1. Design a combinational Verilog module

    module majority(A, Y);

that takes a 3-bit input $A$ and outputs a single bit $Y$ which is equal to 1 if and only the majority of the input bits are 1’s. You should also design a test bench to test your module on the binary equivalents of the decimal inputs $A = 0$ to $A = 7$. Submit printouts of your majority module code, the test bench code, and a waveform plot of a successful test run.

2. Use the axioms and theorems of Boolean algebra to prove several identities involving the NAND function. Please show that for any binary value $a$:

   a. $a \text{ NAND } 0 = (a0)' = 1$
   b. $a \text{ NAND } 1 = (a1)' = a'$
   c. $a \text{ NAND } a = (aa)' = a'$
   d. $a \text{ NAND } a' = (aa')' = 1$

3. We will show using Boolean algebra that the positive edge triggered D flip-flop from the handout only changes the output $Q$ to the value of the input $D$ when the clock changes from 0 to 1. We will start with the circuit in the state listed at $t = 0$ below.

   \[
   \begin{array}{cccccccc}
   C \text{lk} & D & X & Y & S & R & Q & Q' \\
   \hline
   t = 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
   t = 10 & 0 & [a] & 0 & 1 & 1 & 1 & 0 & 1 \\
   & 0 & a & 0 & [a'] & 1 & 1 & 0 & 1 \\
   \end{array}
   \]
   
   a. At $t = 10$, change $D$ to the value $a$, and show that $Q$ remains the same. Propagate the changes throughout the circuit like we did on the board in class until all variables converge, and then indicate convergence with a checkmark and a separating horizontal line. You will need to use the NAND identities from the previous problem. The first step has been done for you.
   b. At $t = 20$, change $C\text{lk}$ to 1, and show that $Q$ takes on the value $a$ soon after.
   c. At $t = 30$, change $D$ to $a'$ and show that $Q$ remains unaffected.
4. We can construct state machines like the ones in Worksheet 2 by writing down a truth table for the state transition function. The inputs to the state transition function are the state variables $S_i$, and the outputs are the “next state” variables $N_i$. From there, we can use K-maps to produce simplified Boolean expressions for each output, and then implement them using gates.

Produce diagrams for these state machines:

a. Using two D flip-flops and any additional gates necessary, construct a state machine that progresses from the state 00 to 01, 10, and finally to 11. Once in the state 11, your state machine should stay there indefinitely.

b. Using three D flip-flops and any additional gates necessary, construct a state machine that progresses through all of the three-bit Gray codes, starting with 000 and continuing through 100 before starting over.