

CSE 140L – Summer Session II – 2017

Instructor: Mohsen Imani

Homework #3

Checkoff Deadline: Wednesday, August 23

PROBLEM 1: GENERAL QUESTIONS

Answer the following questions in a concise way. You are encouraged to look for the answer online. These questions will not be graded for the homework and you are not required to provide a report for them or submit anything, but they might be part of the final exam.

- 1) What is the difference between **wire** and **reg**?
- 2) What is the difference between blocking and nonblocking assignments?
- 3) What is the difference between bit wise, unary and logical operators?
- 4) Write a Verilog code to swap two registers without using a temporary register.
- 5) What is the difference between **posedge** and **negedge** keywords?
- 6) What is the difference between **\$display** and **\$monitor**?
- 7) Given the following Verilog code, what value of **a** is displayed?

```
always @(clk) begin
    a = 0;
    a <= 1;
    $display(a);
end
```

- 8) What is the difference between the following two lines of Verilog code?

```
#5 a = b;
a = #5 b;
```

- 9) What is the difference between the following lines of code?

```
reg1 <= #10 reg2 ;
reg3 = #10 reg4 ;
```

- 10) What is the difference between **===** and **==** in Verilog?

PROBLEM 2: RIPPLE CARRY ADDER

Using full adders as building blocks, implement a 4-bit adder in the **ripple carry** form. Use explicit module connections. Do not use behavioral modeling for this problem. Write a testbench to test your adder with various input values.

PROBLEM 3: CARRY LOOKAHEAD ADDER

Using full adders as building blocks, implement a 4-bit adder in the **carry lookahead** form. Use explicit module connections. Do not use behavioral modeling for this problem. Write a testbench to test your adder with various input values.

PROBLEM 4: CARRY SELECT ADDER

Using the 4-bit ripple carry adder you implemented in Problem 2 as building blocks, implement an 8-bit adder in the **carry select** form. Use explicit module connections. Do not use behavioral modeling for this problem. Write a testbench to test your adder with various input values.

PROBLEM 5: ARITHMETIC LOGIC UNIT (ALU)

Implement an ALU with control signals S0, S1 that performs the following operations on 8-bit input operands A, B, and C and produces the 8-bit output D. You can use behavioral modeling for this problem. Write a testbench to test your module performing different operations on various input values.

Operation ID	S0	S1	OPERATION
0	0	0	$D = (A + B) + C$
1	0	1	$D = (A - B) - C$
2	1	0	$D = (A \text{ xnor } B) \text{ xnor } C$
3	1	1	If $A[0] == 1$ $D = B + C$ Else $D = B - C$

PROBLEM 6: FIRST SYNCHRONOUS DESIGN

Implement a 4-bit down counter (starts at 15 and counts down to 0). The inputs will include:

1. **enable** : the counter counts down unless either **enable** or the counter value is 0 .
2. **reset** : the counter value is reset to 15 if **reset** is 1.

The output will include:

1. **count** : current counter value in binary (4 bits)
2. **alarm** : goes high when the counter value reaches 0 and stays high until the counter is reset.

Test your counter fully by simulating the various scenarios it might encounter.

Checkoff List (To be completed with a tutor, a TA or the instructor by Wednesday 08/23).

NOTE: Before starting the checkoff, you MUST submit your zipped project via TED. The submitted material MUST contain all the source code. The checkoff can only be attempted once, so make sure that your solution is working fine.

For two of the modules you implemented in Problems 2 to 6, selected by TA/tutor, you will be asked to:

- 1) Explain the functionality
- 2) Review the Verilog code
- 3) Compile the code
- 4) Review and run testbench to verify functionality