**UNIVERSITY OF CALIFORNIA, BERKELEY**
**College of Engineering**
**Department of Electrical Engineering and Computer Sciences**

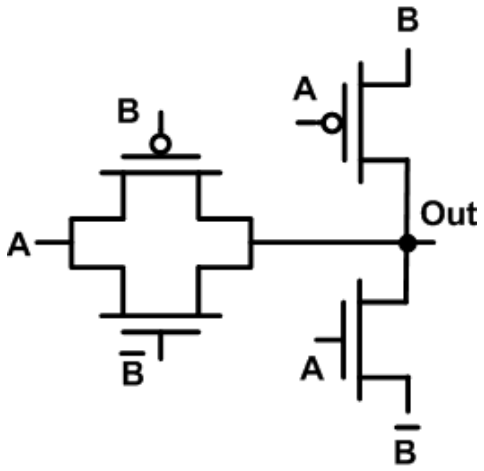
**Elad Alon** **Homework #3** **EECS 150**

*Due Thursday, September 22, 5pm, 240 Cory drop box.*

1. An "N:M compressor" creates an M-bit binary-weighted output that represents the N-bit unit-weighted input. For example, a full adder represents A+B+Cin as R+Cout, and thus implements a 3:2 compressor. For this problem, you will create a 5:3 compressor. The five equally-weighted input bits are represented by 3 binary-encoded outputs.
    a. Download the HW3 Verilog project (instructions at the end of this document).
    b. Implement a 5:3 compressor in "Compressor53.v". A skeleton is provided; do not change the interface.
    c. Write a self-checking testbench for the 5:3 compressor in "HW3Testbench.v". Complete the "Compressor53Test.input" test vector file containing all 32 test cases; see DDCA Section 4.8 – Testbenches for an example. Please edit the existing file so as not to break the simulation script.
    d. Simulate the circuit and show that it works.
    e. Introduce an error in the test vector file and show that the testbench reports a mismatch.

2. DDCA 1.64

3. Logic design:
    a. Derive the logic function for the gate shown below:




    b. What type of gate is this?
    c. Is this a "proper" CMOS gate in that an NMOS transistor passes low values and a PMOS transistor passes high values?

4. Starting from the CMOS edge-triggered flip-flop design shown in class, show three modified flip-flop designs that each implement one of the following features. For all three of these designs, you should try to minimize the total number of transistors.
    a. Synchronous reset.
    b. Asynchronous set.
    c. Clock enable. Note that you should implement the version described by:
    always @(posedge clk)
      if (clk_en) q<=d;

5. Consider the implementation of a 3-LUT using CMOS transistors. Assume the LUT has the following inputs and outputs: data inputs IN0, IN1, and IN2, data output OUT, a special clock signal CCLK used for shifting in configuration bits, C for accepting configuration bits, and COUT allowing chaining together the configuration circuits of multiple LUTs. The configuration is loaded by clocking the configuration chain for the appropriate number of clock cycles and feeding the proper bit pattern to the C input (just like in HW#2).

   Sketch a circuit design for this LUT. For the configuration bits you can just use flip-flops (i.e., you don't need to implement these at the transistor level), but the LUT itself should be drawn at the transistor level. You should attempt to minimize the total number of transistors for your design. Note that although you won't be drawing the flip-flops at the transistor level, you should keep in mind how many transistors are used inside of each flip-flop when counting the total number of transistors in your design.

6. In this problem we will look at how a piece of the programmable interconnect portion of an FPGA might be implemented. Programming the connections generally happens at places where 4 signals/wires (north, south, east, west) meet. Using tri-state buffers, inverters, and/or transmission gates, show how you would implement a circuit that allows any one of the signals to drive any one or more of the other signals. Every signal should be buffered both on its way and on its way out - i.e., you can't just short the wires together, instead each signal must go through at least one tri-state or inverter. Try to minimize the number of transistors required for your implementation.

ADDITIONAL INSTRUCTIONS FOR PROBLEM 1:

To copy the skeleton files, issue the command:

```
curl http://inst.eecs.berkeley.edu/~cs150/fa11/agenda/hw/hw3.tar.gz | tar -xzv
```

For this exercise, complete `Compressor53.v`, `HW3Testbench.v`, and `Compressor53Test.input`.

Verilog files can be found in hw3/src, and the test vector can be found in hw3/sim/tests/

It may save you time and reduce human errors to create the test vector using your favorite scripting language or MS Excel / Open Office.

You will test your design by using a verilog simulator as in HW2.

**To simulate your design:**

```
# export MGLS_LICENSE_FILE=1717@sunv40z-1.eecs.berkeley.edu:1717@sunv20z-
1.eecs.berkeley.edu
# cd hw3/sim
# make
```

This will run HW3Testbench.v and exhaustively test your compressor. The output of the make command will show the test results.

The simulations also generate waveforms of the signals in your design that you can view to aid in debugging. Issue the following commands:

```
# cd hw3/sim
# ./viewwave results/HW3Testbench.wlf &
```

This opens Modelsim, which shows waveforms for the signals in your device. On the left, you should see a tree of module names and signals. Right click on the one you would like to view, then select `add > to wave > all items in region`.

To submit this exercise, run "`submit hw3`" from your class account. (You may need to run `register` first). Remember to submit from a Solaris machine such as `cory.eecs.berkeley.edu`. Please submit only:

```
src/Compressor53.v
src/HW3Testbench.v
src/Compressor53Test.input
sim/results/HW3Testbench.transcript.pass
sim/results/HW3Testbench.transcript.fail
```

Note that if you would like to see how your design operates in real life, we have provided further skeleton files and information below to you to do so. (Running/testing your design on the FPGA is by no means required however.)

**To run your design on the FPGA:**

```
# cd hw3/
# make
# make impact
```