**Chapter 2, Data Types homework 3**

The purpose of this homework is to practice using user defined types to test a configuration register design. A configuration register is a bank of registers that is used to store setup information for the system (i.e. the configuration). They can be quite complex, with read only bits, write 1 to clear bits, and varying bit-widths. The configuration register design you are to test is very simple. All registers can be read and written and are 16-bits wide. The design contains 8 registers as defined in Table 1 along with their reset values. All bits are writable. The reset is active high.

|  |  |  |  |
| --- | --- | --- | --- |
| **Register** | **Width** | **Address** | **Reset Value** |
| adc0\_reg | [15:0] | 0 | 16'hFFFF |
| adc1\_reg | [15:0] | 1 | 16'h0 |
| temp\_sensor0\_reg | [15:0] | 2 | 16'h0 |
| temp\_sensor1\_reg | [15:0] | 3 | 16'h0 |
| analog\_test | [15:0] | 4 | 16'hABCD |
| digital\_test | [15:0] | 5 | 16'h0 |
| amp\_gain | [15:0] | 6 | 16'h0 |
| digital\_config | [15:0] | 7 | 16'h1 |

Table 1: Config Register Defines

Write a testplan describing how you are going to test the design. Then write a self-checking testbench to perform the testing you described in your testplan. Use a user defined enumerated type to enumerate the registers.  There is 1 bug per register. An encrypted version of the buggy configuration register code is provided. Compile this module just like an unencrypted module is compiled.

Identify the 8 bugs in config\_reg\_buggy.  For each bug, report:

**a) Design Input for Bug to Appear:**

*E.g. Write FFFF to register x*

**b) Expected Behavior:**

*E.g. Bit 15 of register x is writable to a 1*

**c) Observed Behavior**

*E.g. Bit 15 of register x cannot be written to a 1*

Remember, your bug report should contain enough detail for the designer of the configuration register to debug the problem.

Deliverables:

1. Test plan
2. Testbench code
3. Complete bug report for each register