CALIFORNIA STATE UNIVERSITY
LOS ANGELES

Department of Electrical and Computer Engineering

EE-2449 Digital Logic Lab

# EXPERIMENT 6
# COMBINATIONAL LOGIC CIRCUITS

Text: Mano and Ciletti, *Digital Design, 5th Edition,* Chapters 2 and 3

Required chips:

**7400**: quad 2-input NAND **7402**: quad 2-input NOR **7404**: hex inverters
**7408**: quad 2-input AND **7432**: quad 2-input OR **7493**: 4-bit ripple counter
**7410**: three 3-input NAND

Lab report: Note – you will write up experiments 6 and 7 in one laboratory report.

---

**6.1** In this section, you will design (but ***NOT*** build) a circuit with 4 inputs, W,X,Y,Z. The circuit output is F. The Karnaugh map for F is as shown. The numbers at the top of the squares represent the corresponding rows of the truth table for F. Alternatively, you can think of them as the states of a 4-bit counter, WXYZ, whose outputs are input to the circuit that generates F.

| W \ YZ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 3 **1** | 2 **1** |
| 01 | 4 | 5 | 7 **1** | 6 |
| 11 | 12 | 13 | 15 **1** | 14 |
| 10 | 8 **1** | 9 **1** | 11 **1** | 10 **1** |

(Y, X, W, Z)

a) Derive the equation for F from the map. *Make sure it is in its simplest form.* (No reason to draw a truth-table--all the information about the circuit is in the map.)

b) Draw the circuit for F. In your lab notebook you can draw by hand but for your lab report use ExpressSCH and show U-numbers (i.e. U1, U2, etc..) Remember to include "ports" for W,X,Y,Z, and output F. Use only *one each* of the following chips: 7408, 7432, and 7404. If you need more, your equation for F is not in its simplest form.

c) Suppose your circuit were actually built and tested with inputs WXYZ from a counter. Let the following table represent oscilloscope waveforms for W and F. The numbers at the top are the 16 states of the counter. The waveform has an error; an incorrect connection was made to one of the AND-gate inputs. Draw the map for F from the waveform, derive F from the map, and compare with the one you found in a), above. What is the incorrect connection? Show your work and explain your answer.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **W** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **F** | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

**6.2*** This section is like 6.1 except only 3 inputs will be used: X,Y,Z (from counter outputs QC, QB, QA). You will be assigned a set of 5 numbered squares in an 8-square K-map. Insert a 1 in each square of your set (the other squares will contain 0's). Then, derive an equation for F(X,Y,Z) based on your map, as discussed below. As in Sec. 6.1, you will need only one each of the following chips: 7408, 7432, and 7404. As a Sum-of-Products (SOP), your equation should have the form

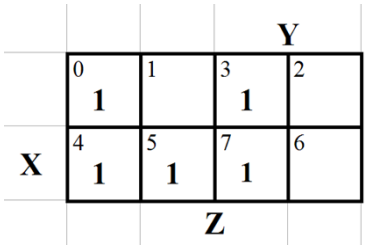
xx + xx + xx.

where each "x" represents an input variable, X, Y, or Z, or its complement.

Here is the list of eight sets of numbered squares from which your instructor will choose your set:

**01347 12356 21567 30457 40237 50126 61257 70346**

The map at the right shows how to fill in the squares for the set 30457. From your map, derive an equation for F in the form xx + xx + xx. (Note: you could write F with four xx terms. In this example you could include both 4-5 and 5-7, but one of the them would be redundant. So use only three terms, but make sure you have included all 1's in the map.)




A) Fill in a map for your set and derive an equation for F in two ways:

1. In SOP form, F = xx + xx + xx, from the 1's of the map.
2. In POS (Product-of-Sums) form, F = (x+x+x)(x+x), derive as follows from your map:
   a) Find the equation for F′ in SOP form from the **0's** of the map. Don't derive F′ by complementing your SOP equation for F, since F′ would then be in POS form, which you don't want.
   b) Complement F′ algebraically to get F as a product-of-sums, POS.

B) Based on your equations for F, use ExpressSCH to draw two circuits for F: an AND/OR for the SOP form, and an OR/AND for the POS form.
You will need some NOT gates as well for one or both forms of the circuit.

C) Include ***both*** of these circuit diagrams in your lab notebook, but **build only the POS one**.

D) Test your circuit with counter outputs X, Y, and Z (i.e. QC, QB, and QA). Bring F and X (msb) to the scope (not Y or Z) and capture their waveforms. (Important: trigger-menu settings: since a new cycle of 8 counts begins each time X falls, *trigger off the channel that displays X and select negative edge*. Adjust the scope's Time/Div setting so that one full period of X spans 8 divisions on the screen, one per count.)

E) Create a map for F from the waveforms and derive an SOP equation for F from the map. Compare it with the SOP equation above from your original map (just as you did in 6.1). If they agree, comment on that fact in your lab notebook. If they do not, debug your circuit to locate the error.

You may be able to do this simply by looking for a difference in the two equations. Suppose your design equation was XY + Y'Z' + YZ. while the one derived from the waveform is X'Y + Y'Z' + YZ. The error is simply that X' was mistakenly connected to an input of the XY gate instead of X.

**6.3** All-NAND/NOR Circuits: The goal of a good design is often to minimize the cost and physical size of a circuit. Reducing "chip count" does both, since fewer chips cost less, take up less PC board space, and require less handling (e.g. soldering). Sometimes, chip count can be reduced by designing with all NAND and NOR chips, instead of ANDs, ORs, and NOTs (inverters).

Please refer to the circuit diagram on page 6 of this experiment. In the top panel of the diagram, the function F = B'D'+ A'BC is realized in the usual way using NOTs, ANDs, and ORs. There is a total of 7 gates from 3 chips. The bottom panel shows the same function realized with all NANDs and NORs. Here, the total is only 5 gates from 2 chips.

**Additional uses of NAND/NOR circuits:**
In this class we are using gates built using Transistor-Transistor Logic (TTL) to design our digital circuits. Another logic family commonly used in today's microprocessors, microcontrollers, memory, and other digital circuits is Complementary Metal Oxide Semiconductors (CMOS). CMOS NAND and NOR gates require fewer transistors (switches) than AND and OR gates. Fewer transistors mean smaller integrated circuits, lower power requirements, and lower cost – all desirable features in today's digital systems. This is another important reason for learning how to implement circuits using only NAND and NOR gates.

Now, it can be rather complicated to derive the NAND/NOR equivalent algebraically. You would have to operate on F = B'D'+ A'BC in several steps:

(1) F = (B+D)'+ [A + (BC)']' OR{NOR (B,D) + NOR[A, NAND(B,C)]}

(2) F' = { (B+D)' + [A + (BC)']'}' NOR{NOR(B,D) + NOR[A, NAND(B,C)}

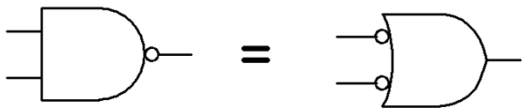
(3) F = [ {(B+D)'+ [A + (BC)']'}'] ' Complement F' with final NOR or NAND (as inverter).

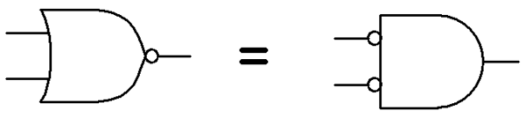
The circuit in the bottom panel on pg. 6 is the implementation of equation 3.

It is easier (*much* easier) to skip the algebra and work directly on the original circuit. By attaching inversion circles or "bubbles" to the inputs and outputs of AND and OR gates, you can turn them into NANDs and NORs. This graphical method uses alternative (DeMorgan) forms of NANDs and NORs wherever they make it easier to follow circuit logic. Recall DeMorgan's Theorem states:

(i) $(X \cdot Y)' = X' + Y'$

(ii) $(X+Y)' = X' \cdot Y'$





If we think in terms of logic gates, then case (i) means that a NAND gate can be an AND-INVERT (an AND gate with an inverted (bubbled) *output*) or an INVERT-OR (an OR gate with inverted (bubbled) *inputs*).

Likewise, case (ii) means that a NOR gate can be an OR-INVERT (an OR gate with an inverted output) or an INVERT-AND (an AND gate with inverted inputs).

Starting from the top diagram on page 6, first reduce the top two inverters to inversion bubbles and attach them to the inputs of the AND. This converts the AND into an invert-AND.

Although this is really a NOR (i.e. a 7402), leave it as an invert-AND so it's easy to see that the output is B'D' (bottom diagram). Next, reduce the inverter at input A to a bubble and attach it to the input of the following AND. This requires that a bubble be attached to the other AND input as well. Compensate for this by bubbling the output of the AND to its left.

The result is that the left-hand AND becomes an AND-invert (NAND), while the other becomes an invert-AND (a NOR). Finish up by bubbling the output of the OR, which becomes a NOR with output F'. Complement this back to F with a final NAND used as an inverter. This NAND is drawn as an invert-OR so its input bubbles will cancel the bubble of the previous NOR. Note that by showing some gates in their DeMorgan form, you make the logic as easy to follow as in the original NOT-AND-OR circuit. All you have to do is ignore the inversion bubbles when they appear at both ends of a connector. (Note: if NANDs and NORs were all shown in their usual non-DeMorgan forms, it would be hard to understand the underlying simple NOT-AND-OR structure since the effect of the bubbles wouldn't cancel out.)

See pages 7 and 8 for more examples that illustrate this.

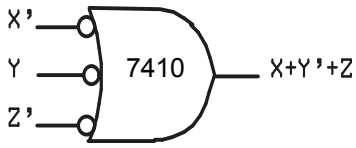
---

**6.4*** Here you will not build a circuit--only draw its diagram (using ExpressSCH). In section 6.2, you chose a function F(X,Y,Z) and designed the circuit using ANDs, ORs, and inverters in SOP and POS form. For this section, convert the SOP (AND/OR) circuit diagram (the one you didn't build) into one containing ***only*** NAND and NOR gates. (Don't use NOT gates; a NOR or NAND will serve as a NOT if you tie all its inputs together.)

Use DM (DeMorgan) equivalent gate symbols where they are needed, but ***only*** then. DM symbols have their inverter bubbles at the front, so use them if their inputs come from gates with bubbled outputs (NORMAL symbols) or from external inputs that are inverted (e.g. X'). Otherwise, do not use them. The point is that connecting wires should either have a bubble at each end so the bubbles cancel, ○———○ , or no bubbles at all (see diagrams on pg. 6).

Since NANDs are just bubbled ANDs and NORs are just bubbled ORs, your NAND/NOR circuit should resemble the AND/OR design of Exp. 6.2, And that's the point; if the connecting wires are bubbled at each end, you can ignore the bubbles and see circuit logic as easily as in 6.2.

Also, in converting to a NAND/NOR circuit, you might be able to replace two OR gates with a 3-input invert-OR. Example: to produce the OR expression X + Y' + Z, use the 3-input NAND (7410) drawn as an invert OR (notice the inputs have been inverted)..

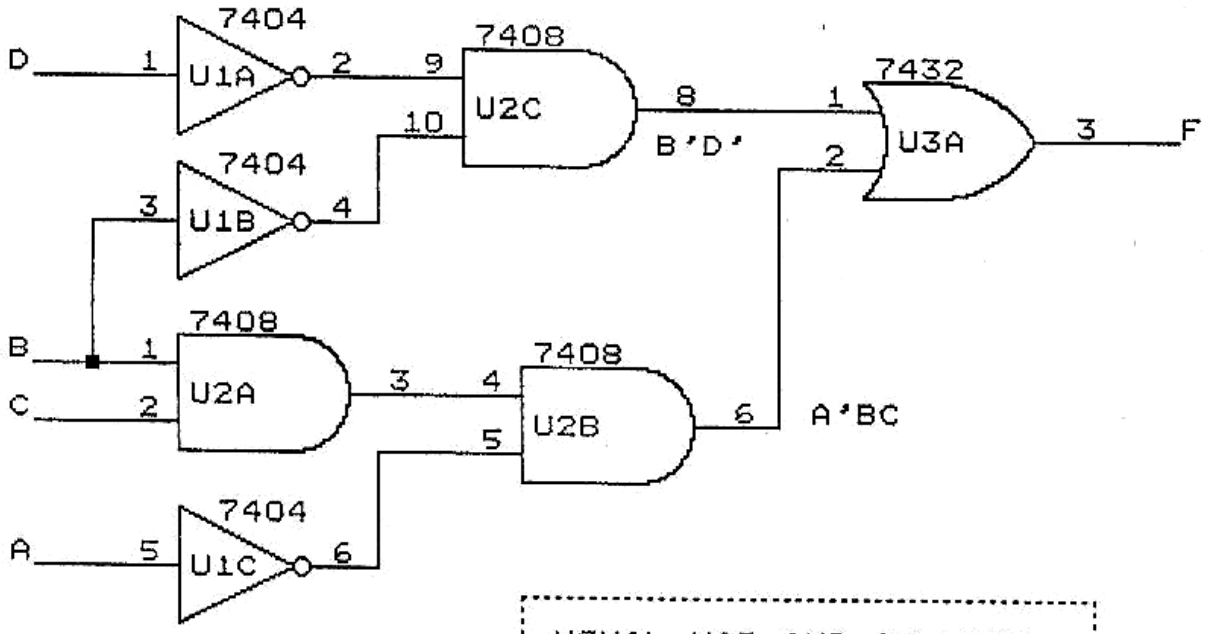



In your lab notebook, include the SOP equation for F and its corresponding circuit diagram from Exp 6.2 as well as the circuit diagram created here. Remember to use DeMorgan gate symbols but only where they are needed. If you use only normal symbols throughout, it will not be possible to follow circuit logic simply by looking at the diagram.

Also, answer the following question:
How many chips (not gates) were required in the circuit of Exp 6.2 compared to the number required here? Did using NANDs and NORs reduce the number of chips?

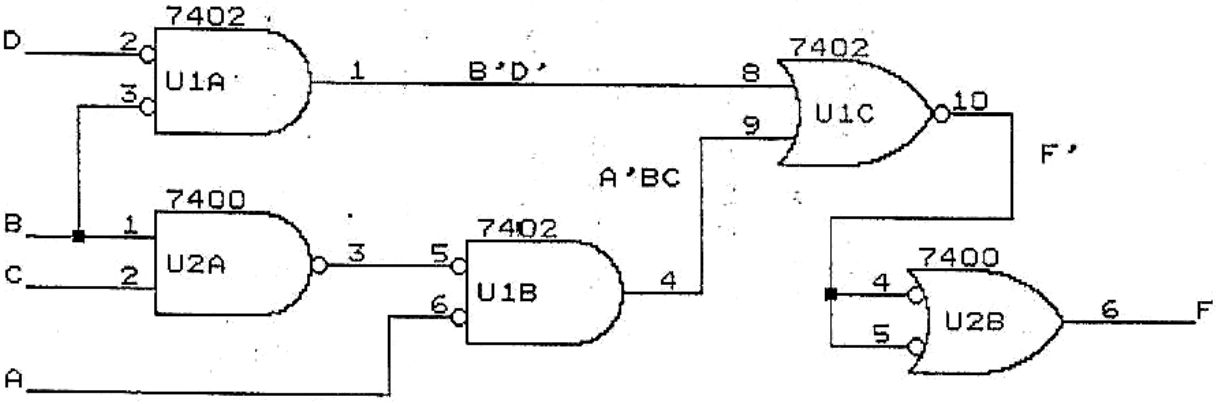
---

GIVEN FUNCTION: F = B'D' + A'BC



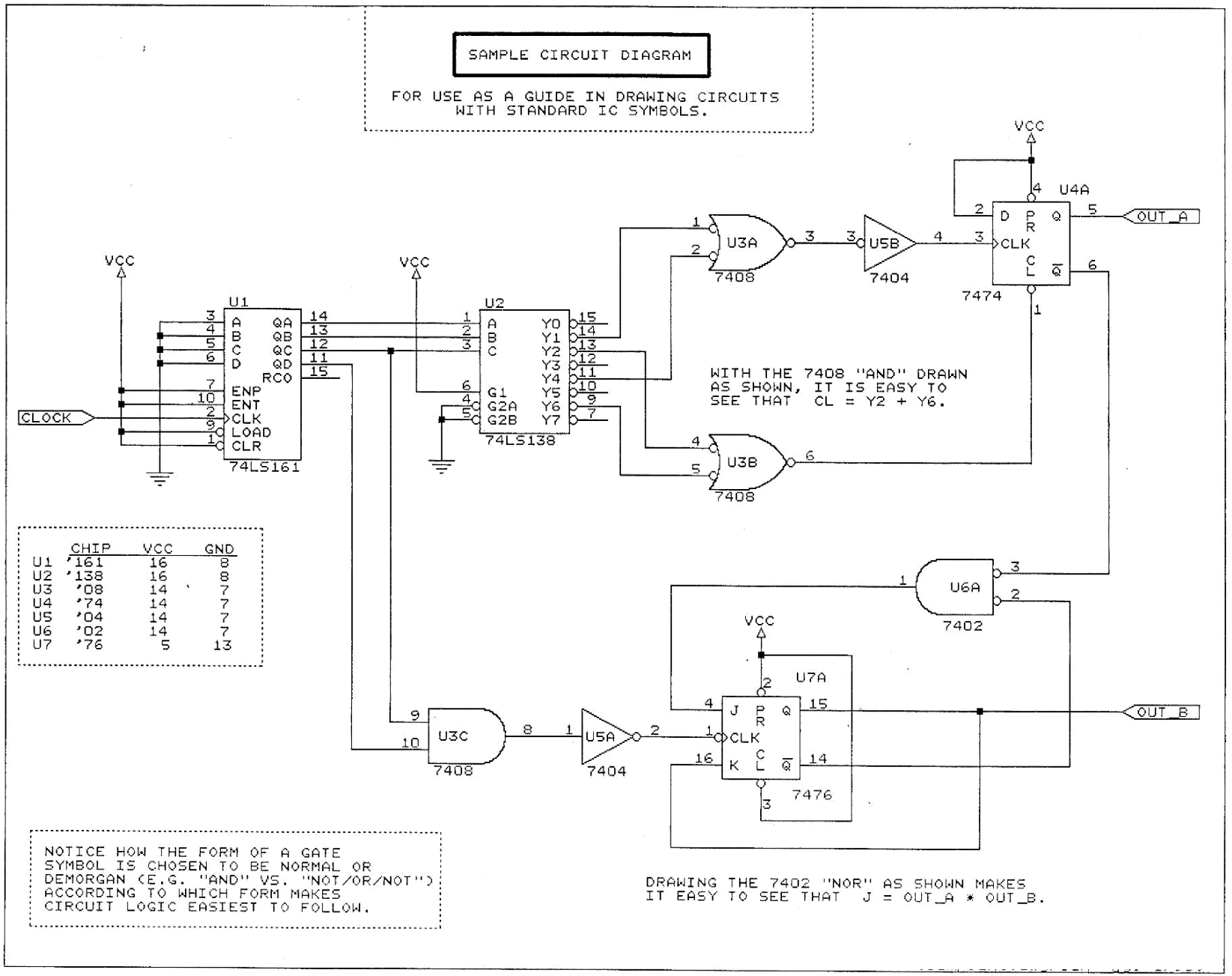

USUAL NOT–AND–OR FORM.
3 NOTS, 3 ANDS, 1 OR.
3 CHIPS TOTAL.

SAME FUNCTION USING NANDS AND NORS




CHOOSE "NOT/OR" FORM OF NAND
AND "NOT/AND" FORM OF NOR
WHEN DOING SO CANCELS OUT BUBBLES.

2 NANDS, 3 NORS.
2 CHIPS TOTAL.


SAMPLE CIRCUIT DIAGRAM
FOR USE AS A GUIDE IN DRAWING CIRCUITS WITH STANDARD IC SYMBOLS.
VCC
U1
74LS161
A QA B QB C QC D QD RCO ENP ENT CLK LOAD CLR
CLOCK
U2
74LS138
A B C G1 G2A G2B Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
U3A
7408
U5B
7404
U4A
7474
D PR Q CLK CL Q
OUT_A
WITH THE 7408 "AND" DRAWN AS SHOWN, IT IS EASY TO SEE THAT CL = Y2 + Y6.
U3B
7408
CHIP VCC GND
U1 '161 16 8
U2 '138 16 8
U3 '08 14 7
U4 '74 14 7
U5 '04 14 7
U6 '02 14 7
U7 '76 5 13
U6A
7402
U7A
7476
J PR Q CLK K CL Q
OUT_B
U3C
7408
U5A
7404
NOTICE HOW THE FORM OF A GATE SYMBOL IS CHOSEN TO BE NORMAL OR DEMORGAN (E.G. "AND" VS. "NOT/OR/NOT") ACCORDING TO WHICH FORM MAKES CIRCUIT LOGIC EASIEST TO FOLLOW.
DRAWING THE 7402 "NOR" AS SHOWN MAKES IT EASY TO SEE THAT J = OUT_A * OUT_B.