CS 247 - Project 1

For the following problems, hand in printouts of all the VHDL Source Code and Signal Waveforms. Annotate the signal waveform printout with hand written comments to document them. Also, be sure to answer those questions requiring a written explanation or diagram. Your written explanation should be based on and backed up by simulation results. Note that future projects may incrementally build upon previous projects. In fact, problems within a project may build upon each other. Therefore, be sure to adopt a file naming/archiving scheme whereby designs (.vhd files) can be reused for future problems.

1) Enter the entity declaration and behavioral architecture specification of *equiv_tester* exactly as it appears (except for the line numbers) on page 22 of the class notes. Compile the entity and its behavioral specification, link, and then invoke the PeakVHDL simulator on it. Monitor the value of signals a and b to verify that their values match those as shown in the table at the bottom of page 22. [2 pts]

2) Enter the entity declaration and behavioral architecture specification of *equiv* exactly as it appears (except for the line numbers) on page 20 of the class notes. Enter the entity declaration and structural architecture specification of the *testbench* exactly as shown on page 23 of the class notes. Put these two modules into the same .acc project used for *equiv_tester* in problem 1. Compile all the entities and their architecture specifications, link them, and then invoke the PeakVHDL simulator. Monitor the value of all signals (including TMP). Note that there is no time delay between output signal changes and input signal changes since the behavioral architecture body of *equiv* does not specify any time delays. [2 pts]

3) Add a delay time of 2ns and 3ns to lines 4 and 5 of *equiv* respectively. Recompile, relink, and run the simulator to observe the output as a function of time. How long does it take for output c to react to a change on either a or b? How long does it take for the signal tmp to react to a change on either a or b? Are these time delays as expected? [2 pts]

4) Reverse lines 4 and 5 of the behavioral model of *equiv* (the one with delay times) to verify that the order of statements is not important in event-driven simulation. What would the result of this statement order reversal be if VHDL behaved as a typical programming language ? [2 pts]

5) The following is a simple, commonly used model to generate a perpetual clock signal in VHDL. Write an entity declaration and behavioral model for a one pin device (call that pin CLK). The behavioral model will contain just one signal assignment statement of the form:

```
CLK <= not CLK after 20 ns;
```

Assuming that CLK was needed to drive the input of another device, what direction (in, out, or inout) of signal does it need to be and why ? [2 pts]

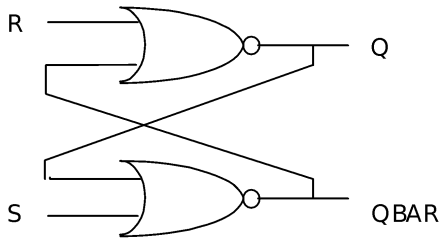
6) Use the same type of model as problem 5 to develop an entity called *counter* that has three clocks (clk1, clk2, clk3) to generate a truth table consisting of all eight exhaustive binary values for the three signals. That is, assume clk1 is the LSB and clk3 is the MSB of a truth table. Describe your technique in words. In particular, what would be a simple rule to employ for the delays between different bit positions assuming you want to count binary values at fixed time intervals, say every t ns ? [3 pts]

7) Write an entity declaration and behavioral architecture specification for a two input EXOR function. Assume that this EXOR function responds to changes in its inputs after 10 ns of inertial delay. Compile the EXOR entity and invoke the VHDL simulator on it. Monitor the value of all signals. Verify proper operation by setting the inputs to all four possible input combinations and observing the output signal's response as a function of simulation time. Hint: reuse as much of equiv_tester and testbench from problems 1 and 2. Note that this EXOR "function" only specifies "stimulus-response" behavior. This type of model is often used by engineers to obtain a high-level functional verification for the "building blocks" that they intend on using in a design. In problem 9, you will refine the design of the EXOR function to the gate level of detail by specifying a structural implementation for it. [2 pts]

8) Write an entity declaration and behavioral architecture specification for a two-input NOR gate. Assume that this NOR gate responds to changes in its inputs after 4ns of inertial delay. Verify its proper operation. Again, reuse as many entities as possible from previous problems. [2 pts]

9) Assume that the only parts you have for building various functions are NOR gates. Using only the NOR gates defined in problem 8 with 4ns delay as components, design a 2-input EXOR function. Draw a schematic of the circuit and develop a VHDL structural model of the circuit based upon your schematic. Label your schematic to show correspondence with your structural VHDL model. Apply the four input test vectors needed to completely test the 2-input part to verify proper performance. The responses obtained here should match those of the behavioral model from problem 7 (except for timing differences). Note that you can use the same entity declaration for EXOR as that used for problem 7; however, if you do so, be sure to specify that you want the simulator to use the structure model (instead of the behavior model) as the design unit. [6 pts]

10) Using the NOR gates with a 4ns delay as the fundamental component of an RS flip flop, write the entity declaration and structural architecture model which corresponds to the gate-level schematic below. Do not put any initial values on any signals within the model itself.



Write a simple tester and testbench which just initializes R to '1' at simulation time = 0. Do not do anything else to R or S yet. Compile the RS flip flop entity and start the simulator. Verify that the Q output goes to 0 and QBAR goes to 1 after the proper gate delays. [4 pts]

11) Modify the tester file for problem 10 so that R is not initialized, and neither is S. Without initializing R (or any other signals) to '1' at simulation time = 0, rerun the simulation for problem 10 above so that the simulator is in the default initial startup state (i.e. current sim time = 0 and, for PeakVHDL, all signals = '0'). Explain what happens. Could such a phenomena actually occur in real life using real physical components ? Explain why or why not. *[Hint: Think about how a flip-flop works. Consider what simulation is and how it models the world. Then consider what a physical flip-flop would do in the first few moments after power-up, which is what is being modeled by the simulator at sim time = (0 + some very small delta) with all signals initialized to '0'.]*
[12 pts, based primarily upon the written explanations you provide.]

12) Write an entity declaration and behavioral specification for a test generation circuit appropriate for use on the RS flip flop of problem 10 that will generate the following waveform. Verify that the RS flip flop can be set and reset by R and S. As with any flip-flop, note that at any given time, only either R or S is '1'. List all input and output signals of the UUT to verify proper operation. Explain the UUT's response when either the set or reset inputs are 1 ns pulses. [5 pts]

13) Repeat the simulation performed as part of problem 12 using TRANSPORT delay (instead of the default inertial delay) on the NOR gates of the RS flip flop. Explain how and why the UUT's response is different from the inertial delay model. [5 pts]

[Note: For problems 14 - 17, please turn in this sheet with your answers recorded on it.]

14) Perform the following 4-bit binary additions. The numbers are in 2's complement form. Which ones produce an overflow, and how should the answer be interpreted knowing that an overflow occurred. [4 pts]

0101	0101	1110	0011
0100	1100	1100	0100

15) Compute the decimal value which corresponds to the following binary numbers using the various representation schemes listed. Assume all of the numbers are positive (no sign bit is used). Recall that Modified Booth's is based on a radix four recoding. [6 pts]

Representation Scheme	Binary value	Decimal value
Floating Point:	1011.01	_____
A four bit, base 2, biased 8 exponent:	.100 E 1001	_____
Normalized notation and Phantom 1:	.100	_____
Booth's Notation:	+1 0 -1	_____
Modified Booth's Notation:	+1 -1	_____
Modified Booth's Notation:	+2 -1	_____

16) Give the base 10 value of the following 4-bit binary numbers using the three number representation schemes. Distinguish between +0 and -0. [2 pts]

Binary	Signed Magnitude	1's Complement	2's Complement
1111			
1000			
0000			
1011			
0111			

17) Assume the following two numbers are normalized. They are shown below with their mantissa values in binary and their exponent values in decimal. Perform a floating point addition showing the pre-addition denormalization step and the post-addition renormalization step. [2 pts]

X = .10011 E 2

Y = .11101 E 6

18) Determine an algorithm that could be used for detecting an overflow by comparing the sign of the sum with the signs of the augend and addend. Assume the numbers are in signed-2's complement representation. Explain in words how and why the algorithm works. Next, formulate a hardware implementation of the scheme by drawing a simple schematic. [4 pts]

19) Perform the multiplication 7 x 15 where you are given 7 as the multiplicand and 15 as the multiplier. Both original operands are 8 bit numbers. Use unsigned, ordinary binary notation and the ordinary multiplication algorithm similar to that shown in the class notes. [4 pts]

20) Perform the multiplication 7 x 15 where you are given 7 as the multiplicand and 15 as the multiplier. Both original operands are 8 bit numbers. Use Modified Booth's Algorithm to recode the multiplier. Show all partial products that are generated along with any sign extension which needs to be done in order to produce a product that will fit into a 16 bit register. Use 2's complement representation and math. [5 pts]

21) Perform 12 divided by 4 using the restoring division algorithm in binary. Show and explain all steps involved by using a running commentary in decimal similar to that used in the class notes. Compute a 4 bit answer. [5 pts]

22) Perform 12 divided by 4 using the non-restoring division algorithm in binary. Show and explain all steps involved by using a running commentary in decimal similar to that used in the class notes. Compute a 4 bit answer. [5 pts]

[Note: For problems 23, 24, 25, please turn in this sheet with your answers recorded on it.]

23) Compute the proper entries for the table below which shows a carry lookahead adder implementation to compute the sum of X and Y as given. Assume that the carry into the LSB is zero. [4 pts]

i	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	
X	0	1	0	0	1	0	1	1	Time 0
Y	0	1	1	0	0	1	1	1	
Pi									Time 1
Gi									
Ci									Time 2
Si									Time 3

24) Compute the proper entries for the table below which shows a carry select adder implementation to compute the sum of two eight bit numbers, X and Y as given. Assume that C0 = 0. [8 pts]

i	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	Bit Position
X	0	1	0	1	1	1	0	1	Addend
Y	1	0	0	1	0	1	0	0	Augend
S_0 C_0									Step 1
S_1 C_1									
S_0 C_0									Step 2
S_1 C_1									
S_0 C_0									Step 3
S_1 C_1									

25) Using the reference tables for Booth's and Modified Booth's representation, recode the two binary numbers below. [2 pts]

0 1 0 1 1 1 1 0 1 0 0 1

Booth's Recoding: _____

0 1 0 1 1 1 1 0 1 0 0 1

Modified Booth's: _____