

ECE 464/520 - Sample Practical Test

Dr. P. Franzon

Instructions

You have three hours to complete this test. It is an open book test but you are not allowed to consult with anyone during the test. Turn off zephyrs to your computer (kill the zwgc process), do not have a mailer up, and turn off your cell phone. Any observed attempt to communicate with someone else will result in a failure being recorded for you. You are allowed to ask the TA to clarify the question but not to help you with debugging, etc.

Please be aware that though every section will have a different question, that the complexity is roughly the same each time.

At the end of the three hours you are to show the following to the TA:

- Your design and verification code.
- The simulation results from pre-synthesis verification.
- The code synthesizing in Design Compiler
- The netlist from Design Compiler
- The view_command.log file from Design Compiler

You are not required to do post-synthesis simulation.

Grading Instructions

This is an "all of nothing" exam. I.e. The grade is either 0 or 100%. You get 100% by satisfying the following:

- The code meets the specification below. Specifically it matches the waveform specified.
- The code simulates (pre-synthesis only) correctly.
- The design compiles correctly with no HDL or RTL errors or warnings, no unintended latches, and it meets your specified timing.

If you fail this exam, you will be permitted one retest at a later date. If you fail that, you get 0 for this course component.

Sample Question

Design and implement a simple Checksum error detector to work on a packet-based protocol. A packet of data usually contains a header, the data itself and a checksum. In this case the packet is organized as follows:

```
Byte 0 : 0101 0101 \\This byte ONLY indicates that a correct
packet is following
Bytes 1-7 : Data
Byte 8 : Checksum, ie. sum of data rounded down to 8-bits
```

For example, the following packet has a correct checksum of 05 ($01 + 01 + 01 + 01 + 01 + 01 + FF = 05$ when truncated to 8 bits):

```
55 01 01 01 01 01 01 FF 05
```

while the following example has an error in the data, as the checksum is incorrect:

```
55 01 01 02 01 01 01 FF 05
```

Design and code in Verilog a synthesizable module that will detect the packet start and work out if the packet is correct or not. The I/O are:

```
module checksum (clock, reset, data, error);
  input clock;
  input reset; \\ reset active low
  input [7:0] data;
  output error;
```

Error is to be set low by the global reset and is to remain low until an error is detected and then is to remain high until a new packet arrives.

For example:

```
clock 0101010101 01 01 01 01 01 01 01 01 01 01 01 01 01 etc.
reset  1 0 1
data   0 0 0 0 55 01 01 02 01 01 01 FF 05 FF AA 55 01 01 02 01 01 01 FF 05
error  x x 0 0                0 0 1 1 1 0                0 0
```