ELEX 2117 : Digital Techniques 2
2020 Winter Term

Additional Hints for Lab 5

Follow the VHDL Coding Guidelines

Read and follow the coding guidelines; they will save you time. First, because following these practices will result in fewer errors and secondly because you will need to rewrite your code if don't.

Do the Pre-Lab First

Draw the detailed block diagrams as required by question 1 of the pre-lab before trying to write the VHDL. Add signal names to the block diagrams. You have to draw these diagrams anyways so this is not wasted time.

If the diagrams are correct, it should be straightforward to write VHDL that is functionally correct. If you do it in the reverse order, (VHDL first) you are more likely to lose your way.

Implement Efficiently

Implement bottom-up. Implement and test only as much as you are confident you can get right.

This is, by far, the most important skill to learn if you want to be able to design and implement complex systems.

In this lab, for most students, this will be one register and the associated combinational logic (one of the small blocks given on page 3).

If you test a complete solution and it doesn't work you won't know where to look for errors. If you test a small block and it doesn't work you know the problem is with the part you were testing.

You can define as many test outputs as you want. Slow signals can be viewed on LEDs, fast ones can be viewed on a 'scope.

Follow the Examples

Follow the examples in the lecture notes. They are not complete so you can't just copy them. But if you understand them you will be able to modify or extend them to do what you want.

Examples of how to declare and use an array as a 7-segment decoder are given in the subsection titled "Modeling Memory with Arrays" in the "HDL Idioms" lecture notes.

Avoid Surprises

Make your code as regular and predictable as possible. The human brain is very good at detecting anomalies in regular patterns and this makes it easier to spot errors. For example:

- Indentation should reflect the structure of the code.
- Signal names should follow predictable patterns. An example is adding suffixes as recommended in the Coding Guidelines document. This will help you spot errors.
- Although the ordering of I/O ports doesn't affect functionality, it will be easier to spot mistakes if you use a consistent order (e.g. all inputs first, clocks & reset last, alphabetical order, or whatever makes sense to you).
- Although the order of concurrent statements doesn't matter, it will be easier to read and understand your code if signals in your code "flow" from top to bottom. This is similar to drawing schematics so that signals flow from left to right.
- If the lab notes suggest specific signal names and pins, you can save yourself time by using the same ones.