HW 2

Due: 23:00, Wednesday May 22, 2019

CS ID 1: d6a2b

CS ID 2: b9i2b

Instructions:

1. Do not change the problem statements we are giving you. Simply add your solutions by editing this latex document.

2. If you need more space, add a page between the existing pages using the `\newpage` command.

3. Include formatting to clearly distinguish your solutions from the given problem text (e.g. use a different font colour for your solutions). Improperly or insufficiently typeset submissions will receive a penalty.

4. Export the completed assignment as a PDF file for upload to Gradescope.

5. On Gradescope, upload only **one** copy per partnership. (Instructions for uploading to Gradescope are posted on the assignments page of the course website.)

6. Late submissions will be accepted up to 24 hours past the deadline with a penalty of 20% of the assignment's maximum value

Academic Conduct: I certify that my assignment follows the academic conduct rules for of CPSC 121 as outlined on the course website. As part of those rules, when collaborating with anyone outside my group, (1) I and my collaborators took no record but names away, and (2) after a suitable break, my group created the assignment I am submitting without help from anyone other than the course staff.

Version history:

- 2019-05-15 00:15 – Initial version for release

1. **[10 marks]** *Tautologies.* Which of the following are tautologies? If the statement is a tautology, first give a proof using the appropriate rules of logic at each step of the proof.

   If something is not a tautology, then justify your answer by giving a counter-example, i.e., an assignment of truth values to variables which makes the proposition False.

   a. **[2 marks]** $(\sim p \wedge (q \rightarrow p)) \rightarrow \sim q$

   b. **[2 marks]** $\sim(a \wedge b) \rightarrow (b \rightarrow (a \oplus b))$

c. **[3 marks]** $\sim c \to ((\sim b \to c) \wedge (c \to \sim a))$.

d. **[3 marks]** $((a \wedge c) \to (b \to c)) \wedge (c \to a)$.

1(a).
Proof:
$\equiv (\sim p \wedge (q \rightarrow p)) \rightarrow \sim q$ [Start]
$\equiv (\sim p \wedge (\sim q \vee p)) \rightarrow \sim q$ [IMP]
$\equiv \sim(\sim p \wedge (\sim q \vee p)) \vee \sim q$ [IMP]
$\equiv (p \vee \sim(\sim q \vee p)) \vee \sim q$ [IMP]
$\equiv p \vee \sim(\sim q \vee p) \vee \sim q$ [ASS]
$\equiv \sim q \vee p \vee \sim(\sim q \vee p)$ [COM]
$\equiv (\sim q \vee p) \vee \sim(\sim q \vee p)$ [ASS]
$\equiv T$ [NEG]
□
This is a tautology.

1(b).
Proof:
$\equiv \sim(a \wedge b) \rightarrow (b \rightarrow (a \oplus b))$ [Start]
$\equiv (a \wedge b) \vee (b \rightarrow (a \oplus b))$ [IMP]
$\equiv (a \wedge b) \vee (\sim b \vee (a \oplus b))$ [IMP]
$\equiv (a \wedge b) \vee (\sim b \vee (\sim a \wedge b) \vee (a \wedge \sim b))$ [XOR]
$\equiv (a \wedge b) \vee (\sim b \vee (a \wedge \sim b) \vee (\sim a \wedge b))$ [COM]
$\equiv (a \wedge b) \vee (\sim b \vee (\sim a \wedge b))$ [ABS]
$\equiv (a \wedge b) \vee (\sim b \vee \sim a \wedge \sim b \vee b)$ [DIST]
$\equiv (a \wedge b) \vee (\sim b \vee \sim a \wedge T)$ [NEG]
$\equiv (a \wedge b) \vee (\sim b \vee \sim a)$ [I]
$\equiv (a \wedge b) \vee \sim(a \wedge b)$ [DM]
$\equiv T$ [NEG]
□
This is a tautology.

1(c).
Proof:
$\equiv \sim c \rightarrow ((\sim b \rightarrow c) \wedge (c \rightarrow \sim a))$ [Start]
$\equiv \sim c \rightarrow ((b \vee c) \wedge (c \rightarrow \sim a))$ [IMP]
$\equiv \sim c \rightarrow ((b \vee c) \wedge (\sim c \vee \sim a))$ [IMP]
$\equiv c \vee ((b \vee c) \wedge (\sim c \vee \sim a))$ [IMP]
$\equiv (c \vee b \vee c) \wedge (c \vee \sim c \vee \sim a)$ [DIST]
$\equiv (c \vee c \vee b) \wedge (c \vee \sim c \vee \sim a)$ [COM]
$\equiv (c \vee b) \wedge (c \vee \sim c \vee \sim a)$ [ID]
□
This does not simplify to T. A sample case to prove it's not a tautology is when b is 0 and c is 0. This is not a tautology.

1(d).
Proof:
$\equiv ((a \wedge c) \rightarrow (b \rightarrow c)) \wedge (c \rightarrow a)$ [Start]
$\equiv ((a \wedge c) \rightarrow (b \rightarrow c)) \wedge (\sim c \vee a)$ [IMP]
□
This does not simplify to T. A sample case to prove it's not a tautology is when a is 0 and c is 1. This is not a tautology.

2. **[10 marks]** Later in the semester we will discover strategies for proving that for integers a, b, c, d, and m, if $a \equiv b \bmod m$ and $c \equiv d \bmod m$, then $a \cdot c \equiv b \cdot d \bmod m$, and $a + c \equiv b + d \bmod m$. In this problem, we will investigate the meaning of these statements, and see their implications for number representation. Feel free to use a calculator for parts (a), (b), (f), (g), (h), (i) and (l).

   a. **[0.5 marks]** Write the 16-bit unsigned representations of 30924.

   b. **[0.5 marks]** What is 30924 mod 32, expressed in decimal?

   c. **[0.5 marks]** Write the 5-bit unsigned representation of your answer to part (b).

   d. **[0.5 marks]** Describe the relationship between your answers to parts (a) and (c).

   e. **[1 marks]** Explain how you can compute the remainder when 30924 is divided by 16 (in decimal) *in 10 seconds or less.*

f. **[0.5 marks]** If $a = 30924$, and $b = 48701$, find the least significant 5 bits of $a + b$ (in binary).

g. **[0.5 marks]** Find $30924 \cdot 8 \bmod 32$ (in decimal).

h. **[1 marks]** Write the 16-bit two's complement *signed* representation of -30924.

i. **[1 marks]** Find $-30924 \bmod 32$ (in binary).

j. **[1 marks]** Describe the relationship between your answers to parts (c) and (i).

k. **[0.5 marks]** If $a = -30924$, and $b = 16653$, find the least significant 5 bits of $a + b$ (in binary).

l. **[0.5 marks]** Find $-30924 \cdot 8 \bmod 32$ (in decimal).

m. **[2 marks]** By exploration, we have illustrated some of the implications of the simple rules of modular arithmetic. Show that the rule cannot be extended to the division operation, by finding a *counter example*. That is, find unsigned integers a, b, c, d, and m, so that $a \equiv b \mod m$ and $c \equiv d \mod m$, but $a/c \not\equiv b/d \mod m$.

2(a).
$30924_{10} = 0111100011001100_2$

2(b).
30924 mod 32 = 12

2(c).
$12_{10} = 01100_2$

2(d).
12 in binary appears in 5 least significant digits of 30924 in binary.

2(e).
I know that 30924 mod 32 = 12, and 16 is a common denominator of 32, so x mod 32 must output x mod 16, or a number in between 16 to 31. In this case, the answer to 30924 mod 32 = 12, so 30924 mod 16 must also = 12.

2(f).
$30924_{10} = 0111100011001100_2$
$48701_{10} = 1011111000111101_2$
Adding only the 5 least significant digits:
01100 + 11101 = 101001
Five least significant digits of a + b in binary is 01001.

2(g).
30924 mod 32 = 12
8 mod 12 = 8
12*8 = 96
96 mod 32 = 0
30924·8 mod 32 = 0

2(h).
$-30924_{10} = 1000011100110100_2$

2(i).
-30924 mod 32 = 20 = 10100_2

2(j).
They are both 5 least significant digits of the number that 32 mod into.

2(k).
$-30924_{10} = 1000011100110100_2$
$16653_{10} = 0100000100001101_2$
Adding only the 5 least significant digits:
10100 + 01101 = 100001
Five least significant digits of a + b in binary is 00001.

2(l).
-30924 mod 32 = 20
8 mod 12 = 8

20*8 = 160
160 mod 32 = 0
30924·8 mod 32 = 0

2(m).
3 = -32 mod 5
4 = 9 mod 5
-32/9 = -3.556
-3.556 mod 5 = 1.444
3/4 = 0.75
1.44 $\neq$ 0.75
Modular operation rule can't extend to division, proven by counter example above.

3. **[17 marks]** Logical Arguments

   a. **[4 marks]** Consider the following argument:

   > If there is a chance of rain, or he ate bean chili for lunch, Geoff will not ride his bicycle. Unless Geoff washed his car in the morning, there is no chance of rain. Today Geoff neither washed his car nor ate bean chili. Therefore, Geoff will ride his bicycle today.

   First represent this argument symbolically. Then determine whether it is valid or not. Justify your answer.

b. **[4 marks]** Consider the following argument:

> If Geoff works hard enough and does not get fired, then he will get paid. If he gets paid, then he will buy food to eat. Geoff has not bought food to eat. Therefore either Geoff did not work hard enough, or he got fired.

First represent this argument symbolically. Then determine whether it is valid or not. Justify your answer.

c. **[5 marks]** Consider the following argument:

> If 30,000 cookies disappeared from Christie's factory, then either Christie destroyed the cookies, or Keebler stole the cookies. If it is not the case that a tree-dwelling elf became the new CEO of Christie and the new Christie CEO colluded with Keebler, then 30,000 cookies disappeared from Christie's factory. If the new Christie CEO did not have a secret meeting with Keebler's industrial spies, then Keebler did not steal the cookies. Christie did not destroy the cookies, and the new Christie CEO had a secret meeting with Keebler's spies. Therefore, Christie's new CEO did not collude with Keebler! No collusion!

First represent this argument symbolically. Then determine whether it is valid or not. Justify your answer.

d. **[4 marks]** Determine whether the following argument is valid. If it is valid, show a formal proof and if it is invalid provide a counter-example, i.e. an assignment of truth values that demonstrate a contradiction.
This argument is _____. (Fill in the blank with "valid" or "invalid".)

$$\begin{array}{r} \sim p \wedge q \\ r \rightarrow p \\ \sim r \rightarrow (s \wedge t) \\ s \rightarrow (t \vee p) \\ \hline \therefore t \end{array}$$

Proof:

inference	rule

Invalidating truth assignment:

$r =$, $s =$, $t =$, $u =$, $w =$

3(a).
a = "There is a chance of rain"
b = "Geoff ate bean chili for lunch"
c = "Geoff will ride his bicycle"
d = "Geoff washed his car in the morning"

Argument:
[1]$(a \vee b) \rightarrow \sim c$
[2]$\sim d \rightarrow \sim a$
[3]$\sim d \wedge \sim b$
Therefore [Conclusion] c

Proof:
[5]$\sim(a \vee b) \vee \sim c$ [IMP 1]
[6]$\sim b$ [SPEC 3]
[7]$\sim d$ [SPEC 3]
[8]$\sim a$ [M.PON 2, 7]
[9]$\sim a \wedge \sim b$ [CONJ 6, 8]
[10]$a \vee b$ [DM 9]
[11]$\sim c$ [M.PON 1, 10]
□

Argument is not valid since premises are true, but conclusion is false.

3(b).
a = "Geoff works hard enough"
b = "Geoff gets fired"
c = "Geoff gets paid"
d = "Geoff buys food to eat"

Argument:
[1]$a \wedge \sim b \rightarrow c$
[2]$c \rightarrow d$
[3]$\sim d$
Therefore [Conclusion] $\sim a(+)b$

Proof:
[4]$a \wedge \sim b \rightarrow d$ [TRANS 1, 2]
[5]$\sim(a \wedge \sim b)$ [T.MOL 3, 4]
[6]$\sim a \vee b$ [DM 5]
□

Argument is invalid since there's at least one truth assignment where all the premises are true but the conclusion is false. A sample one is:
a = 0, b = 1, c = X, d = X

3(c).
a = "30,000 cookies disappeared from Christie's factor"
b = "Christie destroyed the cookie"
c = "Keebler stole the cookie"

d = "a tree-dwelling elf became the new CEO of Christie"
e = "the new Christie CEO colluded with Keebler"
f = "new Christie CEO have a secret meeting with Keebler's industrial spies"

Argument:
[1]$a \to b \vee c$
[2] $\sim(d \wedge e) \to a$
[3] $\sim f \to \sim c$
[4] $\sim b$
[5]f
Therefore [Conclusion] $\sim e$

Argument is invalid since there's at least one truth assignment where all the premises are true but the conclusion is false. A sample one is:
a = 1, b = 0, c = 1, d = 0, e = 1, f = 1

3(d).
This argument is valid.

Argument:
[1] $\sim p \wedge q$
[2]$r \to p$
[3] $\sim r \to (s \wedge t)$
[4]$s \to (t \vee p)$
$\therefore [Conclusion]t$

Proof:
[5] $\sim p$ [SPEC 1]
[6] $\sim r$ [M.TOL 2,5]
[7]$(s \wedge t)$ [M.PON 3, 6]
[8]t [SPEC 7]
□

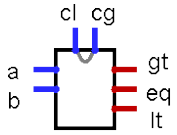
Argument is valid because conclusion and premises are true.

4. **[17 marks]** More CPU functions

Comparison is an important operation which a CPU needs to perform. It is a critical feature that allows us to write conditional branches in programs executed on this CPU. Logisim provides a *comparator* component in its arithmetic library - in this exercise we will construct our own components to perform arithmetic comparison between two unsigned integer inputs.

We will begin our construction with a 1-bit comparator, described as follows:

- 1-bit inputs: a, b, cl, cg
- 1-bit outputs: gt, eq, lt



For multi-bit comparison between two n-bit unsigned values $A = a_{n-1} \ldots a_0$ and $B = b_{n-1} \ldots b_0$, $gt = 1$ if $A > B$, $eq = 1$ if $A = B$, $lt = 1$ if $A < B$, and only one of gt, eq, and lt will be active. For single-bit comparison, cl and cg must be considered - these receive information about the comparison result of the bit(s) to the right of the current bit being compared. For example, suppose we are comparing bit 1 of $A = 10$ and $B = 11$ - what should the output be, assuming that bit 0 has already been compared, and $a_1 = b_1$?

a. **[5 marks]** Complete the function table for the 1-bit comparator, based on your understanding of the description above. Some rows have already been completed for you. For the comparison of some bit i, is it possible for the result from bits $i - 1 \ldots 0$ to be both greater than *and* less than?

a	b	cl	cg	gt	eq	lt
0	0	0	0	0	1	0
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0	0	0	1
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0	1	0	0
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0	0	1	0
1	1	0	1			
1	1	1	0			
1	1	1	1			

b. **[8 marks]** For this question, please refer to the Logisim User's Guide, section "Subcircuits" - "Creating circuits", and "Using subcircuits". In Logisim, choose "Project -> Add Circuit", and call the new circuit "comparator-1". In the drawing area for your comparator-1 component, use no more than 13 logic gates *in total* to design a circuit which implements the behaviour of this 1-bit comparator. Your solution should produce all of the three required outputs on a single circuit.

Your circuit diagram here:

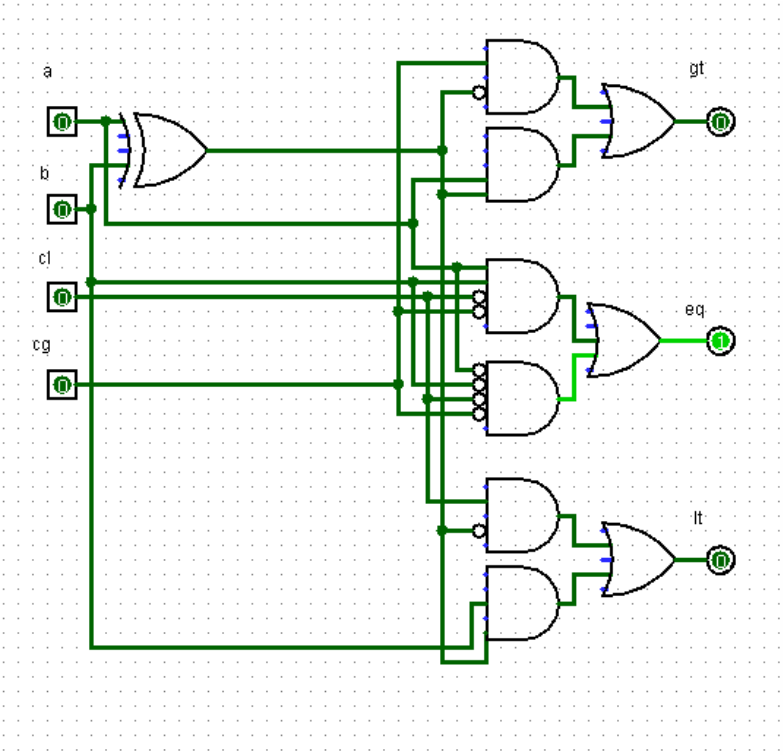
c. **[4 marks]** For this question, you will use the subcircuit for comparator-1 produced in the previous question. In the drawing area for your main circuit, use four (4) copies of your comparator-1 component, and any other gates necessary, as well as power/constant-1 and ground/constant-0, to construct a 4-bit unsigned comparator. Connect 4-bit inputs A and B (by setting data width to 4) and test your circuit. You will need to use splitters to access the individual bits of your 4-bit inputs.

Your circuit diagram here:

4(a).
Assuming carry lower = 1 if result from bits i-1...0 is A < B, and carry greater = 1 if result from bits i-1...0 is A > B. cl and cg can't be both 1 at once, so those values can be considered as don't-care terms. This assumption is in addition to the givens, where $gt = 1$ if $A > B$, $eq = 1$ if $A = B$, $lt = 1$ if $A < B$.

a	b	cl	cg	gt	eq	lt
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	0	0	1
0	0	1	1	X	X	X
0	1	0	0	0	0	1
0	1	0	1	0	0	1
0	1	1	0	0	0	1
0	1	1	1	X	X	X
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	X	X	X
1	1	0	0	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1
1	1	1	1	X	X	X

4(b).
Without doing a K-map or a logic simplification, you can just look at it and see that gt and lt is a mux output using a xor b as the selector. When a xor b is 1, gt and lt takes a and b, respectively. When a xor b is 0, gt and lt takes cg and cl, respectively. We can use mux equivalence in logic gates to wire up gt and lt. eq is just a simple sum of minterms.



4(c).
It's connecting the 4 corresponding bits of A and B to each comparator, and connecting gt and lt of comparator i-1 to cg and cl of comparator i. cg and cl of comparator 0 should be grounded since it's the first digit / no carry yet. Final output is hooked to the 4th comparator (comparator 3). You can keep adding comparators to make your input however many digits you'd like.