18-240: Structure and Design of Digital Systems

Electrical & Computer ENGINEERING

HW7 [4 problems, 64 points]

Covers lectures L14 – L15 *Due: 6 November 2023*

Homework sets are due at 5:00PM on the due date. Upload your answers, to Gradescope by then. No late homework will be accepted. Remember, we let you drop two homework assignments over the semester.

By now, you know how to use **`handin240`** to create the files for the Gradescope upload. Refer to the course wiki or to previous homework if you need refresher instructions.

Discussions about homework in small groups are encouraged — think of this as giving hints, not solutions, to each other. However, homework must be written up individually (no copying is allowed). If you discussed your homework solutions with someone else, either as the giver or receiver of information, your write-up must explicitly identify the individuals and the manner information was shared.

You must show details of your work. There is no credit for just writing down an answer.

Drill Problems [26 points]

1. [26 points, Lecture 14] For this problem, design a datapath to determine if a string located in a memory bank is a palindrome. A palindrome is a pattern that reads the same when read forward and backward. Famous example palindrome strings include "Able was I ere I saw Elba" and "Ma is as selfless as I am." Spaces and punctuation are normally ignored when viewing a palindrome. However, to keep from complicating your datapath overmuch, every character will count.

   Your system will be given three inputs:

   - **`StringAddress`**: a 16-bit value that is the memory address of the first byte of the potential palindrome string.
   - **`StringLength`**: a 16-bit value that is the length of the string in bytes. You are guaranteed that the string will not go beyond the bounds of the memory bank (i.e. **`StringAddress`** + **`StringLength`** $< 2^{16}$).
   - **`Ready`**: indicates that the memory bank is loaded with the test string and that **`StringAddress`** and **`StringLength`** contain good data. When asserted, your datapath should begin its calculation.

   Your system will generate 2 outputs:

   - **`Done`**: when asserted, indicates that the palindrome check is complete.

- **`isPalindrome`**: when asserted, indicates that the test string is, in fact, a palindrome.

Both outputs should be asserted until the clock edge when **`Ready`** is asserted again.

The memory bank is a 64Kx8 bit memory, with the same interface discussed in class (combinational read, synchronous write) and in your library.

Part A: Data [2 points]

List all the data elements that you will have to keep in registers. These are the "variables" of your computation.

Part B: Transformations [2 points]

List all of the transformations needed by the datapath. This should be a list of $a \Leftarrow b$ sort of statements about how registers will get new information.

Part C: Datapath [10 points]

First, **neatly** draw a datapath to accomplish this problem. Clearly indicate the width of any multi-bit signals in your datapath. Clearly label the names of control signals to be driven by the control FSM. Clearly label the names of status signals examined by the control FSM.

You are allowed to use any components from your library. You do not need to minimize the datapath. You do not need to minimize the execution time. We're going for "simple" and "clear" in this problem.

Part D: Control/Status Points [2 points]

Since we want you to be *very* clear about how the FSM is going to talk to the datapath, we're going to ask for the same stuff on the diagram again, in an explicit set of lists:

- Give the list of control points (i.e. signals from control FSM to datapath).
- Give the list of status points (i.e. signals from datapath to control FSM).
- For each input from the external world specify a destination as FSM or datapath.
- For each output, does it get generated by your FSM or your datapath?

Part E: Control FSM [10 points]

Neatly draw the state transition diagram for a control FSM (you choose: Mealy or Moore style, whichever works).

Hint: the simple RTL example at the end of Lecture 16 is an excellent place to start. It has a memory with the same control signals (combinational read, synchronous write); it has the required bus transceiver (bidirectional/tristate) block to talk to the memory; it has arithmetic logic to deal with the address input for the memory; it has some logic to "to work on" the data

read from the memory; it has a Mealy style FSM to control everything, in particular, doing both memory reads and writes.

▷ **Submit your work in your PDF file.**

Non-Drill Problems [38 points]

2. [12 points, Lecture 15] Memory systems are created from several memory (or other) components, each using some subset of an overall address range. In this problem, you will build a memory system.

   Your overall memory system will have a 32-bit address range of single byte memory locations. It will be constructed from three different memory components:

   (a) A "Read-only" memory from address **`32'h0000_0000`** - **`32'h0FFF_FFFF`**. Use your **`Memory`** component, but make sure that a write would not change any stored values. We will ignore how it gets initialized for this problem.

   (b) A "Code and Data" memory from address **`32'h1000_0000`** - **`32'h8FFF_FFFF`**. This memory should be readable and writeable.

   (c) A "Interrupt Table" memory from address **`32'hF800_0000`** - **`32'hFFFF_FFFF`**. Readable and writeable.

   You will notice that there is a gap in the coverage (from address **`32'h9000_0000`** - **`32'hF7FF_FFFF`**). That's deliberate, as the customer doesn't want to pay for any more memory components. Make sure this gap remains in your design.

   How does information get into and out of your memory system? In almost the same way as information gets into and out of individual memory components. An address bus carries the address, a data bus carries the data and read-enable and write-enable signals send control information. The only difference is that we are going to assume the user has registers connected to the address and data buses and we will make those part of the system. We are, however, going to simplify things a bit and assume the user can magically get values into and out of those registers without you having to worry about how that happens.

   Design the memory system, including the registers on the address and data buses. The **`re`** and **`we`** signals are inputs to your system. Show the three memory components, plus any other components necessary to make this work. For instance, how are you going to choose which memory gets written to or read from?

   ▷ **Submit a neatly drawn schematic in your PDF. Ensure that all mult-bit signals have widths specified. Make sure you have clearly labeled all components.**

3. [20 points, Lecture 15] Design a datapath to calculate a final grade for some non-240 course. The grade is calculated from a weighted average, based on individual homework, lab, exam and class participation grades.

   The grading algorithm is somewhat like 18-240. Individual homework grades are summed to get an overall homework average. Since general division in hardware is difficult, let's image that the individual grades have already been divided by the number of homework assignments. All you have to do is add all the homework scores to get an overall homework grade.

   Ditto for the lab scores, exam scores and class participation scores.

   Then, the overall homework grade is multiplied by 25%. The overall lab score is multiplied by 25%, the overall exam score by 37.5% and the class participation score by 12.5%. The resultant products are then all added together to get an overall class score.

   If the overall class score is above 90%, then the student gets an A grade. B, C, D grades are assigned for each 10% lower, with R grades for below 60%.

   Whew! That seems rather straightforward, though how to do some of the math may take you a few minutes to figure out. You should be able to accomplish all of this without actually multiplying, just by shifting and adding and comparing.

Interface

Your datapath will have four inputs, in addition to **clock** and **reset_L**. The **score** input is a 7-bit value with the grade for an individual work. Another input is **score_type**, a two-bit value which specifies what type of grade this work is. It will be **00** for a homework, **01** for a lab, **10** for an exam and **11** for class participation. **start** will be asserted for a single clock edge at the beginning of the computation. Imagine that it signifies the beginning of the semester. Any time it is asserted, you should zero out the state of the previous computation and start again. **grade_it** will be asserted to indicate that **score_type** and **score** hold a valid grade that should be added to the computation. If **grade_it** is not asserted, those inputs should be ignored.

Your datapath will have outputs for each potential grade: **grade_A**, **grade_B**, **grade_C**, **grade_D**, and **grade_R**.

The module interface looks like this:

```
module hw7prob3
  (input  logic [6:0] score,
   input  logic [1:0] score_type,
   input  logic       start, grade_it,
   output logic       grade_A, grade_B, grade_C, grade_D, grade_R,
   input  logic       clock, reset_L);
```

You should follow the same design process that you did for problem #2. Figure out the data values, list the transformations, design the datapath hardware, list the control and status points,

design the FSM to manage the system, then write the SystemVerilog, using components from your library.

You can find a basic testbench on Canvas as **`hw7prob3_test.sv`**. Expand it to fully test your system.

▷ **Submit your code as a file named `hw7prob3.sv`. You should also include your design work in your PDF file.**

4. [6 points, Lecture 15] Lecture #15, contained a block diagram of a 1Gb DDR SDRAM chip. It is organized as 256M x 4-bits, with a multiplexed address bus.

   (a) How many address bits are there for this chip? You can try to interpret the block diagram, or just figure it out from the 2nd sentence above. (1 point)

   (b) I just bought a new MacBook Pro with 16GBytes of memory. If the entire memory unit of the laptop consisted of nothing but this chip, how many chips would be required. (Spoiler, it is a lot). (1 point)

   (c) Sketch a partial schematic to show how a 64-bit physical address bus and 8-bit data bus would be connected to the collection of chips required by your answer to part B. Obviously, I'd rather not see every chip. Instead, *sketch* the schematic, showing just enough to prove you know what components are needed and how they would be connected. Also, the multiplexing part is more complex than I intend for this question (though you still could figure it out). So, don't worry about the address lines on each chip. Instead, focus on how the chip select ($\overline{CS}$) is connected. (4 points)

   ▷ **Submit your work in your PDF file.**