumulator_output` are taken at times 310 μs and 630 μs, which are the times `counter_full_bar` is low for successive seeds. If your microprocessor is working properly both values for `accumulator_output` will be 16'H90eb.

   It is a very important that seed `8'HAA` be used in two successive input cycles (i.e. two 320 μs cycles) and that the resulting `accumulator_output` be 16'H90eb both times. For the test bench used in the exam generates a sequence of signals that repeat every 320 μs. One period of the signal is referred to as an input cycle.

   The seed `8'HAA` is used for the first two input cycles to check the response of the student's circuit to `sync_reset`. If `accumulator_output` has a different value on the second cycle, then the circuit has not been properly reset.

   If the circuit is not properly reset it is possible to get the correct value for `accumulator_output` for the first occurrence of seed `8'HAA` and the incorrect value for `accumulator_output` for the exam dependent seed.

10. If your microprocessor is working properly `accumulator_output` should read 16'H1f5f when the exam dependent seed is `8'HFF`, i.e. when `seed == 8'HFF`. The correct answer for exam dependent seed `8'HFF` must be obtained from the waveform in the wave window.

    The exam dependent seed that you are to use in the exam will be given in the answer sheet. You will have to change the exam dependent seed in the testbench.

    During the exam the value of `accumulator_output` for the exam dependent seed will be printed as the third one-line message in the transcript window.

11. To help get your preamble working a wave window data set called `final_preamble_1.wlf` is provided in the same folder as the other preamble files. This data set can be viewed in the wave window by first closing the wave window if it is open and then issuing the command

    ```
    vsim -view gold=final_preamble_1.wlf
    ```

    followed by the command

    ```
    do final_preamble_wave.do
    ```

    Once the data is displayed it can be manipulated with the display tools, which of course includes the zoom.

    To make comparison reasonably easy the gold waveform should be displayed in a second wave window. Unfortunately, the command that opens a second wave window, i.e. "`view wave -new`" is not supported on the version of ModelsimAltera we have.

    However, the gold waveforms can be viewed in a second wave window by setting up a second ModelsimAltera project and opening ModelsimAltera a second time. The second project

should be in a separate file folder and needs to have two files in it: `final_preamble_1.wlf` and "`final_preamble_wave.do`". After opening the second ModelsimAltera project execute the two commands above to display the data.

12. Modify the program sequencer, instruction decoder, and computational unit to make `from_PS` = `8'H00`, `from_ID` = `8'H00` and `from_CU` = `8'H00`.

    With these changes the micro is ready for the final exam.

13. Recompile etcetera. Since `from_PS`, `from_ID` and `from_CU` are hardwired to ground, the Quartus compiler will issue a warning that 24 signals are stuck at ground.

    After running the testbench simulation, `accumulator_output` should read 16'H4463 both times `seed == 8'HAA` and 16'H75A2 when `seed == 8'HFF`.

    While unnecessary, for ease of mind the correct wave window data set for the final exam ready microprocessor has also been provided on the website. It is file `final_preamble_2.wlf`.

14. **Save copies of the exam-ready prototypes in a folder called `copy_of_prototypes_final_exam`. These copied files will have to be recopied during the exam to the Quartus project used for the exam as every question in the final exam is based on a microprocessor that uses these prototypes as a starting point**.