CME341 Preamble for the Final Exam: V2023

About the Final Exam

The exam will start in the classroom assigned on the exam schedule. Students will be given the exam booklet at the time the exam is scheduled to start and also given time to ask clarifying before the exam officially starts. The length of the question period depends on the number of questions asked, but is usually from 15 to 30 minutes.

During the question period the students will be informed where they will write the computer part of the exam. Usually the student numbers are such all the students will not fit into a single lab. During the question period student will be informed of the lab where they are to write the exam. At some point the invigilator will declare the question period over and ask the students to move the computer lab assigned to them.

Once in the assigned lab the students are to immediately down load the exam files from the class web site, start up Quartus, start up Questa, open the Questa testbench in an editor and change `exam_dependent_seed` to the value that has been assigned and then wait for the instructor to announce "begin the exam". After the instructor has established that everyone in the lab has downloaded their files and has both Quartus and Questa running, the instructor will announce "you may begin the exam". The exam will end 3 hours after the instructor announced "begin".

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 values 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

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

For purposes of grading and only for purposes of grading, each part within a question is considered to be a separate question.

Each question is stand alone. However, if a question has multiple parts, for example a), b) and c), then part b) builds on part a) and part c) builds on part b).

Pre-Exam Preparation

Prior the entering the exam place copies of your prototypes for the microprocessor in a separate folder on their H drive - say a folder called `prototypes_final_exam`.

Modify the prototypes that were copied to folder `prototypes_final_exam` and prepare Quartus and Questa 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 first to test the connection path from the program sequencer to the testbench it is to be connected to the program counter, 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, to test the connection path from the instruction decoder to the testbench it will first be made equal 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`, but to test the connection path from the computational unit to the testbench it is first 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`. It can be downloaded from the class website in folder `preamble_files_for_final_exam`.

   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 (or .sv), .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 Questa project called `final_exam_testbench` in this directory. Include `final_exam_test` 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 $320\,\mu$**s 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 $320\,\mu$s. 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.

    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. However, for the preamble, where the exam dependent seed is `8'HFF`, the third one-line message is a warning that the exam dependent seed is still `8'HFF`.

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 added to the wave window that has your signals displayed. This is done as follows:

    (a) Download file `final_preamble_1.wlf` from the class website and place it in the directory containing your testbench. This Waveforms Log File (i.e. .wlf file) contains the instructor's preamble waveforms.

    You are able to add the waveforms in `final_preamble_1.wlf` to your wave window by following the instructions below.

    (b) Pull down the file menu in the main Questa window and select "Datasets..". This will open a window called "Dataset Browser".

    If you have already ran your simulation, the browser will show a directory path to your simulation. Otherwise the window will be blank.

    (c) Click "open". This will bring up another window called "Open Dataset". Browse to select `final_preamble_1.wlf`. Selecting the file will also auto-fill the "Logical Name for Dataset" box. Change the Logical Name to `G_std`, which is short for gold standard.

    The Logical Name was changed to make it short, because it will be prepended to all the signals displayed in the wave window.

    Then click OK.

    (d) Close the "Dataset Browser" window by clicking on Done.

(e) After the "Dataset Browser" window closes, another tab named `G_std` will be added at the bottom of the window shared by "project", "library", etc. Select that tab, if it hasn't been automatically selected.

When tab `G_std` is selected, the signals in `final_preamble_1.wlf` are place in the object window. They will have the same names as the signals in your simulation, except they will be prepended with `G_std:`.

They can be added to the wave window in exactly the same way the signals in your simulation file were added. The easiest way to add the signal is to run the command: `do final_preamble_wave.do`.

If you want to add more signals from your project to the wave window, you must make sure the "sim" tab is selected so that the signals in the "objects" window are from your simulation.

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**.

Post-Exam Requirements

After the instructor has signaled the time for writing the exam has expired and has collected your answer sheet, you **must** upload all of the .v, .sv and .hex files used to answer the questions on the exam to a specified depository. The depository will be either specified on the exam question booklet or announced by the instructor.

Warning: When grading the exam, the instructor may wish to corroborate or verify the answers on the answer sheet. This will be done by examining the .v or .sv files that were uploaded to the depository. Should the .v of .sv files be missing no credit will be given for the answer.