COMS W4995-02
Languages for Embedded System Design
Homework 4

Prof. Stephen A. Edwards
Columbia University

Assigned November 18, 2003
Due December 4, 2003

You may submit the solutions either on paper or electronically, but not both. If you submit it electronically, send a single file that is text, PostScript, PDF, or Word. Don't send multiple files or an archive such as tar or zip. I just print out whatever you submit and read it; I don't try to run the programs.

Make sure your name appears at the beginning of the file you send.

I want the paper or electronic versions at the beginning of class (4:10 PM EDT) on the due date. This applies to both on-campus and CVN students.

1. **(5 points)** Book, Exercise 16-1: What are the three types of processes in SystemC and why were they included?

2. **(10 points)** Book, Exercise 16-2: For each type of process in SystemC, devise an example of each and show how each one could be implemented in Verilog. I'm looking for three short fragments of Verilog; you do not have to show me the SystemC process specification from which you derived it.

3. **(5 points)** Book, Exercise 16-4: Synchronous processes run in a nondeterministic order, yet SystemC is generally deterministic. What did the language designers do to ensure this?

4. **(10 points)** Book, Exercise 3-4:

   A continuous assignment can be implemented as an `always` block.

   (a) Write an always block for the continuous assignment

   ```
   assign #15 {carry_out, sum_out} = carry_in + ina + inb;
   ```

   (b) Can an always block always be written as a continuous assignment? Why?

5. **(10 points)** Book, Exercise 3-6: Write a short `always` block that behaves differently if all its blocking assignments are replaced with nonblocking assignments.

6. **(60 points)** Verilog programming.

   I've installed a Verilog simulator on the cunix cluster (cunix.columbia.edu) in my home directory `~se2007/verilog`. It's a demo copy that comes with Thomas & Moorby and is limited to programs of less than 1000 lines. This should not be a problem for this assignment. You might also want to check out Icarus Verilog (www.icarus.com), a free implementation.

   As an example, running the "verilog" executable on the following program (in `~se2007/verilog/testxor.v`)

```
module testxor;
  reg a, b;
  wire c;

  xor (c, a, b);

  initial begin
     $monitor($time,,,"a b c: %b%b%b", a, b, c);
     #10 a = 0;
     #10 b = 0;
     #10 a = 1;
     #10 b = 1;
     #10 a = 0;
  end
endmodule
```

produces, along with lots of other junk,

```
 0  a b c: xxx
10  a b c: 0xx
20  a b c: 000
30  a b c: 101
40  a b c: 110
50  a b c: 011
```

(a) Write a structural Verilog module for a circuit built from NAND gates that generates the output for the “a” (topmost) segment of a seven-segment display (see problem 2-5 on page 29 of the text). Include a testbench like the one shown above and show me both the Verilog source file and the output from the testbench.

(b) Write the smallest behavioral Verilog module you can that decodes every segment. Again, show me the source and the output from the testbench showing every number is decoded properly.

(c) Hook two copies of your decoder module to a behavioral module that counts synchronously (i.e., every “posedge clk”) from 00 to 99 in BCD and show me the source and output from your testbench for the numbers 00 to 12.