Advanced Modeling -- Spring 2003 – Test 3 Solutions

1. (30 pts) Please refer to the attached Verilog module named ‘serialnrzi’. This is from a student solution whose gate level result did not match the RTL simulation.

a. The ‘always @(posedge clk)’ block violates the Verilog synthesis coding guidelines we discussed in class. What are these violoations? Show how this ‘always’ block should be written.

The always block used blocking assignments (‘=’) when it should have used non-blocking assignments (‘<=’). Just change all assignments to non-blocking assignments.

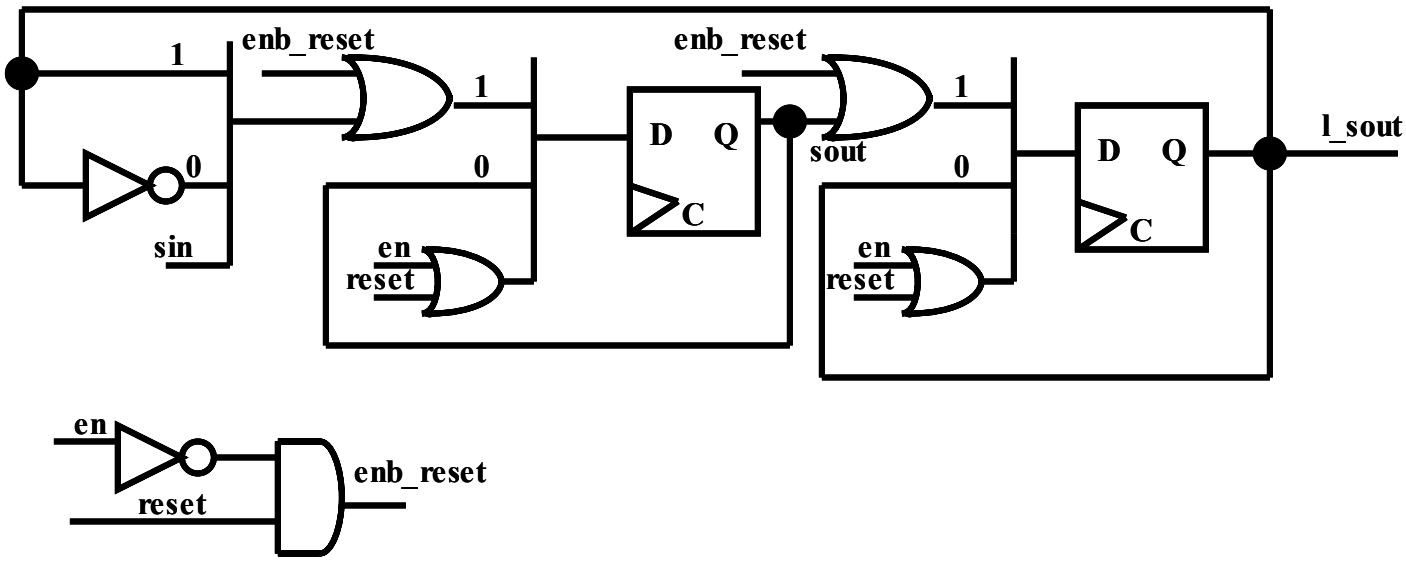
```
always @(posedge clk) begin
  if (reset)   begin
     state <= `S0; l_sout <= 1; sout <= 1;
    end
   else state = nstate;

   if (en) begin
     if (sin == 'b0) sout <= ~l_sout;
     else sout <= l_sout;
     l_sout <= sout;
   end
end
```

b. What problems can arise from violation of these guidelines. Give a SPECIFIC example, using the code in this example

Clearly, the l_sout value should be the previous value of s_out. However, because of the blocking assignments, it will be assigned the current value of s_out. In terms of hardware, what should be synthesized is two DFFs, with the sout DFF feeding the l_sout DFF. However, with this code only a single DFF will be synthesized, and l_sout and s_out will be the same output.

c. After your corrections, draw the logic synthesized for output ‘l_sout’ in terms of signals ‘reset’, ‘en’, ‘sout’, and ‘clk’.



2. (30 pts) Please refer to the attached Verilog module called 'fsm'. This is from a student solution whose gate level result did not match the RTL simulation.

a. The 'always @(state or start or flag)' block is supposed to implement the combinational logic associated with the finite state machine. This code violates the Verilog coding guidelines we discussed in class. Give specific examples from this code illustrating coding guideline violations (you do not have to give all violations).

The always block is supposed to be combinational logic, as such, it should only use blocking assignments ('='). Instead, it uses a mixture of blocking and non-blocking assignments – change all assignments to blocking assignments.

b. The 'always @(state or start or flag)' block also violates RTL synthesis coding guidelines that are true when coding in either VHDL or Verilog that will cause RTL simulation to mismatch gate-level simulation. Give specific examples of these violations, and list as many as you can identify.

There are no defaults for output values; this means that latches will be synthesized on the outputs which is incorrect since this should be a pure combinational block.

The sensitivity list should contain all signals on the RHS side of assignments if RTL simulation is to match synthesized netlist simulation. 'data', 'zcount', 'din' are missing from the sensitivity list.

c. The 'always @(state or start or flag)' block also some violates some basic digital design guidelines. What can you spot in this RTL code that will result in a non-functional gate level implementation. The answer to this question may overlap somewhat with your answer to part 'b', so I will also look for answers to this question in part 'b'. It would help to for you to think of the logic that will be synthesized.

This is supposed to be a combinational block of logic; the statement 'zcount = zcount + 1' will create a combinational feedback path that will oscillate. The same can be said for the statement that appears to be trying to do a shift operation 'data[6:0] = data[7:1]'.

Answer 8 of the next 10 questions, cross out the two questions that you do not want me to grade.

3. (5 pts) What is the difference between the NMOS and RNMOS built-in transistor primitives in Verilog?

RNMOS is the strength reducing transistor version of NMOS.

4. (5 pts) What construct in Verilog can be used to simulate a capactive storage node in a circuit?

   the trireg statement is used to simulate a wire with a capacitive hold value.

5. (5 pts) Describe the basic strength system in Verilog.

   The strength system has 8 values – 0 through 7, with the strongest strength being known as 'supply' and the weakest strength as high impedance.

6. (5 pts) What features of Verilog make it well suited for low level simulation problems?

Transistor and gate-level primitives are a part of the language standard.

7. (5 pts) In Verilog AMS, what do the 'idt' and 'ddt' operators provide?

idt is the time integral, ddt is the derivative with respect to time.

8. (5 pts) Give an example of two NATURES in Verilog-AMS.

voltage, current

9. (5 pts) What is the difference between a conservative DISCIPLINE in Verilog-AMS versus a non-conservative DISCIPLINE?

   A conservative discipline has two different natures -- one bound to flow and one to potential; while a non-conservative nature only has one nature that is bound to either flow or potential.

10. (5 pts) Explain the semantics of the '<+' operator in Verilog-AMS.

    *It is the analog assignment operator that **sums** the current value of the RHS to the previous value of the LHS.*

11. (5 pts) Give at least two motivations for using System-C instead of Verilog or VHDL.

Code compiles to native machine code, so faster simulation execution.

You can synthesize directly from your high level model written in System-C to a netlist without having to first translate to different HDL like Verilog or VHDL.

12. (5 pts) What is the IBIS standard? Why was it needed?

IBIS is a standard for modeling input/output buffers. A portable standard was needed to model I/O cells for ASICS in such a way that allowed portability between different simulation environments, and also protected the proprietary design information of the IO cell.