hw3-circuits-and-logic: Sample Solutions

CSE20S24

In this assignment, you will consider how circuits and logic can be used to represent mathematical claims. You will use propositional operators to express and evaluate these claims.

Relevant class material: Week 2 and Week 3.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw3-circuits-and-logic".

Assigned questions

1. Fixed-width addition.

   (a) (*Graded for completeness*) [1] Choose example width 5 first summand and width 5 second summand so that in the binary fixed-width addition (adding one bit at time, using the usual column-by-column and carry arithmetic, and ignoring the carry from the leftmost column), the example satisfies all three conditions below simultaneously

   (1) When interpreting each of the summands and the result in binary fixed-width 5, the result represents the actual value of the sum of the summands **and**

   (2) when interpreting each of the summands and the sum in sign-magnitude width 5, the result represents the actual value of the sum of the summands **and**

   (3) when interpreting each of the summands and the sum in 2s complement width 5, the result represents the actual value of the sum of the summands.

   Solution:

   Let the first summand be $(00011)_{2,5}$ and the second summand be $(00001)_{2,5}$. Computing the addition gets us

$$\begin{array}{cccccc} & & & 1 & 1 & \\ & 0 & 0 & 0 & 1 & 1 \\ + & 0 & 0 & 0 & 0 & 1 \\ \hline & 0 & 0 & 1 & 0 & 0 \end{array}$$

   Now let's check each of the three conditions:

[1]This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we expect you to include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

(1) Converting the summands from binary fixed-width 5 to decimal gives us $(00011)_{2,5} = 2 + 1 = 3$ and $(00001)_{2,5} = 1$. The sum $(00100)_{2,5} = 4 = 3 + 1$.

(2) With sign-magnitude, the left-most bit is the sign bit. In this case, both summands have a 0 as their left-most bit, so the summands are positive. Converting the summands from sign-magnitude width 5 to decimal gives us $[00011]_{s,5} = 1 \cdot 3 = 3$ and $[00001]_{s,5} = 1 \cdot 1 = 1$. The sum $[00100]_{s,5} = 4 = 3 + 1$.

(3) With 2s complement, the value denoted by the column of the left-most is subtracted from each summand. In this case, both summands have a 0 in that left-most position, so nothing is subtracted and the numbers are positive. Converting the summands from 2s complement width 5 gives us $[00011]_{2c,5} = 2 + 1 = 3$ and $[00001]_{2c,5} = 1$. The sum $[00100]_{2c,5} = 4 = 3 + 1$.

(b) (*Graded for correctness*) [2] Choose an example width 5 first summand and second summand so that in the binary fixed-width addition (adding one bit at time, using the usual column-by-column and carry arithmetic, and ignoring the carry from the leftmost column), the example satisfies all three conditions below simultaneously

(1) When interpreting each of the summands and the result in binary fixed-width 5, the result **does not** represent the actual value of the sum of the summands **and**

(2) when interpreting each of the summands and the sum in sign-magnitude width 5, the result **does not** represents the actual value of the sum of the summands **and**

(3) when interpreting each of the summands and the sum in 2s complement width 5, the result represents the actual value of the sum of the summands.

A complete solution will clearly specify each summand and the result of binary fixed-width addition with this choice of summands; will specify the value of each summand and the result for binary fixed-width 5, sign-magnitude width 5, and 2s complement width 5 (and include calculations connecting with the definitions of these representations to explain these values); and a conclusion connecting the calculations to the properties laid out in the question.

Solution:
Let the first summand be $(11001)_{2,5}$ and the second summand be $(01001)_{2,5}$. Computing the addition gets us

$$\begin{array}{cccccc}
1 & 1 & & & 1 & \\
 & 1 & 1 & 0 & 0 & 1 \\
+ & 0 & 1 & 0 & 0 & 1 \\
\hline
 & 0 & 0 & 0 & 1 & 0
\end{array}$$

Notice that we ignore the carry from the leftmost column. Now let's check each of the three conditions:

[2]This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

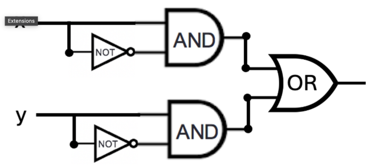
(1) Converting the summands from binary fixed-width 5 to decimal gives us $(11001)_{2,5} = 16+8+1 = 25$ and $(01001)_{2,5} = 8+1 = 9$. The sum $(00010)_{2,5} = 2 \neq 25+9$.

(2) In sign-magnitude, the left-most bit denotes the sign of the number. For the first summand, the left-most bit is 1, so its conversion to decimal is $[11001]_{s,5} = -1 \cdot (8+1) = -9$. For the second summand, the left-most bit is 0, so its conversion to decimal is $[01001]_{s,5} = 8+1 = 9$. The sum also has a 0 as its left-most bit, so its conversion to decimal is $[00010]_{s,5} = 2 \neq -9+9$

(3) In two's complement, the value denoted by the left-most bit is subtracted from the rest of the number. Since the first summand has a 1 in its left-most bit, its conversion to decimal is $[11001]_{2c,5} = -16+8+1 = -7$. Since the second summand has a 0 in its left-most bit, no subtraction is necessary. $[01001]_{2c,5} = 8+1 = 9$. The sum alsp has a 0 as its left-most bit, so its conversion to decimal is $[00010]_{s,5} = 2 = -7+9$.

2. Circuits.

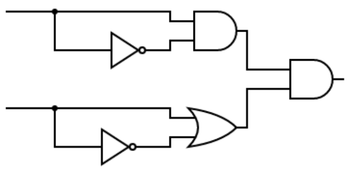
(a) (*Graded for completeness*) Consider the circuit below with inputs x and y. Identify a pair of gates that could be switched without changing the input-output table of the circuit. If you do, write out the input-output table that results, and briefly explain why this choice of gates works. If there is no such pair of gates, explain why not with reference to the definitions of the logic gates.



Solution:

Input-output table of the original circuit:

Input-output table				
x	y	$x \wedge \neg x$	$y \wedge \neg y$	$output = (x \wedge \neg x) \vee (y \wedge \neg y)$
1	1	0	0	0
1	0	0	0	0
0	1	0	0	0
0	0	0	0	0

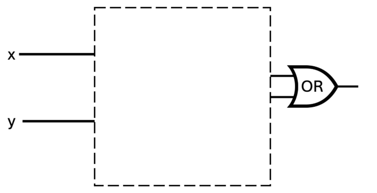


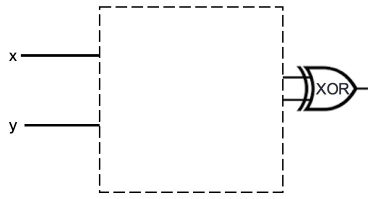
Switch: We can switch one of the AND gate with the OR gate. For example, we can switch the AND gate connected with y with the OR gate (as shown in the circuit above), and the new table would look like:

Input-output table				
x	y	$x \wedge \neg x$	$y \vee \neg y$	$output = (x \wedge \neg x) \wedge (y \vee \neg y)$
1	1	0	1	0
1	0	0	1	0
0	1	0	1	0
0	0	0	1	0

Explanation: Even though $y \vee \neg y$ will always output 1, we still have the $x \wedge \neg x$ which will always output 0. Since we moved one AND gate to the end, the input 0 to the AND gate will make sure the final output will still be constantly 0.

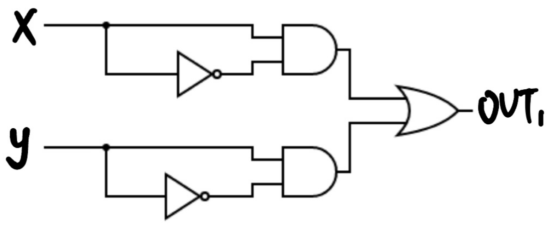
(b) (*Graded for correctness*) Is there a way to fill in the blank portion of the two logic circuits below *with the same gates connected in the same way* so that the resulting circuits have the same input-output value *even though* one uses an OR gate at the end and the other uses an XOR gate? If so, design the circuit that would be used, write out the input-output table that results, and briefly explain why your design works. If not, explain why not with reference to the definitions of the logic gates.

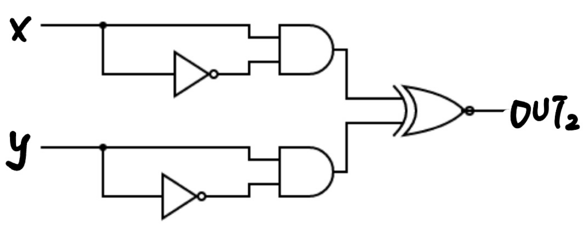




Sample solution:

Explanation: From 2(a), we learned that $x \wedge \neg x$ will always produce 0. If we combine it with the fact that $0 \vee 0 = 0 = 0 \oplus 0$, we can make a circuit where the two inputs to the final gate are both 0:





Input-output table:

Input-output table					
x	y	$x \wedge \neg x$	$y \wedge \neg y$	$OUT_1 = (x \wedge \neg x) \vee (y \wedge \neg y)$	$OUT_2 = (x \wedge \neg x) \oplus (y \wedge \neg y)$
1	1	0	0	0	0
1	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0

3. Compound propositions. The set of strings of length 4 whose characters are 0s or 1s is the result of four successive set-wise concatenations: $\{0,1\} \circ \{0,1\} \circ \{0,1\} \circ \{0,1\}$. Let's call this set X_4. Consider the function $f : X_4 \to X_4$ defined by

$$f(x) = \begin{cases} y & \text{when } (x)_{2,4} < 15 \text{ and } (y)_{2,4} = (x)_{2,4} + 1 \\ 1111 & \text{when } x = 1111 \end{cases}$$

for each $x \in X_4$. In other words, we can describe the function as: f takes a string, interprets it as the binary fixed-width 4 expansion of an integer, and then adds 1 to that integer (unless x is already representing the greatest integer that can be represented in binary fixed-width 4) and outputs the binary fixed-width 4 expansion of the result.

(a) (*Graded for completeness*) Fill in the blanks in the following input-output definition table with four inputs x_3, x_2, x_1, x_0 and four outputs y_3, y_2, y_1, y_0 so that $f(x_3x_2x_1x_0) = y_3y_2y_1y_0$.

x_3	x_2	x_1	x_0	y_3	y_2	y_1	y_0
1	1	1	1	1	1	BLANK1	1
1	1	1	0	1	1	1	1
1	1	0	1	1	BLANK2	1	0
1	1	0	0	1	1	0	1
1	0	1	1	1	1	0	0
1	0	1	0	1	0	1	1
1	0	0	1	1	0	1	0
1	0	0	0	BLANK3	0	0	1
0	1	1	1	1	0	0	0
0	1	1	0	0	1	1	BLANK4
0	1	0	1	0	1	1	BLANK5
0	1	0	0	0	1	0	1
0	0	1	1	0	1	0	0
0	0	1	0	0	0	1	1
0	0	0	1	0	0	1	0
0	0	0	0	BLANK6	0	0	1

Solution:

f(1111) = 1111, so BLANK1 = 1.

f(1101) = 1101 + 1 = 1110, so BLANK2 = 1

f(1000) = 1000 + 1 = 1001, so BLANK3 = 1

f(0110) = 0110 + 1 = 0111, so BLANK4 = 1

f(0101) = 0101 + 1 = 0110, so BLANK5 = 0

f(0000) = 0000 + 1 = 0001, so BLANK6 = 0

Final table:

x_3	x_2	x_1	x_0	y_3	y_2	y_1	y_0
1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1
1	1	0	1	1	1	1	0
1	1	0	0	1	1	0	1
1	0	1	1	1	1	0	0
1	0	1	0	1	0	1	1
1	0	0	1	1	0	1	0
1	0	0	0	1	0	0	1
0	1	1	1	1	0	0	0
0	1	1	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	0	1
0	0	1	1	0	1	0	0
0	0	1	0	0	0	1	1
0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	1

(b) (*Graded for correctness*) Construct an expression (as a compound proposition) for y_0 in terms of the inputs x_3, x_2, x_1, x_0. Justify your expression by referring to the definition of the logic gates XOR, AND, OR, NOT and the definition of the function f. Hint: our work on the half-adder might be helpful.

Solution:

If we firstly disregard the $x = 1111$ case, we notice that y_0 is the **sum bit** (s_0) of the **half-adder** (from Week 3 Wed) with inputs x_0 and 1. Hence, when $x \neq 1111$, $y_0 = x_0 \oplus 1 = \neg x_0$.

Then we can consider the $x = 1111$ case by adding an OR expression when $x_3 = x_2 = x_1 = x_0 = 1$, which can be represented by $x_3 \wedge x_2 \wedge x_1 \wedge x_0$.

Together, we reach the result $y_0 = (x_3 \wedge x_2 \wedge x_1 \wedge x_0) \vee (\neg x_0)$.

(c) (*Graded for correctness*) Construct an expression (as a compound proposition) for y_1 in terms of the inputs x_3, x_2, x_1, x_0. Justify your expression by referring to the definition of the logic gates XOR, AND, OR, NOT and the definition of the function f. Hint: our work on the half-adder might be helpful.

Solution:

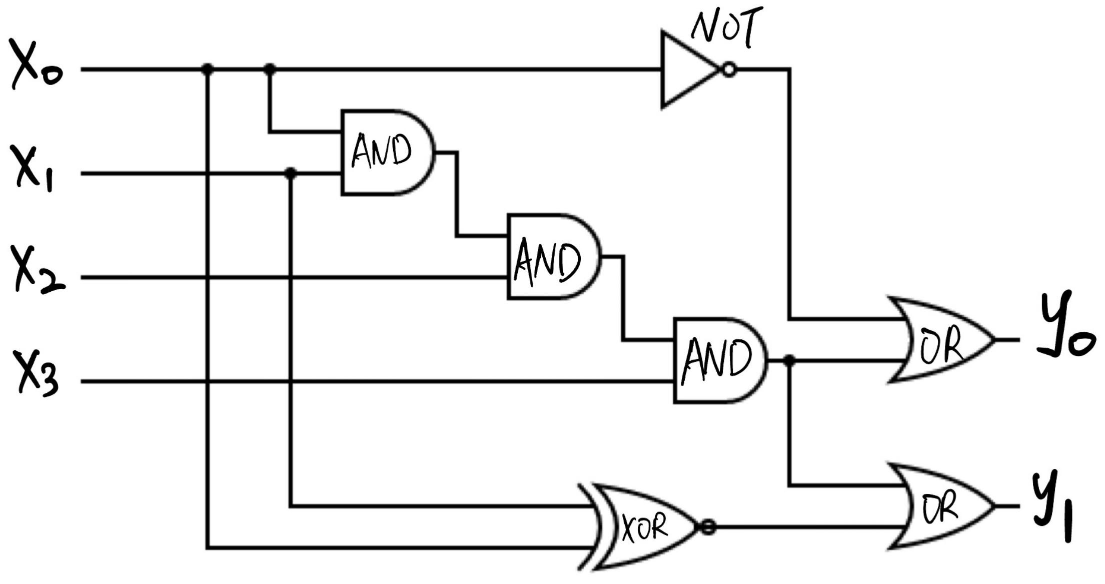
Similarly, if we firstly disregard the $x = 1111$ case, we notice that y_1 is the **second sum bit** (z_1) of the **two digits half-adder** (also from Week 3 Wed) with inputs x_1x_0 and 01. Hence, when $x \neq 1111$, $y_1 = (x_0 \wedge 1) \oplus (x_1 \oplus 0) = x_0 \oplus x_1$.

Then we can consider the $x = 1111$ case by adding an OR expression when $x_3 = x_2 = x_1 = x_0 = 1$, which can be represented by $x_3 \wedge x_2 \wedge x_1 \wedge x_0$.

Together, we reach the result $y_1 = (x_3 \wedge x_2 \wedge x_1 \wedge x_0) \vee (x_0 \oplus x_1)$.

(d) (*Graded for completeness*) Draw a combinatorial circuit corresponding to these compound propositions. Remember that the symbols for the inputs will be on the left-hand-side and the symbol for the outputs y_0 and y_1 will be on the right-hand side. Use gates (draw the appropriate shapes and add labels for clarity) and wires to connect the inputs appropriately to give the output.

Solution:



(e) (*Graded for completeness*) Construct expressions (as a compound propositions) for y_2 and y_3 in terms of the inputs x_3, x_2, x_1, x_0. Are these similar to the expressions for y_0 and y_1?

Sample Solution:

For y_2, we can apply similar logic using half-adders. From (c), we know the carry-out bit of the two digit half-adder with inputs x_1x_0 and 01 is $(x_0 \land 1) \land (x_1 \oplus 0) = x_0 \land x_1$. We can then apply the half-adder again to find the sum bit after adding it to x_2, which equals $x_2 \oplus (x_0 \land x_1)$. Again, we still need to add back the $x = 1111$ case to reach the final result: $y_2 = (x_3 \land x_2 \land x_1 \land x_0) \lor (x_2 \oplus (x_0 \land x_1))$.

For y_3, we can either apply half-adders again, or we can observe a pattern in the table that the $y_3 = 1$ only in the first 9 rows, and the first 8 rows correspond exactly when $x_3 = 1$. This makes sense as $x_3 = 1 \implies y_3 = 1$, and the only case when $x_3 = 0$ and $y_3 = 1$ is when $x_2 = x_1 = x_0 = 1$ which we can express as $x_2 \land x_1 \land x_0$. Hence, $y_3 = x_3 \lor (x_2 \land x_1 \land x_0)$.

The expression I construct for y_2 is similar to the expressions for y_0 and y_1, but the expression for y_3 is not.

4. Logical Equivalence. Imagine a friend suggests the following argument to you: "The compound

proposition

$$(x \vee y) \wedge z$$

is logically equivalent to

$$x \vee (y \wedge z)$$

because I can transform one to the other using the following sequence of logical equivalences:

$$(x \vee y) \wedge z \equiv (x \vee (y \wedge y)) \wedge z \equiv x \vee ((y \vee y) \wedge z) \equiv x \vee (y \wedge z)$$

because y is logically equivalent to both $y \wedge y$ and to $y \vee y$".

(a) (*Graded for correctness*) Prove to your friend that they made a mistake by giving a truth assignment to the propositional variables x, y, z so that the two compound propositions $(x \vee y) \wedge z$ and $x \vee (y \wedge z)$ have different truth values. Justify your choice by evaluating these compound propositions using the definitions of the logical connectives and include enough intermediate steps so that a student in CSE 20 who may be struggling with the material can still follow along with your reasoning.

Sample Solution:
Idea: By observing that the second proposition is always true when x is true, we will be trying to find an assignment to y and z to make the first proposition false when x is true. Then we can spot that $(x \vee y) \wedge z$ is false when z is false, which will result in the first proposition being false.

Assignment: $x = \mathrm{T}$, $y = \mathrm{T}$, $z = \mathrm{F}$.
Input-output table:

Input-output table						
x	y	z	$x \vee y$	$y \wedge z$	$(x \vee y) \wedge z$	$x \vee (y \wedge z)$
T	T	F	T	F	F	T

(b) (*Graded for completeness*) Help your friend find the problem in their argument by pointing out which step(s) were incorrect.
Sample Solution:
If we evaluate every proposition in the friend's sequence of the logical equivalences, we can see that:

Input-output table						
x	y	z	$(x \vee y) \wedge z$	$(x \vee (y \wedge y)) \wedge z$	$x \vee ((y \vee y) \wedge z)$	$x \vee (y \wedge z)$
T	F	T	F	F	T	T

Hence, we can conclude that the problem in the argument must be the step where the friend claims: $(x \vee (y \wedge y)) \wedge z \equiv x \vee ((y \vee y) \wedge z)$.

(c) (*Graded for completeness*) Give **three** different compound propositions that are actually logically equivalent to (and not the same as)

$$(x \vee y) \wedge z$$

Justify each one of these logical equivalences either by applying a sequence of logical equivalences or using a truth table. Notice that you can use other logical operators (e.g. $\neg, \vee, \wedge, \oplus, \rightarrow, \leftrightarrow$) when constructing your compound propositions.

Sample solution:

Easy ones: use $a = a \vee a = a \wedge a$, we can change the proposition to:

$$\begin{aligned}(x \vee y) \wedge z &= ((x \vee x) \vee y) \wedge z \\ &= ((x \vee x) \vee y) \wedge (z \wedge z) \\ &\text{etc.}\end{aligned}$$

More advanced ones: we can apply DeMorgan's Laws:

$$\begin{aligned}(x \vee y) \wedge z &= \neg(\neg(x \vee y) \vee \neg z) \\ &= \neg((\neg x \wedge \neg y) \vee \neg z)\end{aligned}$$

Bonus; not for credit (do not hand in): How would you translate each of the equivalent compound propositions in English? Does doing so help illustrate why they are equivalent?