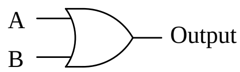
Sum-of-Products and Product-of-Sums expressions

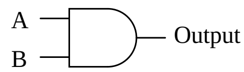
Resources and methods for learning about these subjects (list a few here, in preparation for your research):

Question 1

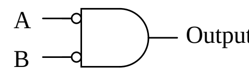
Identify each of these logic gates by name, and complete their respective truth tables:



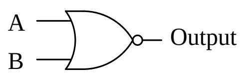
A	B	Output
0	0	
0	1	
1	0	
1	1	



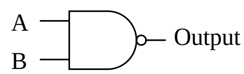
A	B	Output
0	0	
0	1	
1	0	
1	1	



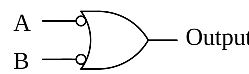
A	B	Output
0	0	
0	1	
1	0	
1	1	



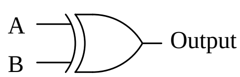
A	B	Output
0	0	
0	1	
1	0	
1	1	



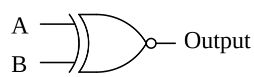
A	B	Output
0	0	
0	1	
1	0	
1	1	



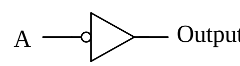
A	B	Output
0	0	
0	1	
1	0	
1	1	



A	B	Output
0	0	
0	1	
1	0	
1	1	



A	B	Output
0	0	
0	1	
1	0	
1	1	

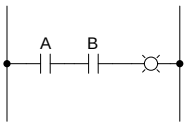


A	Output
0	
1	

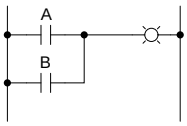
file 01249

Question 2

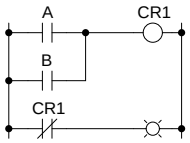
Identify each of these relay logic functions by name (AND, OR, NOR, etc.) and complete their respective truth tables:



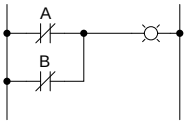
A	B	Output
0	0	
0	1	
1	0	
1	1	



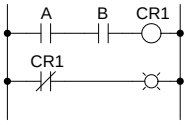
A	B	Output
0	0	
0	1	
1	0	
1	1	



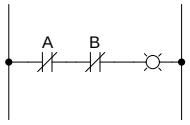
A	B	Output
0	0	
0	1	
1	0	
1	1	



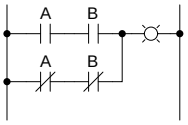
A	B	Output
0	0	
0	1	
1	0	
1	1	



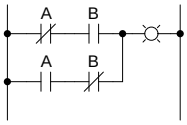
A	B	Output
0	0	
0	1	
1	0	
1	1	



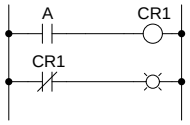
A	B	Output
0	0	
0	1	
1	0	
1	1	



A	B	Output
0	0	
0	1	
1	0	
1	1	



A	B	Output
0	0	
0	1	
1	0	
1	1	



A	Output
0	
1	

file 01335

Question 3

Inspect each of these Boolean expressions, and determine whether each one is a *sum of products*, or a *product of sums*:

$$(B + \overline{C} + D)(\overline{A} + B)$$

$$A\overline{B}\,\overline{C} + \overline{A}BC$$

$$(X + \overline{Y} + \overline{Z})(\overline{Y} + Z)(\overline{X} + Y)$$

$$\overline{M}\,\overline{N}\,\overline{O} + MN\overline{O} + M\overline{N}O$$

$$(X + \overline{Y + Z})(\overline{Y + \overline{Z}})$$

$$\overline{ABC} + A\overline{B}C$$

file 01324

Question 4

Sum-of-Product Boolean expressions all follow the same general form. As such, their equivalent logic gate circuits likewise follow a common form. Translate each of these SOP expressions into its equivalent logic gate circuit:

$$AB + A\overline{B}$$

$$A\overline{B} + \overline{A}B$$

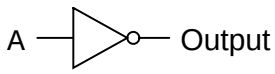
$$ABC + \overline{A}B\overline{C} + AB\overline{C}$$

file 01325

Question 5

Although it is seldom done, it is possible to express a truth table in verbal form, by describing what conditions must be met in order to generate a "high" output.

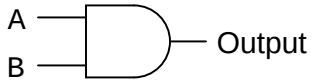
Take for example this simple truth table, for an inverter circuit:



A	Output
0	1
1	0

For this truth table, we could say that the output goes high when A is low. A different way of saying this would be to state that "the output is *true* when $\overline{A}$ is *true*."

Let's look at another example, this time of an AND gate:

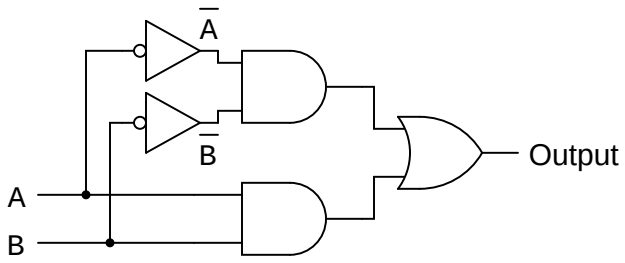


A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

For this truth table, we could say that the output goes high when A and B are both high. A different way of saying this would be to state that "the output is true when A is true and B is true." To use a half-Boolean, half-verbal description:

A AND B

Examine this logic gate circuit and corresponding truth table:



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Express the functionality of this truth table in words. What Boolean conditions must be satisfied ("true") in order for the output to assume a high state?

file 01326

Question 6

Develop a verbal description of this truth table, specifying what conditions must be met ("*true*" in a Boolean sense) in order for the output to assume a high state:

A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

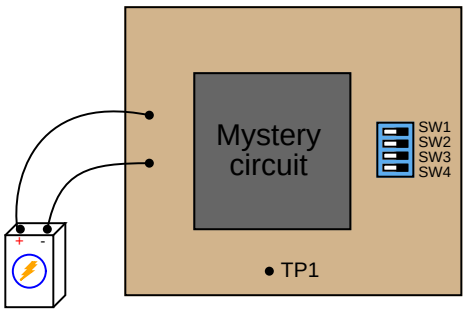
Do the same for this truth table as well:

A	B	C	Output
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

file 01327

Question 7

Suppose you were faced with the task of writing a Boolean expression for a logic circuit, the internals of which are unknown to you. The circuit has four inputs – each one set by the position of its own micro-switch – and one output. By experimenting with all the possible input switch combinations, and using a logic probe to "read" the output state (at test point TP1), you were able to write the following truth table describing the circuit's behavior:



SW1	SW2	SW3	SW4	TP1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Based on this truth table "description" of the circuit, write an appropriate Boolean expression for this circuit.

file 01304

Question 8

Write a Boolean SOP expression for this truth table, then simplify that expression as much as possible, and draw a logic gate circuit equivalent to that simplified expression:

A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

file 02822

Question 9

Write an SOP expression for this truth table, and then draw a gate circuit diagram corresponding to that SOP expression:

A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

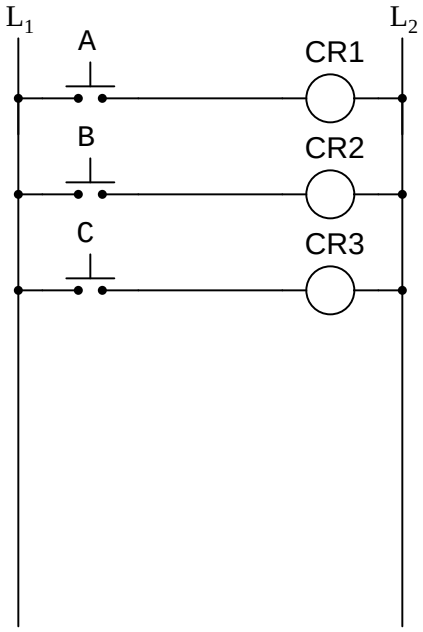
Finally, simplify this expression using Boolean algebra, and draw a simplified gate circuit based on this new (reduced) Boolean expression.

file 01333

Question 10

Write an SOP expression for this truth table, and then draw a ladder logic (relay) circuit diagram corresponding to that SOP expression:

A	B	C	Output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Implement the SOP logic function using contacts of relays CR1, CR2, and CR3. A partial ladder logic diagram has been provided for you.

Finally, simplify this expression using Boolean algebra, and draw a simplified ladder logic diagram based on this new (reduced) Boolean expression. When deciding "how far" to reduce the Boolean expression, choose a form that results in the minimum number of relay contacts in the simplified ladder logic diagram.

file 01334

Question 11

Design the simplest relay circuit possible (i.e. having the fewest contacts) to implement the following truth table:

A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

file 02827

Question 12

Product-of-Sum Boolean expressions all follow the same general form. As such, their equivalent logic gate circuits likewise follow a common form. Translate each of these POS expressions into its equivalent logic gate circuit:

$$(A + B)(\overline{A} + \overline{B})$$

$$(\overline{A} + \overline{B})(\overline{A} + B)$$

$$(A + B + C)(\overline{A} + B + \overline{C})(A + B + \overline{C})$$

file 02825

Question 13

Product-of-Sum Boolean expressions all follow the same general form. As such, their equivalent logic gate circuits likewise follow a common form. Translate each of these POS expressions into its equivalent logic gate circuit:

$$(A + B)(A + \overline{B})$$

$$(A + \overline{B})(\overline{A} + B)$$

$$(A + B + C)(\overline{A} + B + \overline{C})(A + B + \overline{C})$$

file 01336

Question 14

In an SOP expression, the minimum requirement for the expression's total value to be equal to 1 is that at least one of the product terms must be equal to 1. For instance, in the following SOP expression, we know that the value will be equal to 1 if $ABC = 1$ or if $A\overline{B}\,\overline{C} = 1$ or if $AB\overline{C} = 1$:

$$ABC + A\overline{B}\,\overline{C} + AB\overline{C}$$

What is the minimum requirement for a POS expression to be equal to 0? Take the following POS expression, for instance:

$$(A + B + C)(A + \overline{B} + C)(\overline{A} + B + C)$$

At the very least, what has to occur in order for this expression to equal 0?

file 01337

Question 15

Examine the following truth table:

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

We know that this table represents the function of a NAND gate. But suppose we wished to generate a Boolean expression for this gate as though we didn't know what it already was, and we chose to generate an SOP expression based on all the "high" output conditions in the truth table:

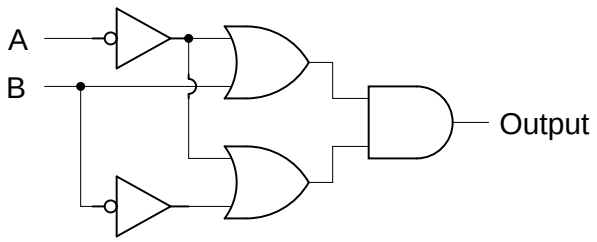
$$\overline{A}\,\overline{B} + \overline{A}B + A\overline{B}$$

Seems like a lot of work for just one gate, doesn't it? The fact that this truth table's output is mostly 1's causes us to have to write a relatively lengthy SOP expression. Wouldn't it be easier if we had a technique to generate a Boolean expression from the single *zero* output condition in this table? If we had such a technique, our resulting Boolean expression would have a lot fewer terms in it!

We know that a Negative-OR gate has the exact same functionality as a NAND gate. We also know that a Negative-OR gate's Boolean representation is $\overline{A} + \overline{B}$. If there is such a thing as a technique for deriving Boolean expressions from the "0" outputs of a truth table, this instance ought to fit it!

Now, examine the following truth table and logic gate circuit:

A	B	Output
0	0	1
0	1	1
1	0	0
1	1	0



Derive a Boolean expression from the gate circuit shown here, and then compare that expression with the truth table shown for this circuit. Do you see a pattern that would suggest a rule for deriving a Boolean expression directly from the truth table in this example (and the previous example)?

Hint: the rule involves *Product-of-Sums* form.

file 01338

Question 16

Examine this truth table and then write both SOP and POS Boolean expressions describing the Output:

A	B	C	Output
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Which of those Boolean expressions is simpler for this particular truth table? Which will be easier to reduce to simplest form (for the purpose of creating a gate circuit to implement it)?

file 02823

Question 17

Write a POS expression for this truth table, and then draw a ladder logic circuit corresponding to that expression:

A	B	C	Output
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

file 01340

Question 18

Write a Boolean expression for this truth table, then simplify that expression as much as possible, and draw a logic gate circuit equivalent to that simplified expression:

A	B	C	Output
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

file 01341

Question 19

Write a Boolean expression for this truth table, then simplify that expression as much as possible, and draw a logic gate circuit equivalent to that simplified expression:

A	B	C	Output
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

file 02826

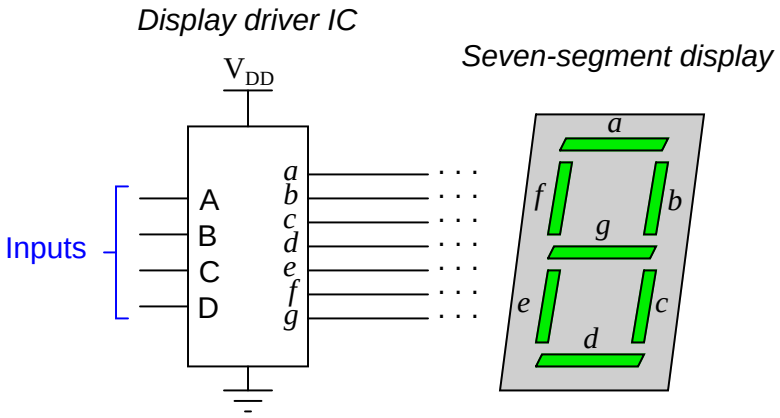
Question 20

Write two Boolean expressions for the Exclusive-OR function, one written in SOP form and the other written in POS form. Show through Boolean algebra reduction that the two expressions are indeed equivalent to one another. Then, draw the simplest ladder logic circuit possible to implement this function.

file 01346

Question 21

A *seven segment decoder* is a digital circuit designed to drive a very common type of digital display device: a set of LED (or LCD) segments that render numerals 0 through 9 at the command of a four-bit code:



The behavior of the display driver IC may be represented by a truth table with seven outputs: one for each segment of the seven-segment display (a through g). In the following table, a "1" output represents an active display segment, while a "0" output represents an inactive segment:

D	C	B	A	a	b	c	d	e	f	g	Display
0	0	0	0	1	1	1	1	1	1	0	"0"
0	0	0	1	0	1	1	0	0	0	0	"1"
0	0	1	0	1	1	0	1	1	0	1	"2"
0	0	1	1	1	1	1	1	0	0	1	"3"
0	1	0	0	0	1	1	0	0	1	1	"4"
0	1	0	1	1	0	1	1	0	1	1	"5"
0	1	1	0	1	0	1	1	1	1	1	"6"
0	1	1	1	1	1	1	0	0	0	0	"7"
1	0	0	0	1	1	1	1	1	1	1	"8"
1	0	0	1	1	1	1	1	0	1	1	"9"

Write the unsimplified SOP or POS expressions (choose the most appropriate form) for outputs a, b, c, and e.

file 02824

Question 22

Don't just sit there! Build something!!

Learning to analyze relay circuits requires much study and practice. Typically, students practice by working through lots of sample problems and checking their answers against those provided by the textbook or the instructor. While this is good, there is a much better way.

You will learn much more by actually *building and analyzing real circuits*, letting your test equipment provide the "answers" instead of a book or another person. For successful circuit-building exercises, follow these steps:

1. Draw the schematic diagram for the relay circuit to be analyzed.
2. Carefully build this circuit on a breadboard or other convenient medium.
3. Check the accuracy of the circuit's construction, following each wire to each connection point, and verifying these elements one-by-one on the diagram.
4. Analyze the circuit, determining all logic states for given input conditions.
5. Carefully measure those logic states, to verify the accuracy of your analysis.
6. If there are any errors, carefully check your circuit's construction against the diagram, then carefully re-analyze the circuit and re-measure.

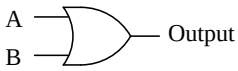
Always be sure that the power supply voltage levels are within specification for the relay coils you plan to use. I recommend using PC-board relays with coil voltages suitable for single-battery power (6 volt is good). Relay coils draw quite a bit more current than, say, semiconductor logic gates, so use a "lantern" size 6 volt battery for adequate operating life.

One way you can save time and reduce the possibility of error is to begin with a very simple circuit and incrementally add components to increase its complexity after each analysis, rather than building a whole new circuit for each practice problem. Another time-saving technique is to re-use the same components in a variety of different circuit configurations. This way, you won't have to measure any component's value more than once.

file 01205

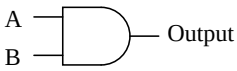
Answer 1

OR



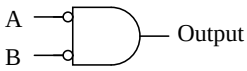
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

AND



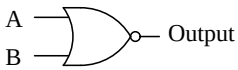
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Neg-AND



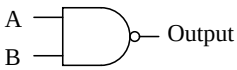
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

NOR



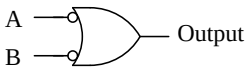
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

NAND



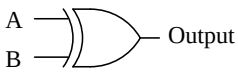
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Neg-OR



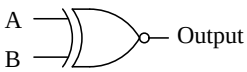
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

XOR



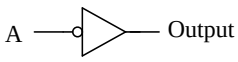
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

XNOR



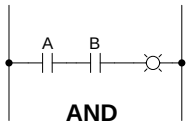
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

NOT



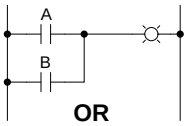
A	Output
0	1
1	0

Answer 2



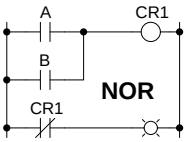
AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



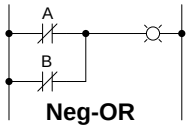
OR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



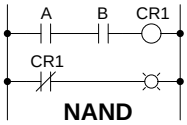
NOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



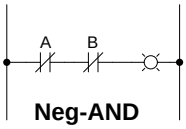
Neg-OR

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



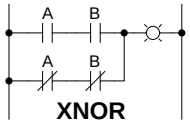
NAND

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



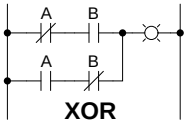
Neg-AND

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



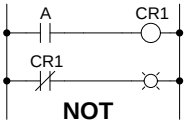
XNOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1



XOR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



NOT

A	Output
0	1
1	0

Answer 3

$$(B + \overline{C} + D)(\overline{A} + B) \qquad \textbf{POS}$$

$$A\overline{B}\,\overline{C} + \overline{A}BC \qquad \textbf{SOP}$$

$$(X + \overline{Y} + \overline{Z})(\overline{Y} + Z)(\overline{X} + Y) \qquad \textbf{POS}$$

$$\overline{M}\,\overline{N}\,\overline{O} + MN\overline{O} + M\overline{N}O \qquad \textbf{SOP}$$

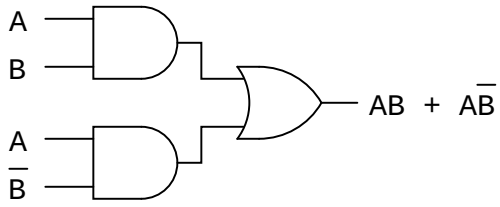
The last two expressions are "trick" questions: while technically being the product of summed variables, and the sum of multiplied (product) variables, respectively, do not follow "standard" POS and SOP forms, because they both have long complementation bars:

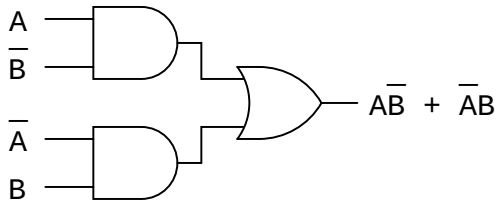
$$(X + \overline{Y + Z})(\overline{Y + \overline{Z}})$$

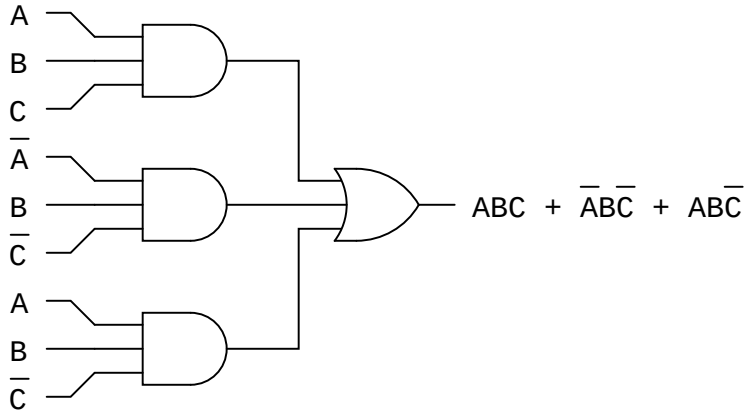
$$\overline{ABC} + A\overline{B}C$$

For an expression to properly follow the SOP or POS canonical form, no complementation bar should cover more than one variable!

Answer 4







Answer 5

The output of this circuit is high when $\overline{A}$ is true and $\overline{B}$ is true, or when A is true and B is true:

($\overline{A}$ AND $\overline{B}$) OR (A AND B)

Answer 6

For the first truth table: the output of this circuit is high when A is true and $\overline{B}$ is true and C is true:

$$A \text{ AND } \overline{B} \text{ AND } C$$

For the second truth table: the output of this circuit is high when $\overline{A}$ is true and $\overline{B}$ is true and C is true, or when $\overline{A}$ is true and B is true and $\overline{C}$ is true:

$$(\overline{A} \text{ AND } \overline{B} \text{ AND } C) \text{ OR } (\overline{A} \text{ AND } B \text{ AND } \overline{C})$$

Follow-up question: do you suspect we could write a formal Boolean expression for each of these truth tables? What would those expressions be, and what form would they be in (SOP or POS)?

Answer 7

To make things easier, I'll associate each of the switches with a unique alphabetical letter:

- SW1 = A
- SW2 = B
- SW3 = C
- SW4 = D

Now, the Boolean expression:

$$A B \overline{C} D$$

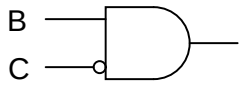
Answer 8

Original SOP expression:

$$\overline{A} B \overline{C} + A B \overline{C}$$

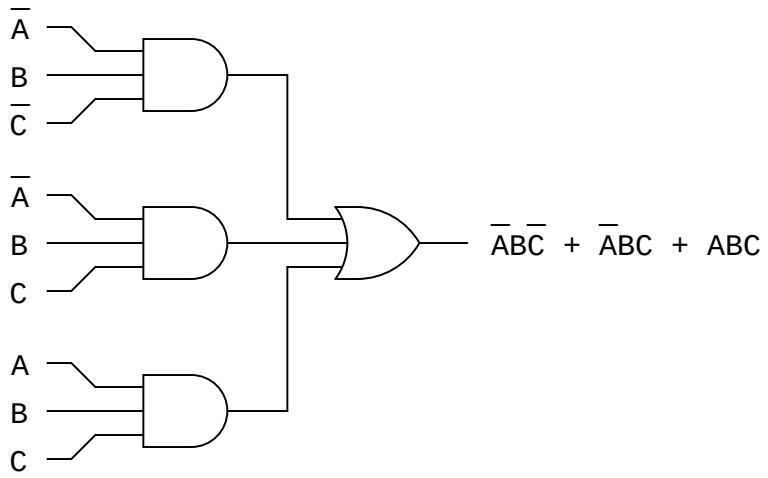
Simplified expression and gate circuit:

$$B \overline{C}$$

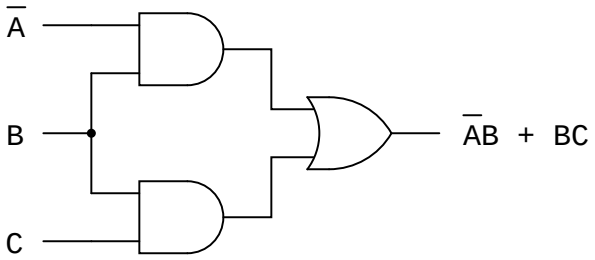


Answer 9

Original SOP expression and gate circuit:

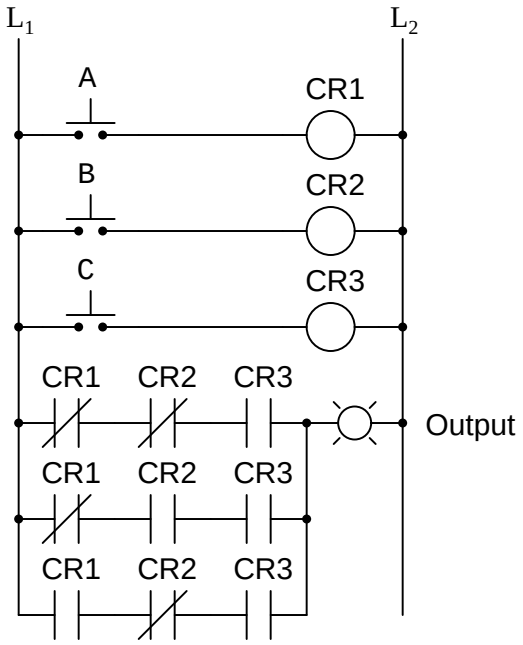


Reduced expression and gate circuit:

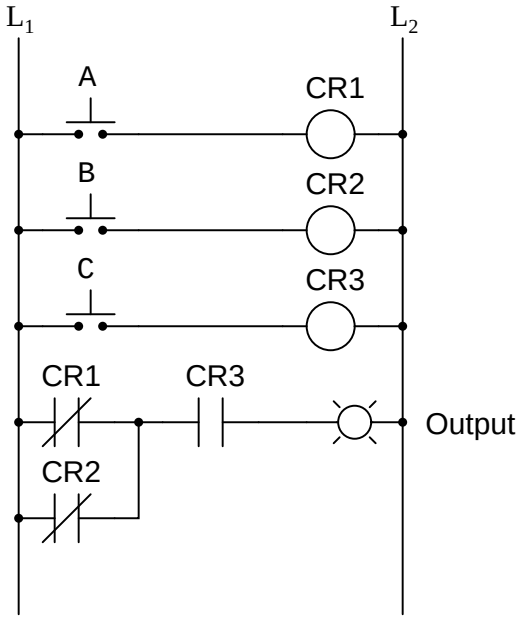


Answer 10

Original SOP expression and relay circuit:

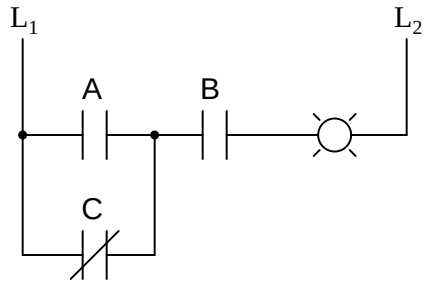


Reduced expression and relay circuit:

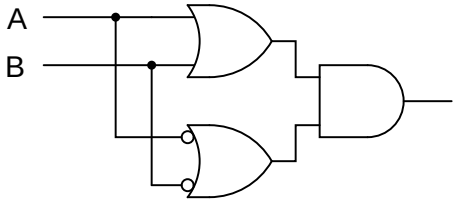


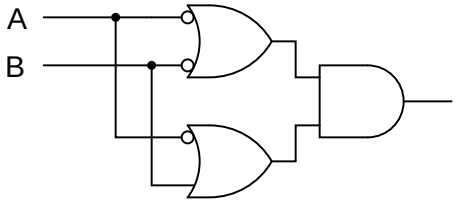
Answer 11

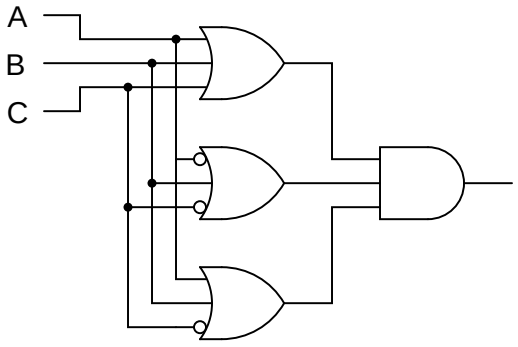
Simplest relay circuit possible:



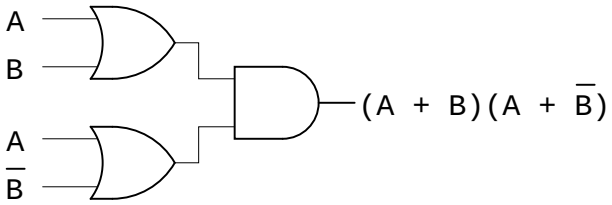
Answer 12

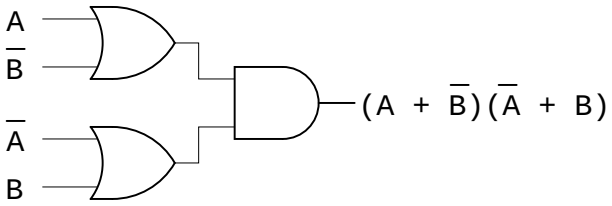


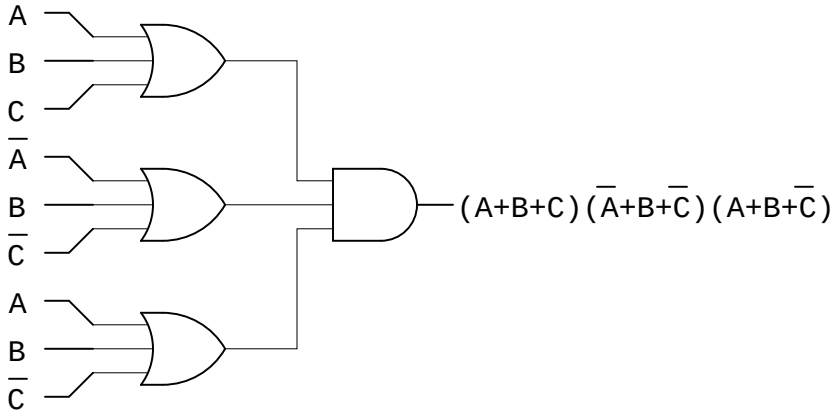




Answer 13







Answer 14

At least one of the sum terms must be equal to zero.

Follow-up question: in order for one of these terms to be equal to zero, thus making the whole expression equal to zero, what must be true about each of the Boolean literals (a *literal* is either a variable or the complement of a variable) within at least one of the sum terms?

Answer 15

Boolean expression for second gate circuit:

$$(\overline{A} + B)(\overline{A} + \overline{B})$$

Challenge question: we know that $\overline{AB}$ is also a valid Boolean expression for the first gate (NAND) circuit, in addition to $\overline{A} + \overline{B}$. Is there a rule you can think of to derive $\overline{AB}$ directly from an inspection of the truth table? Can you apply this rule to the second gate circuit and the manipulate the resulting expression using Boolean laws and rules to obtain the expression $(\overline{A} + B)(\overline{A} + \overline{B})$?

Answer 16

SOP expression:

$$\overline{A}\,\overline{B}\,\overline{C} + \overline{A}B\overline{C} + A\overline{B}C + AB\overline{C}$$

POS expression:

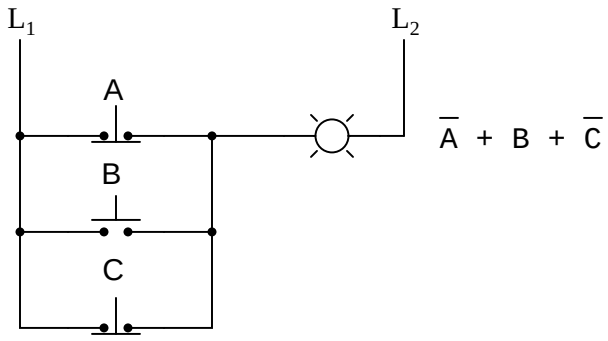
$$(A + B + \overline{C})(A + \overline{B} + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + \overline{C})$$

Note: before deciding which expression is simpler, remember that the POS expression must be distributed before we may apply any of the standard Boolean simplification rules.

Follow-up question: compare and contrast the procedures for generating SOP versus POS expressions from a truth table. What states (1 or 0) are you looking for when writing each type of expression? Explain why.

Challenge question: what truth table scenarios do you suppose would "favor" an SOP expression over a POS expression, and visa-versa? In other words, under what conditions does a truth table yield a simpler SOP expression, versus a simpler POS expression?

Answer 17

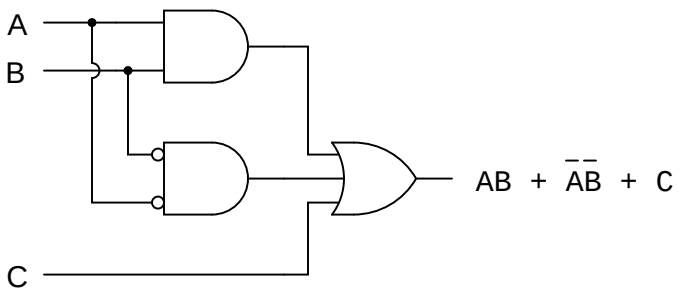


Answer 18

Original POS expression:

$$(A + \overline{B} + C)(\overline{A} + B + C)$$

Simplified expression and gate circuit:

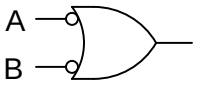


Answer 19

Original POS expression:

$$(\overline{A} + \overline{B} + C)(\overline{A} + \overline{B} + \overline{C})$$

Simplified gate circuit:



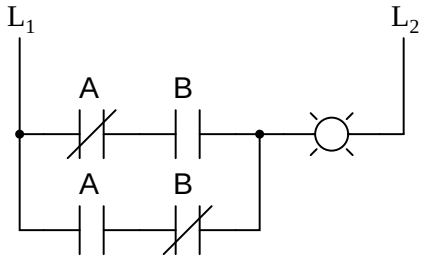
Challenge question: what other single gate type will satisfy the truth table (besides a Negative-OR gate)?

Answer 20

SOP form: $\overline{A}B + A\overline{B}$

POS form: $(\overline{A} + \overline{B})(A + B)$

I'll let you do the algebra showing these two expressions to be equivalent!



Answer 21

Raw (unsimplified) expressions:

$$a = (D + C + B + \overline{A})(D + \overline{C} + B + A)$$

$$b = (D + \overline{C} + B + \overline{A})(D + \overline{C} + \overline{B} + A)$$

$$c = D + C + \overline{B} + A$$

$$e = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A} + \overline{D}CB\overline{A} + D\overline{C}\,\overline{B}\,\overline{A}$$

Challenge question: use the laws of Boolean algebra to simplify each of the above expressions into their simplest forms.

Answer 22

Let the electrons themselves give you the answers to your own "practice problems"!

Notes

Notes 1

In order to familiarize students with the standard logic gate types, I like to given them practice with identification and truth tables each day. Students need to be able to recognize these logic gate types at a glance, or else they will have difficulty analyzing circuits that use them.

Notes 2

In order to familiarize students with standard switch contact configurations, I like to given them practice with identification and truth tables each day. Students need to be able to recognize these ladder logic sub-circuits at a glance, or else they will have difficulty analyzing more complex relay circuits that use them.

Notes 3

Even if your students have never heard of Boolean algebra before, they should still be able to determine which of the first four expressions are SOP and which are POS. If there is any confusion on this point, ask your students to define what "sum" and "product" mean, respectively, and then discuss what it means for an expression to be a product (singular) of sums (multiple), or a sum (singular) of products (multiple).

Notes 4

The translation from Boolean SOP to gate circuit should not be difficult. The point of this question is to get students thinking in terms of sum-of-products form, so they will be ready for the next step: linking this concept with truth tables.

Notes 5

Expressing truth table conditions "verbally" is a way to introduce students to the concept of deriving Boolean expression from them.

Notes 6

I find this "verbal" approach works well to introduce students to the concept of deriving Boolean expressions from truth tables.

Be sure to ask your students what Boolean expressions they derived for both these truth tables. Given the answers in "verbal" form, this should not be difficult for them!

Notes 7

This problem gives students a preview of *sum-of-products* notation. By examining the truth table, they should be able to determine that only one combination of switch settings (Boolean values) provides a "1" output, and with a little thought they should be able to piece together this Boolean product statement.

Though this question may be advanced for some students (especially those weak in mathematical reasoning skills), it is educational for all in the context of classroom discussion, where the thoughts of students and instructor alike are exposed.

Notes 8

Challenge your students to implement the original SOP expression directly with logic gates (three-input gates are acceptable to use).

Notes 9

Discuss with your students the utility of Boolean algebra as a circuit simplification tool. Ask your students to compare the original and reduced logic gate circuits, and comment on such performance metrics as reliability, power consumption, maximum operating speed, etc.

Notes 10

Discuss with your students the utility of Boolean algebra as a circuit simplification tool. Ask your students to compare the original and reduced logic gate circuits, and comment on such performance metrics as reliability, power consumption, maximum operating speed, etc.

Notes 11

Ask your students to show all their work in designing the relay circuit. By presenting their thought processes, not only do you help them consolidate their learning, but you also help the other students understand better by allowing them to learn from a peer.

Notes 12

The translation from Boolean POS to gate circuit should not be difficult. The point of this question is to get students thinking in terms of product-of-sums form, so they will be ready for the next step: linking this concept with truth tables.

Notes 13

The translation from Boolean POS to gate circuit should not be difficult. The point of this question is to get students thinking in terms of sum-of-products form, so they will be ready for the next step: linking this concept with truth tables.

Notes 14

This question foreshadows the derivation of POS expressions from truth tables. It also parallels the subject of polynomial roots in real-number algebra. For instance, the polynomial $x^2 - 2x - 8$ may be factored as such:

$$(x - 4)(x + 2)$$

If we were to set this equation equal to zero, the *roots* of the equation would be 4 and -2: the values for x which would make either one of the terms equal to zero.

You may find this mini-review of "normal" algebra to be helpful to your students, as they try to understand POS expressions.

Notes 15

The purpose of this question, if it isn't obvious to you by now, is to have students "discover" the technique for deriving POS expressions from truth tables, based on an evaluation of all the "low" output states.

Your more advanced students should enjoy the challenge question, for it allows one to generate Boolean expressions using a rule more similar to the first one learned: table-to-SOP.

Notes 16

This question is really asking students to compare and contrast SOP against POS expressions, rather than being a question specific to the given truth table. The actual expressions given in the answer are there only for "drill," so students may check their work. The *real* answers relate to the follow-up and challenge questions!

Notes 17

Ask students to contrast the difficulty of writing a POS expression for this function, versus an SOP expression. The difference in complexity is great! Also, ask them to compare the circuitry equivalent to each form of Boolean expression for this truth table. Which form yields a circuit with fewer gates?

Notes 18

Challenge your students to implement the original POS expression directly with logic gates (three-input gates are acceptable to use). Is the "simplified" POS expression shown in the answer really simpler in the context of real gate circuits? Ask your students what lesson this comparison holds for Boolean simplification techniques and their application to real-world circuits.

Notes 19

Challenge your students to implement the original POS expression directly with logic gates (three-input gates are acceptable to use). Is the "simplified" POS expression shown in the answer simpler in the context of real gate circuits? Ask your students what lesson this comparison holds for Boolean simplification techniques and their application to real-world circuits.

Notes 20

Ask your students how many of them used a truth table to solve this problem. This is a helpful hint, as a truth table for an Ex-OR gate is easy to remember (or look up), and it provides a basis for easily constructing an SOP or POS expression.

The Exclusive-OR function is very, very useful in logic circuits. It is well worth students' time to understand how to represent it in Boolean form (and no, not using that funny $\oplus$ symbol, either, but representing it in a form where all the standard laws of Boolean algebra apply!).

Notes 21

This shows a very practical example of SOP and POS Boolean forms, and why simplification is necessary to reduce the number of required gates to a practical minimum.

It has been my experience that students require much practice with circuit analysis to become proficient. To this end, instructors usually provide their students with lots of practice problems to work through, and provide answers for students to check their work against. While this approach makes students proficient in circuit theory, it fails to fully educate them.

Students don't just need mathematical practice. They also need real, hands-on practice building circuits and using test equipment. So, I suggest the following alternative approach: students should *build* their own "practice problems" with real components, and try to predict the various logic states. This way, the relay theory "comes alive," and students gain practical proficiency they wouldn't gain merely by solving Boolean equations or simplifying Karnaugh maps.

Another reason for following this method of practice is to teach students *scientific method*: the process of testing a hypothesis (in this case, logic state predictions) by performing a real experiment. Students will also develop real troubleshooting skills as they occasionally make circuit construction errors.

Spend a few moments of time with your class to review some of the "rules" for building circuits before they begin. Discuss these issues with your students in the same Socratic manner you would normally discuss the worksheet questions, rather than simply telling them what they should and should not do. I never cease to be amazed at how poorly students grasp instructions when presented in a typical lecture (instructor monologue) format!

A note to those instructors who may complain about the "wasted" time required to have students build real circuits instead of just mathematically analyzing theoretical circuits:

What is the purpose of students taking your course?

If your students will be working with real circuits, then they should learn on real circuits whenever possible. If your goal is to educate theoretical physicists, then stick with abstract analysis, by all means! But most of us plan for our students to do something in the real world with the education we give them. The "wasted" time spent building real circuits will pay huge dividends when it comes time for them to apply their knowledge to practical problems.

Furthermore, having students build their own practice problems teaches them how to perform *primary research*, thus empowering them to continue their electrical/electronics education autonomously.

In most sciences, realistic experiments are much more difficult and expensive to set up than electrical circuits. Nuclear physics, biology, geology, and chemistry professors would just love to be able to have their students apply advanced mathematics to real experiments posing no safety hazard and costing less than a textbook. They can't, but you can. Exploit the convenience inherent to your science, and *get those students of yours practicing their math on lots of real circuits!*