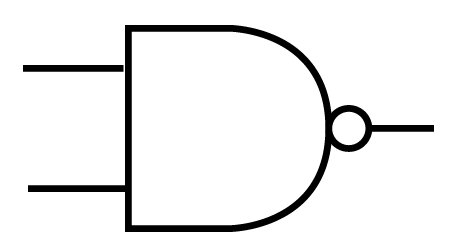
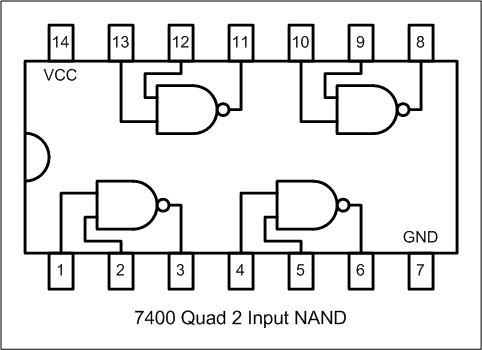
**Max Score = 15 points**

CS 250 2018 Spring Homework 02

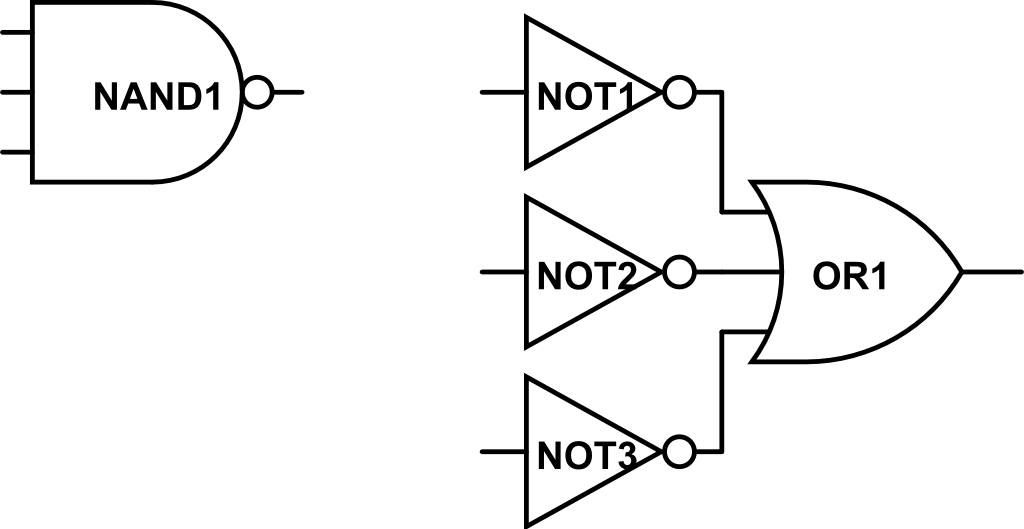
This assignment is due at 11:59:00 pm Thursday, January 25, 2018.

Upload your typewritten answer document in either PDF or Word format to Blackboard.

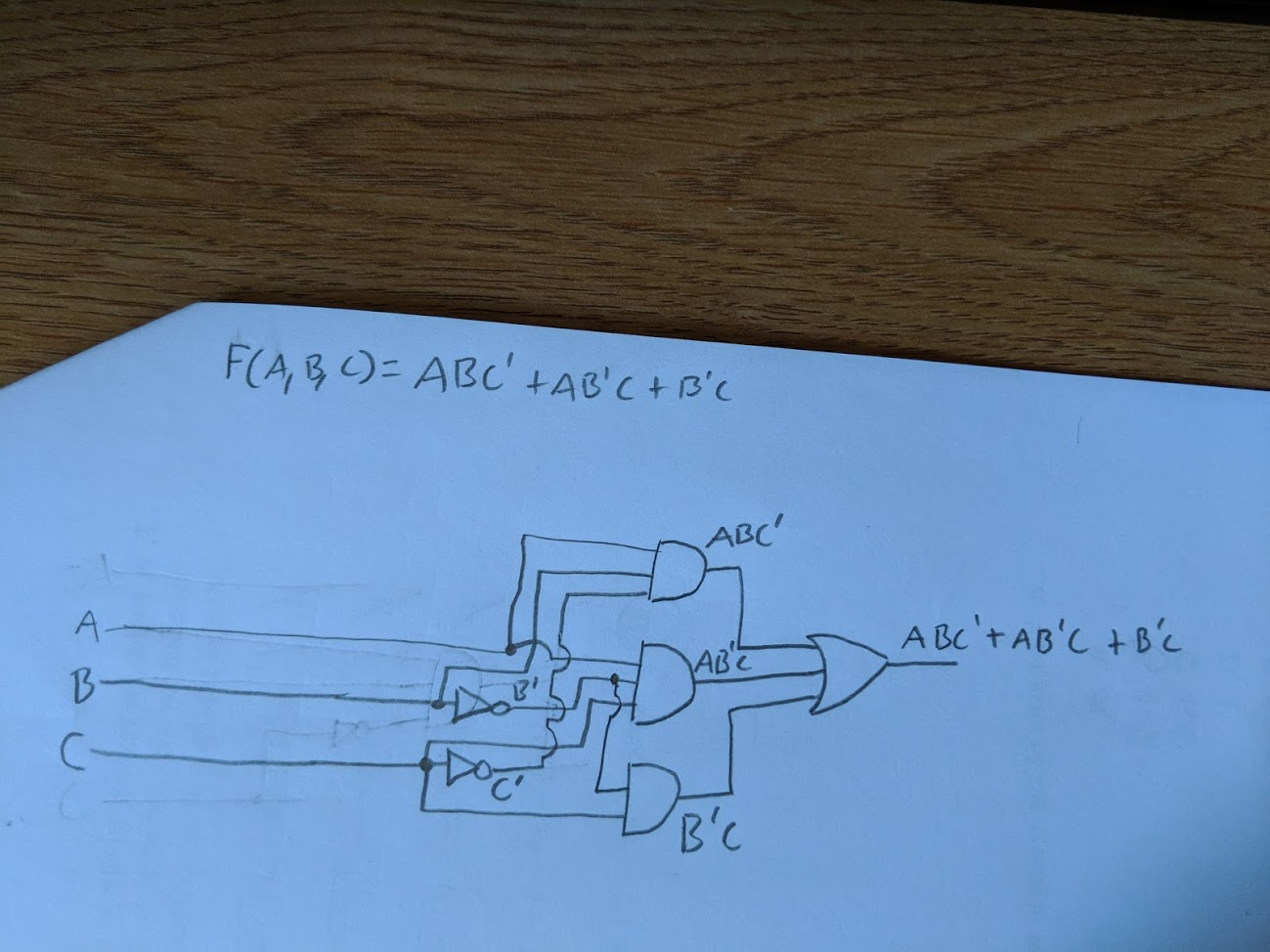
Download from Blackboard to be sure that your upload was successful. Your last upload that is not marked “LATE” by Blackboard is the upload that will be graded. There is no “grace” period for late uploads

1. Consider the computation of the logic function NAND. We can represent this computation in diagrams using the NAND gate symbol shown here.  
    
   1. Figure 2.5 in the textbook shows a representation of the NAND computation at a lower level of abstraction than the gate symbol. What is shown in that textbook figure for NAND that is omitted at the higher level of abstraction used when depicting the NAND computation using the gate symbol shown here? List the lower level symbols omitted when using the gate symbol. **4 CMOS transistors** **are omitted when using the gate symbol.**
   2. This symbol is the pinout diagram representation of the SN74HTC00N quadruple 2-input NAND chip in your lab kit, or 7400 for short.  
         
      Is the level of abstraction of the 7400 pinout diagram higher, lower, or the same with respect to a set of four NAND gate symbols? **The level of abstraction is the same.**
   3. Is the level of abstraction of the logic signal input pins for one gate of the 7400 pinout diagram higher, lower, or the same with respect to the logic signal inputs for Figure 2.5? **The level of abstraction is higher.**
   4. Is the level of abstraction of the VCC and GND power and ground pins for the 7400 pinout diagram higher, lower, or the same as the + voltage (Vdd) and 0 volts connections for the NAND circuit in Figure 2.5? **The level of abstraction is higher.**
2. Fill in the missing computed result columns of the following truth table.

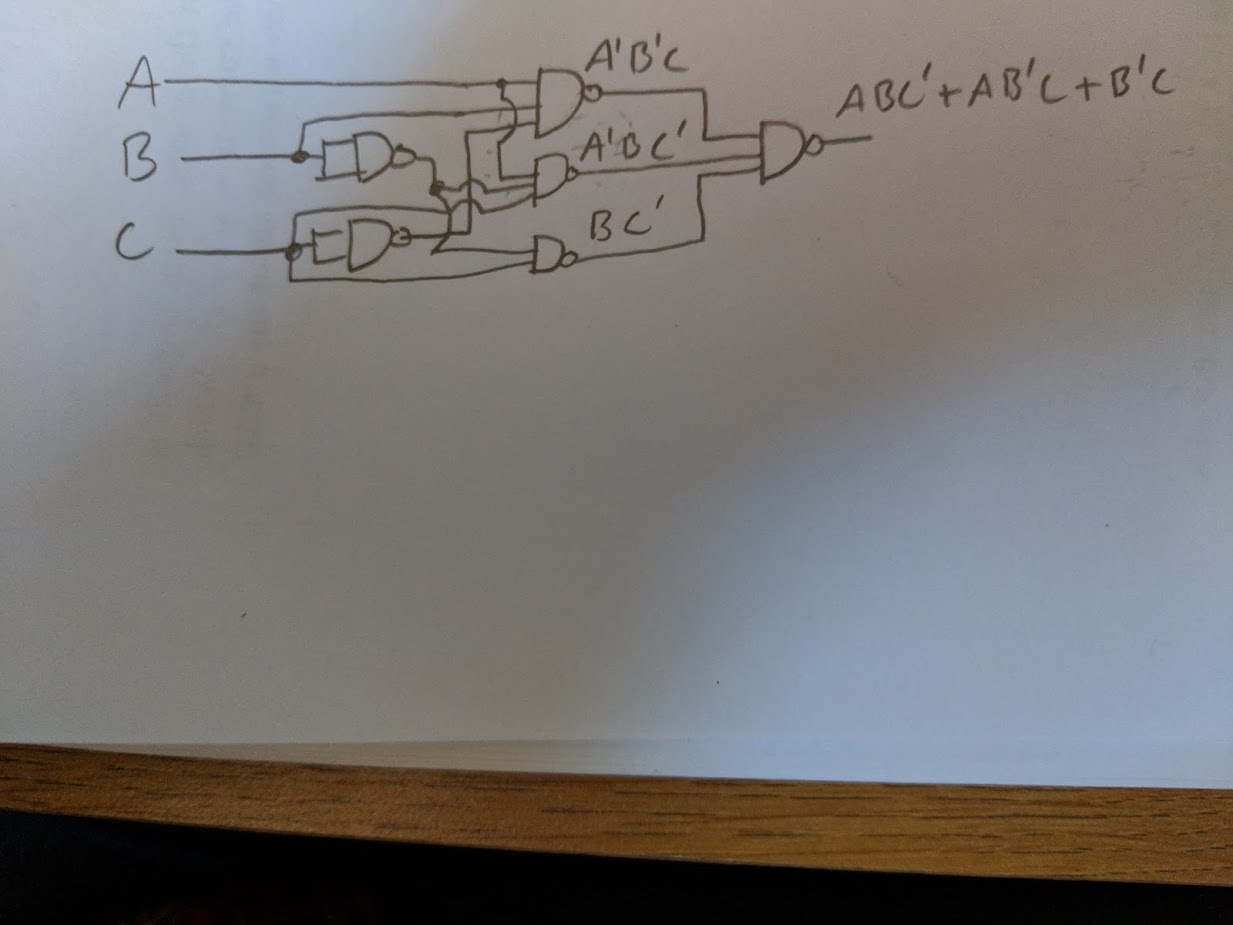
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Inputs | | | Computed results | | | | | |
| A | B | C | ABC | (ABC)’ | A’ | B’ | C’ | (A’+B’+C’) |
| 0 | 0 | 0 | **0** | **1** | **1** | **1** | **1** | **1** |
| 0 | 0 | 1 | **0** | **1** | **1** | **1** | **0** | **1** |
| 0 | 1 | 0 | **0** | **1** | **1** | **0** | **1** | **1** |
| 0 | 1 | 1 | **0** | **1** | **1** | **0** | **0** | **1** |
| 1 | 0 | 0 | **0** | **1** | **0** | **1** | **1** | **1** |
| 1 | 0 | 1 | **0** | **1** | **0** | **1** | **0** | **1** |
| 1 | 1 | 0 | **0** | **1** | **0** | **0** | **1** | **1** |
| 1 | 1 | 1 | **1** | **0** | **0** | **0** | **0** | **0** |

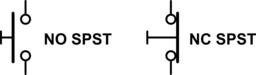
What does the content of the truth table prove about the relationship between these two logic gate circuits? How is this relationship proven? **The truth table proves that they both create the same output. The relationship is proven by giving each of them the same input and getting the same output.**  
  


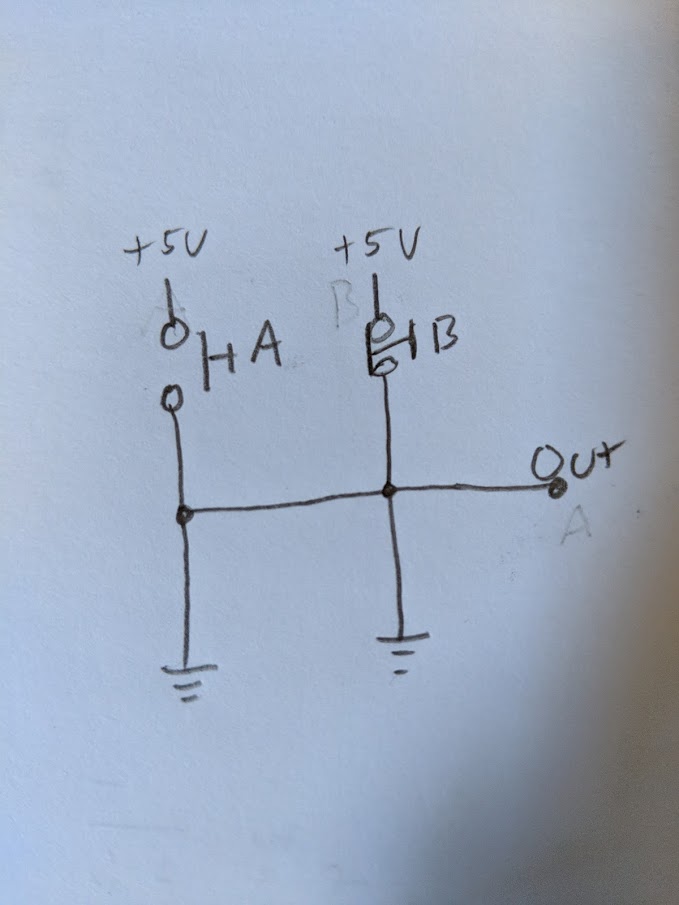
1. Consider the Boolean expression F(A,B,C) = ABC’ + AB’C + B’C.
   1. Draw a gate-level circuit diagram for the logic circuit that computes F(A,B,C) and containing only gates that implement one of the three Boolean algebra operations. Use gates with more than two inputs when this will make the circuit simpler. Abstract out NOT gates for inputs by labeling an input X’ as needed.



* 1. Draw a gate-level diagram containing only NAND gates for a logic circuit computing F(A,B,C). Use gates with more than two inputs when this will make the circuit simpler.



1. By fixing one input of a two input NAND gate to always receive a logic 1 the gate then functions to invert the logic value sent to the free input. Is there a 2-input function that can invert a free input when the other input is held to logic 1 and also not invert the free input when the other input is held to logic 0? If this function has a name, what is it? Otherwise answer “no name.” **The function is a NOR gate.**
2. What error is present in Figure 2.6 of the textbook? **The AND, OR, and XOR gates all only have 1 input instead of two.**
3. The normally-open (NO) single-pole, single-throw (SPST) push button switch (same type as in the lab kit) transitions from high resistance to low resistance when pushed. A normally-closed (NC) SPST push button switch reverses this behavior. The schematic symbols for these two switches are shown here.   
   

Using only switches of these two types, wire, and connections to +5 V and ground, draw a schematic to implement (AB)’. Clearly label your inputs and output.  


1. Consider this truth table for the partially specified function F(A,B,C).

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | F(A,B,C) |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | X |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | X |
| 1 | 1 | 1 | 1 |

* 1. Show the K-map for F(A,B,C).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A\BC | 00 | 01 | 11 | 10 |
| 0 | 0 | 1 | x | 1 |
| 1 | 0 | 1 | 1 | x |

* 1. Think about how you want to choose the values of the two Don’t Care computed results, then show an image of the K-map with circles around the optimal groupings of K-map minterms to most simplify an SOP implementation of F(A,B,C). Write out this simplest SOP expression for F(A,B,C).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A\BC | 00 | 01 | 11 | 10 |
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |

**SOP = B+C**

1. On a new image of the K-map, and re-considering the choice of values for the Don’t Cares, circle the optimal groupings of maxterms to most simplify a POS implementation of F(A,B,C). Write out this simplest POS expression for F(A,B,C).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A\BC | 00 | 01 | 11 | 10 |
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |

**POS = B’C’**