CALIFORNIA STATE UNIVERSITY
LOS ANGELES

Department of Electrical and Computer Engineering

EE-2449 Digital Logic Lab

EXPERIMENT 3
LOGIC GATES

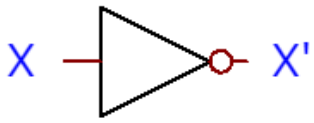
Text: Mano and Ciletti, *Digital Design, 5th Edition,* Chapter 2
Required chips:

7400: quad 2-input NAND	**7402**: quad 2-input NOR	**7404**: hex inverters (NOT)
7408: quad 2-input AND	**7432**: quad 2-input OR	**7486**: quad 2-input XOR
7493: 4-bit ripple counter	**7410:** triple 3-input NAND	

3.1 There are three basic logic functions from which all circuits can be designed: NOT (invert), OR, and AND. In addition, another useful logic function is exclusive-OR or XOR.

The NOT function is also known as INVERT (so a NOT gate is also referred to as an INVERTER, as was explained back in Exp.1.) The NOT function is true (1) if the input is false (0) and vice versa. The logical expression for the inverse of logic variable X is X' (i.e. NOT X).

The logic gate symbol of a NOT/INVERTER is shown at the right. The triangle without the bubble at the output is called a "buffer gate" It does not change the gate's logic level. Instead, for the same logic level, it may output to a load much more current than its input source is capable of.



It is actually the bubble at the end of the gate that indicates inversion. The bubble on the output of any gate means that the output level generated inside the gate is complemented before it appears at the actual gate output.

The AND function is true (1) only if all inputs are true (1's). The NAND function is an AND-NOT function and is the inverse of an AND function. So a NAND function is false (0) if all inputs are true (1's). Conversely, it is true (1) if one or more inputs are false (0). The logical expression for the AND of two variables, X and Y, is X•Y (or XY). The logical expression for the NAND of X and Y is (X•Y)' (or (XY)'. The symbol for a NAND is the same as for an AND except for a bubble at the end (as shown on next page). So if the inputs to a NAND are all 1's, the internal AND function produces a 1 which is then inverted to a 0 by the bubble.

The OR function is true (1) if one or more inputs are true (1). The NOR function is an OR-NOT function and is the inverse of an OR function. So a NOR function is false (0) if one or more inputs are true (1). Conversely, it is true (1) only if all inputs are false (0's). The logic equation for the OR of two logic variables, X and Y, is X+Y. The logic equation for the NOR of X and Y is (X+Y)'.

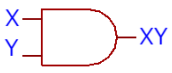
The 2-variable (2-input) XOR function is true (1) if either input is true (1) but not both. The XOR is known as an odd function. If an odd number of its inputs are true (1) the output is true (1) – this definition works for any number of inputs. An XNOR is an XOR-NOT function. For a 2-variable (2-input) XNOR function, the function is true (1) if both inputs are the same. The XNOR is an even function. If an even number of inputs are true (1) the output is true (1) – this definition works for any number of inputs.

Complete the truth table below for the AND, NAND, OR, NOR, XOR, and XNOR functions.

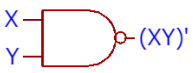
X	**Y**	**X•Y**	**(X•Y)'**	**X+Y**	**(X+Y)'**	**X ⊕ Y**	**(X ⊕ Y)'**
0	0						
0	1						
1	0						
1	1						

The logic gates for these functions are shown below:

AND

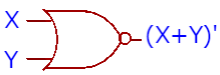


NAND



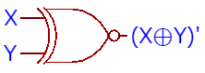
OR

NOR



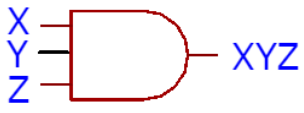
XOR

XNOR



Logic functions can have multiple inputs. Fill in the truth table for the three-input NAND whose gate is shown on the right.

X	**Y**	**Z**	**(X•Y•Z)'**
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

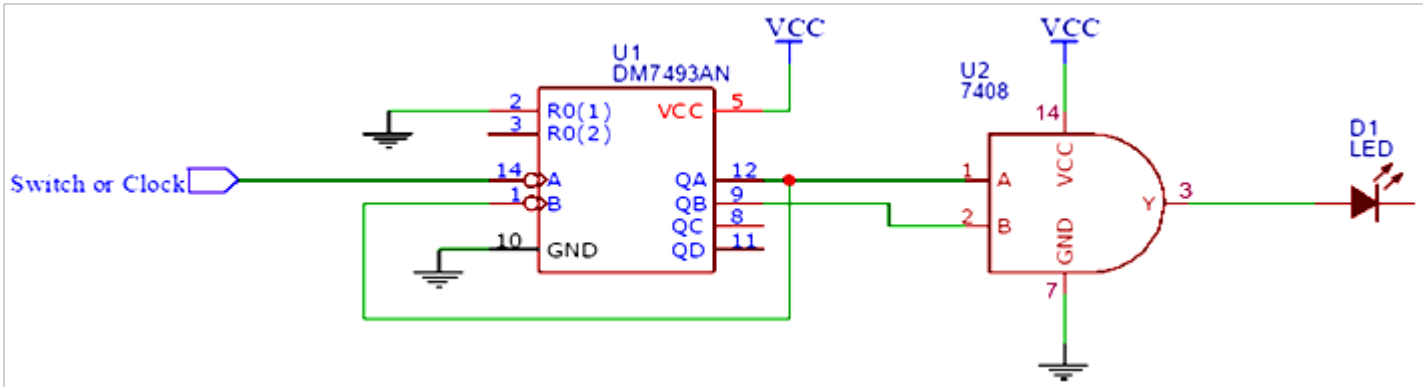


Notice in the above discussion that the terms true and false are used interchangeably with 1 and 0. The reason is that binary variables like X and Y are really statements about things; they can be true or false. We use 1 or 0 when we deal with them mathematically (as in truth tables or expressions like X'Y).

Suppose you define variable D as meaning Dry weather, H as meaning Hot weather, and W as meaning Water some plants. Let W = D•H. In other words, water the plants if both D and H are simultaneously true. In a circuit application where the water is to go on automatically when it's dry and hot, you would need sensors to measure humidity and temperature. If the humidity drops below a certain level, we can say that D is true (sensor D goes from 0 to 1 meaning its voltage goes from low to high). Similarly, if temperature rises above a certain level, we can say that H is true (sensor H goes from 0 to 1; its voltage goes from low to high).

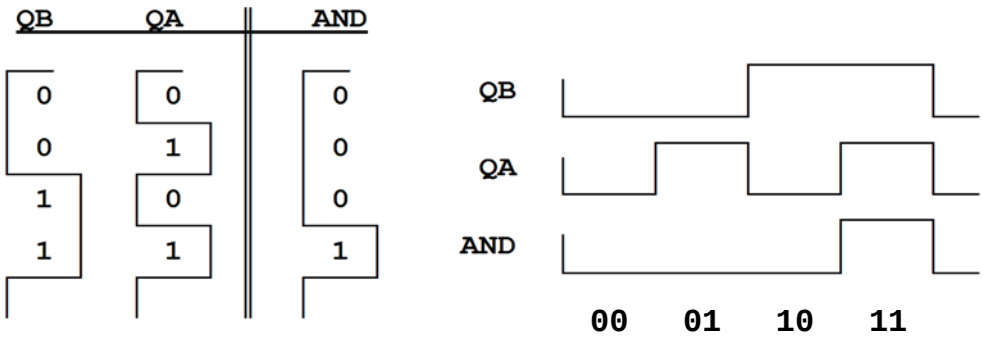
When both are true, then W = D•H = 1•1 = 1 (i.e. W is true--the water turns on). Sensor output voltages D and H are connected to an AND gate, and when the AND gate output voltage, W, goes high (1), it closes a switch which allows water to flow.

3.2* Set up the 7493 as the binary counter of Experiment 2. (Ground at least one reset input or the chip won't count.) Since counter outputs go through all possible combinations in binary order, they can be used to test the behavior of the various gates listed above.



Connect QB and QA to two inputs of a 7408 AND gate. (Don't forget 5V and ground connections; pins 14 and 7. If you forget, the chip is inactive). Referring to the truth tables in 3.1, X = QB and Y = QA. Connect the clock A input of the counter (pin 14) to a pulser output. Connect QB, QA, and the output of the AND gate to LEDs and verify that the correct outputs are produced by referring to the truth table below (or your answer to part 3.1).

The truth table and the corresponding waveforms for an AND gate are shown below:



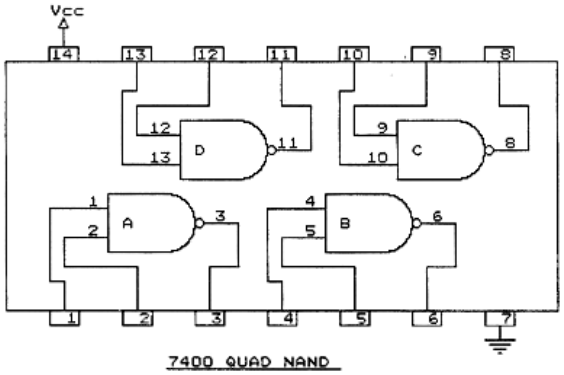
3.3* Notice that the waveforms are just a graphical form of the truth table (rotated 90 degrees counter clockwise). By displaying such waveforms on a scope, you can obtain a description of circuit behavior equivalent to a truth table.

Actually, you don't need both QB and QA on the scope to know where a complete count cycle starts. It always begins where QB, the msb (most significant bit), falls to 0. Thus to see how the AND gate output fits into a complete cycle, it only needs to be displayed with QB. (Suppose QA was not shown above. On the scope you can see that the AND waveform is correctly positioned within a count cycle because it is synchronized with the falling edge of QB.)

Connect the clock A input (pin 14) to the timer output. Switch in the 0.47 μF so the clock frequency will be high (1 KHz). Do this by flipping the capacitor select switch toward the other capacitor (470μF) as you did in a previous experiment. Bring both QB and the AND output to the scope. Adjust the scope's Time/Div until QB goes low for 2 divisions and high for 2 divisions. That way, 4 divisions will represent the full 4-count cycle, QBQA = 00,01,10,11. Thus, QB is 0 (low) for the first two counts and 1 (high) for the next two, exactly as in the above diagram. Then, as the diagram shows, the AND-gate output should remain low for the first 3 counts and go high for the 4th only. (Reminder: instructions on using the scope are found at the end of Experiment 2.)

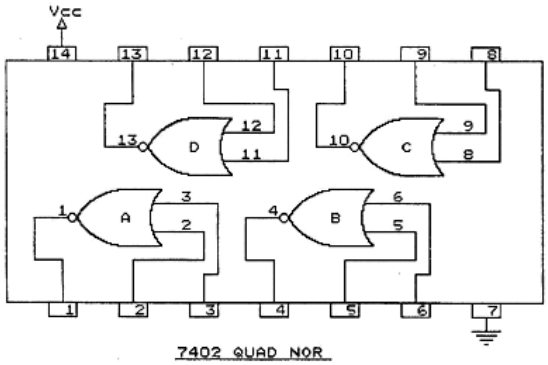
Since a new cycle of 4 counts begins each time QB falls, select negative edge trigger for the channel that displays QB. (Make sure the trigger level lies within QB's waveform.) Draw the waveforms for QB and the AND output in your lab manual. Repeat for a NAND, OR, XOR, and NOR (do the NOR last since it requires some wiring changes). *Leave the scope settings alone as you change gates* so QB will look the same in all 5 images.

Diagrams for gate chips are shown below. Note that all have the same power and ground pin numbers (14 and 7). Also, except for the 7402 (NORs), they have exactly the same input and output pin numbers. So, except for the 7402, no wiring changes will be needed as you replace one gate chip with the next. (*Remember to remove chips **gently** so as not to bend pins.)*



7400 QUAD NAND

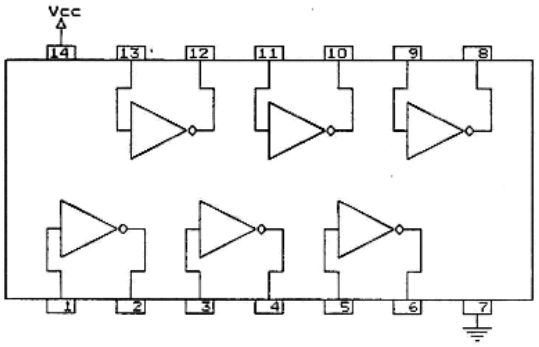
(SAME PINOUT FOR 7408 AND, 7432 OR, 7486 EX-OR)



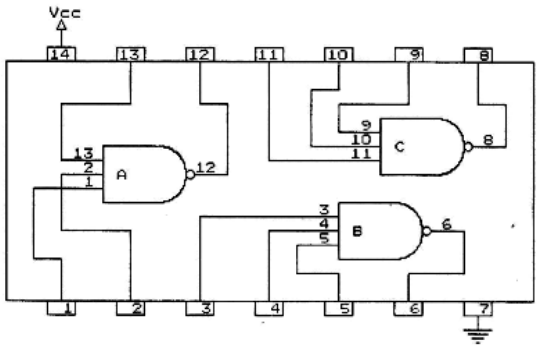
7402 QUAD NOR

For each logic function, draw the wiring diagram of the logic gate (include the pin numbers) in your lab manual. You do not have to redraw the 7493 counter for each wiring diagram. Just label the inputs of the gates QA and QB. For each logic function draw the waveforms of QB and the output of the respective gate and verify that the output matches your truth table values from section 3.1. You can instead save the screen image on a flash drive (as described in the 'scope appendix of Exp 2) or take a photograph. Either way, paste the results into your lab journal.

3.4* Repeat the above for a NOT gate and a 3-input NAND gate. For each chip, connect the Q outputs from the counter to drive the logic gates: QA for the NOT gate and QC,QB,QA for the 3-input NAND. The diagrams for the 7404 chip with six ("hex") inverters and the 7410 chip with 3-input NAND gates are shown below (and at the end of the manual). Don't forget to connect power and ground.



7404 or 7414 HEX INVERTER



7410 TRIPLE 3-INPUT NAND

For each gate, draw the wiring diagram (including pin numbers) in your lab manual. As before, you do not have to redraw the 7493 counter for each wiring diagram. Just label the inputs of the gates with the appropriate Q output from the counter; i.e. QA for the inverter and QC, QB,QA for the NAND. For each gate, draw the waveforms of the msb (most significant input bit) and the output of the respective gate and verify that the output matches your truth table values from section 3.1.

Note: since the NAND has 3 inputs, the counter goes through 8 counts (000 through 111 and back). So, the msb, QC, will be low for 4 counts and high for 4 counts before it repeats.