Name(s): Aaron Wang and Ethan Little
CSE 20221 Logic Design
HW04: Digital Circuits and Boolean Algebra

Preamble

1. Make sure you start from the Google Docs copy of this assignment, found in Google Drive.
2. Copy this assignment to your Google Drive folder and enter your answers in the boxes provided; **each empty box should be filled in with an answer**. You can either type your answers directly or scan handwritten work as long as it is legible. For solutions that require code, use a fixed-width font (Consolas preferred) no smaller than 10 points with proper indentation. Save your solutions as a single PDF file and upload them to Gradescope.
3. See the checklist at the end of the document for a list of what to turn in, and point totals.
4. Type your names at the top of this document - you are encouraged to work in groups of 2.
5. Always double-check your final PDF prior to uploading to Gradescope. In past semesters, students occasionally encountered problems copying output from a terminal to a Google Doc.
6. When copying text (**preferred**) or taking a screenshot, please use black text on a white background, or white text on a black background, otherwise it is very difficult to read.

Guidelines for algebraic manipulation problems:

1. **Show your work.**
2. You can work these by hand and take a photo (or scan) to upload and insert in your Google Doc. Please make sure the writing can be easily read in the final PDF.
3. Unless stated otherwise, you do **not** need to give the names of theorems/laws/properties used at each step.

Part 1

Problem 1

(a) Consider the following Boolean equation: `t = q's' + q(r' + rs').`
Fill in the truth table below for this equation.

q	r	s	t
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1

1	0	1	1
1	1	0	1
1	1	1	0

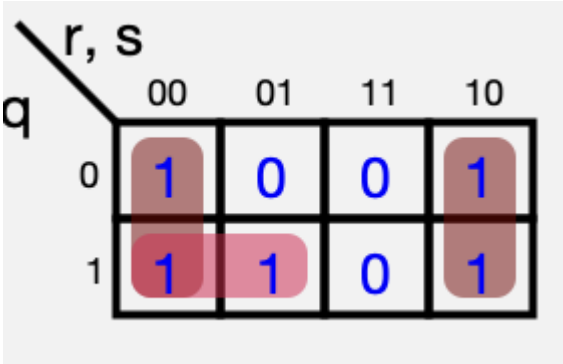
(b) Extract the sum-of-minterms from the truth table in (a):

```
t=q'r's' + q'rs' + qr's' + qr's + qrs'
```

(c) Use a Karnaugh map to minimize the logic equation from (b). For consistency, please place **q** as the single term in the K-map. After extracting the equation from the K-map, do not minimize further.

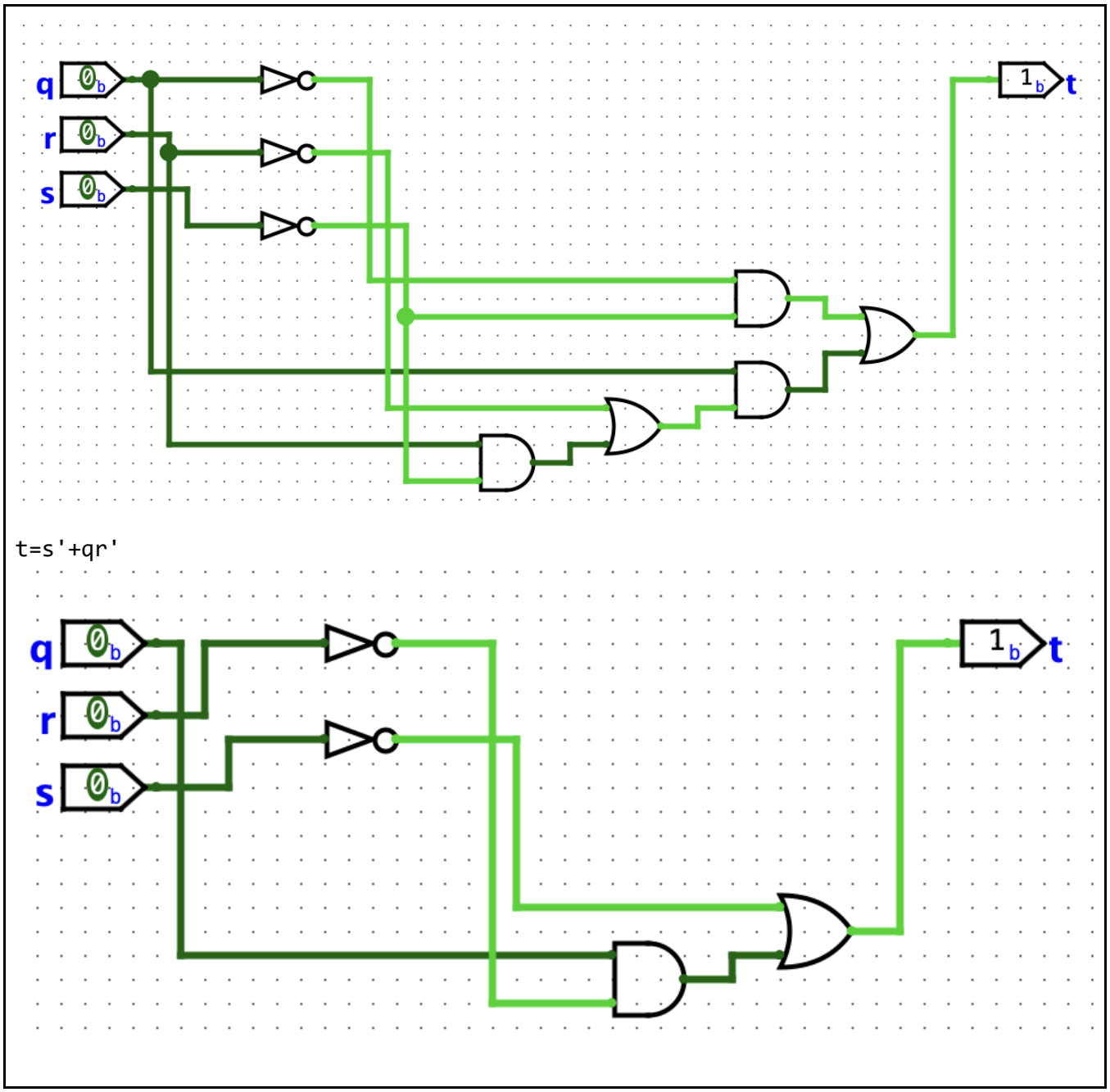
Important: minimizing an equation means going through the entire K-map process ending with the extraction of a new, minimized equation from the groupings in the K-map. In other words, stopping after showing the K-maps with groupings is not a complete answer.

```
t=s'+qr'
```



(d) Draw two circuits: one for the equation from (a), and one from the equation in (c).

```
t = q's' + q(r' + rs')
```



Sidenote (nothing to turn in here): think about the differences between these circuits. More gates → more transistors → more area/power and likely a slower circuit. A theme in this class is how different, equivalent circuits can be used to implement a Boolean equation. Care is needed to choose the best tools to minimize an equation before turning it into a circuit – e.g., properties of Boolean algebra and/or K-maps. Often, the K-map approach will yield a **less efficient** circuit, but one that looks "familiar" to the other circuits we design this semester. **We will sacrifice efficiency in many cases this semester in favor of this structured, "familiar" approach.**

Problem 2

A result of De Morgan's Law is that you can obtain the inverse of any expression by taking the inverse of each of the variables and replacing each of the ORs with ANDs and vice versa.

(a) Using repeated applications of De Morgan's Law to subexpressions, show that:

```
[(a'b + c)' (a + c')]' = ((a + b') c')' + (a'c)
```

```
(a'b + c)''+ (a + c')'
 = [(a'b)'c']'+ (a'c)
 = [(a+b')c']'+ (a'c)
```

(b) Convert the following expression to sum-of-products form via repeated application of De Morgan's Law:

```
f = [(m'n)(mn' + o)(m + n)]'
```

```
= (m'n)'+(mn' + o)'+(m + n)'
 = (m+n')+((mn')'o')+(m'n')
 = (m+n')+((m'+n)o')+(m'n')
 = (m+n')+(m'o'+no')+(m'n')
   M+n'+m'o'+no'+m'n' = f
```

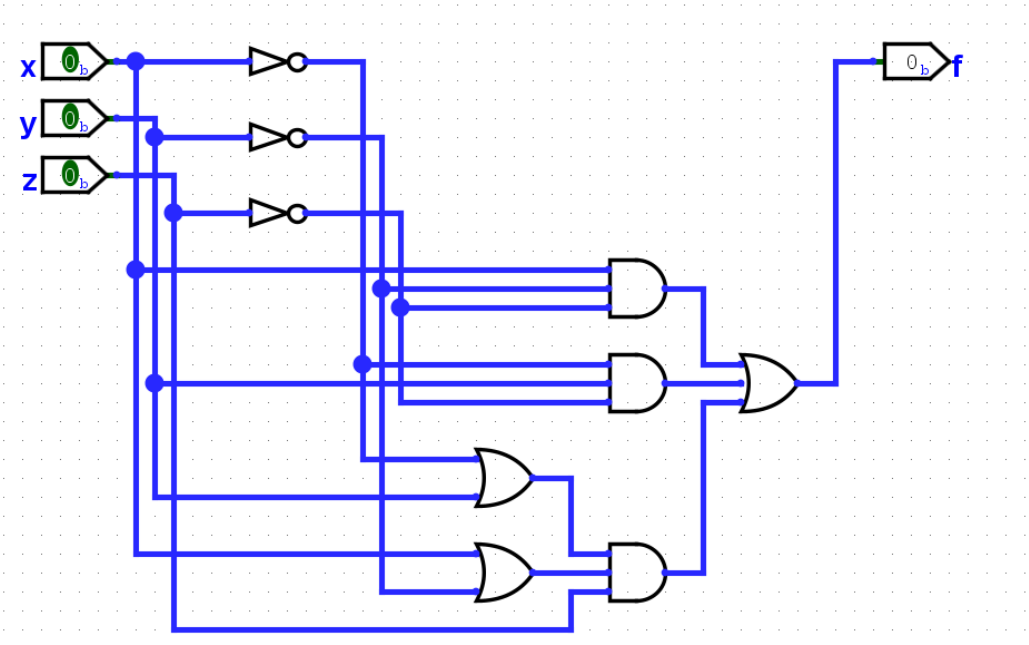
Problem 3

Express the following circuit as a Boolean logic equation and use the properties of Boolean algebra to show the circuit can be simplified to a single 3-input logic gate. In addition to showing your work, explain your answer so it is clear how your final equation matches/implements the 3-input gate.

Tip 0: print a copy (or use a tablet) and mark up the circuit to denote which wires are x, y, z (and x', y', z').

Tip 1: you can prove the original equation matches the 3-input gate by using a truth table.

Hint: look back at HW02, Problem 1, for how xor was implemented using just AND, OR, and NOT.



```
f=(xy'z')+(x'yz')+[z(x+y')(y+x')]
= (xy'z')+(x'yz')+[z(x(y+x')+y'(y+x'))]     Distributive(AND)
= (xy'z')+(x'yz')+[z(xy+xx'+y'y+y'x')]      Distributive(AND)
= (xy'z')+(x'yz')+[z(xy+0+0+y'x')]          Complement(AND)
= (xy'z')+(x'yz')+[z(xy+y'x')]              Identity(OR)
= (xy'z')+(x'yz')+zxy+zy'x'                 Distributive(AND)
= xy'z'+x'yz'+xyz+x'y'z                     Commutative
= x'y'z+x'yz'+xy'z'+xyz                     Commutative
Truth table derived from equation shown below.
```

Same Truth Table as an XOR gate, therefore this equation can be built with a 3 input XOR gate

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

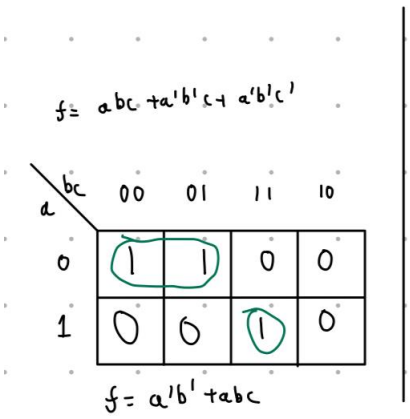
Problem 4

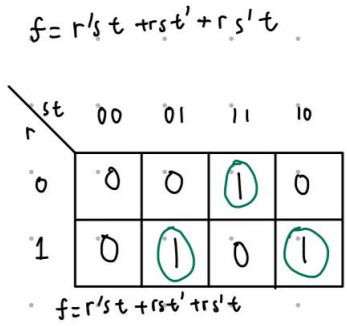
Use Karnaugh maps to minimize the following logic equations. (See the note from Problem 1 about what is expected when minimizing an equation using K-maps.)

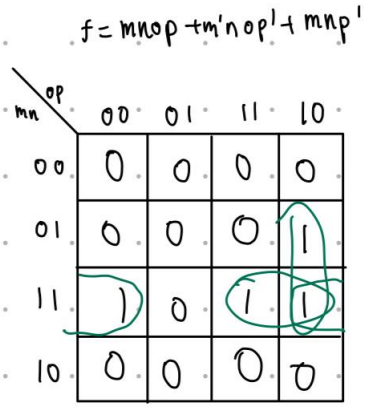
Insert your solutions below in any format that is convenient and easy to read, including photos of handwritten work, Google Drawings, etc.

```
a. f(a,b,c)   = abc + a'b'c + a'b'c'
b. f(r,s,t)   = r'st + rst' + rs't
c. f(m,n,o,p) = mnop + m'nop' + mnp'
```

(the lack of an o in the last term is deliberate)


f= abc +a'b'c+ a'b'c'
a
bc
00 01 11 10
0 1 1 0 0
1 0 0 1 0
f= a'b' +abc


f= r'st +rst' +rs't
r
st
00 01 11 10
0 0 0 1 0
1 0 1 0 1
f=r'st +rst' +rs't


f= mnop +m'nop' + mnp'
mn
op
00 01 11 10
00 0 0 0 0
01 0 0 0 1
11 1 0 1 1
10 0 0 0 0

f=nop'+mno+mnp'

x

Problem 5

A system has three input buttons labeled `in2`, `in1`, and `in0`. The buttons can be pressed individually, or multiple buttons can be pressed at once. In this problem, you will design a digital circuit that computes the total number of buttons pressed, and outputs the result as a 2-bit binary number.

Note: within the horizontal lines enclosing this example, + indicates addition.

For example, if buttons 2 and 0 are pressed (`in2` = 1, `in1` = 0, `in0` = 1), then the circuit should output `10` (`out1` = 1, `out0` = 0).

In other words:

```
              in2
              in1
+             in0
     out1 out0
```

Substituting the inputs for the example above:

```
               1
               0
+              1
            1 0
```

$1_2 + 0_2 + 1_2 = 10_2$, since two buttons were pressed.

---

a. Complete a truth table for the behavior of the circuit

in2	in1	in0	out1	out0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

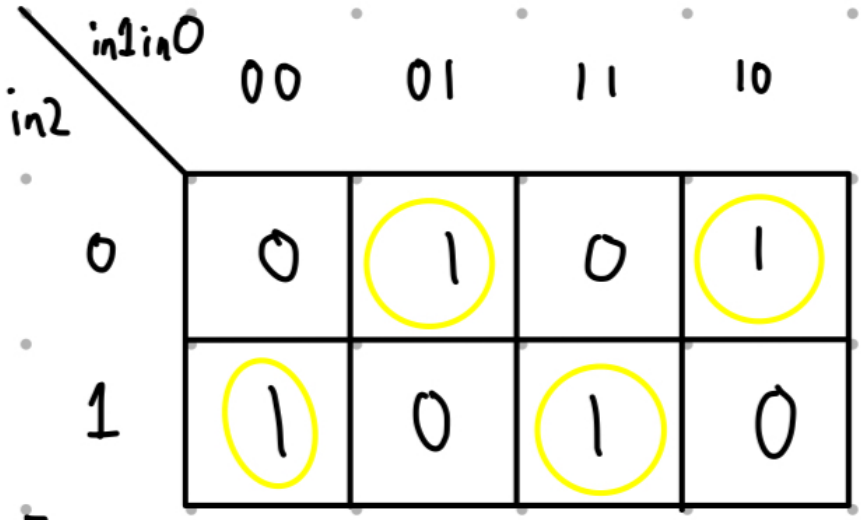
b. Use your truth table to derive equations for the circuit outputs in sum-of-minterms form.

out0=in2’in1’in0+in2’in1in0’+in2in1’in0’+in2in1in0
out1=in2’in1in0+in2in1’in0+in2in1in0’+in2in1in0

c. Minimize the logic for the outputs using Karnaugh maps. (See the note from Problem 1 about what is expected when minimizing an equation using K-maps.) For consistency,

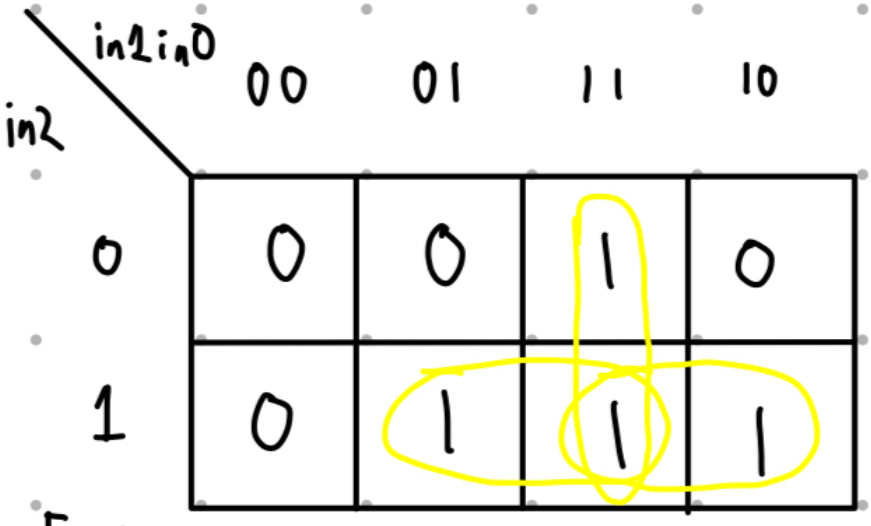
please place **in2** as the lone term in each K-map.

$out_0 = i_2' i_1' i_0 + i_2' i_1 i_0' + i_2 i_1' i_0' + i_2 i_1 i_0$



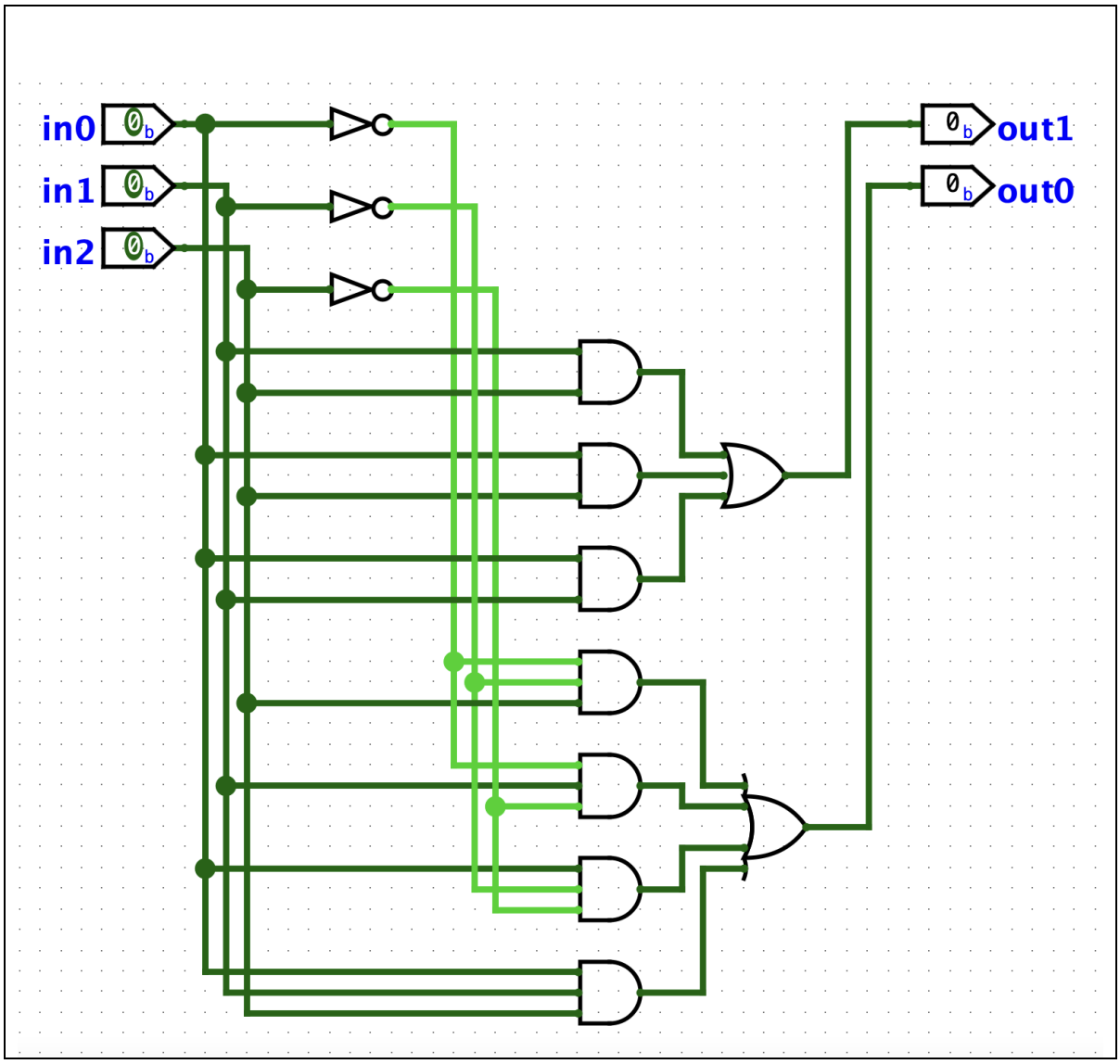
Equation from k-map: $out_0 = i_2' i_1' i_0 + i_2' i_1 i_0' + i_2 i_1' i_0' + i_2 i_1 i_0$

$out_1 = i_2' i_1 i_0 + i_2 i_1' i_0 + i_2 i_1 i_0' + i_2 i_1 i_0$



Equation from k-map: $out_1 = i_2 i_1 + i_2 i_0 + i_1 i_0$

d. Draw logic circuits that implement your minimized equations from c.



Part 2

Problem 1: Logisim-evolution Installation and Tutorial

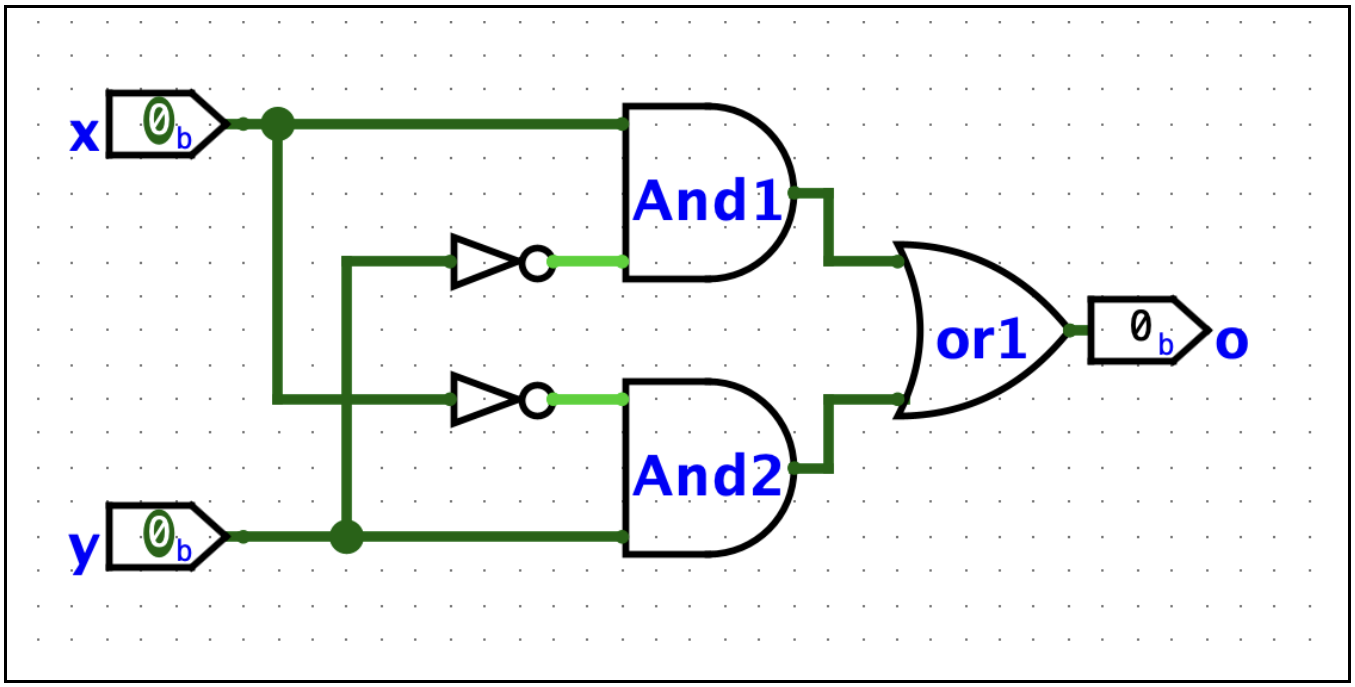
Going forward, our assignments will typically require you to implement and test one or more circuits using the Logisim-evolution digital circuit designer. You will need to install Logisim-evolution (free, open source) on your personal computer and complete the *Beginner's Tutorial* steps *0-4*.

1. First, you will need to install Logisim-evolution
    a. Download Logisim-evolution version 3.9.0. All versions are listed at this link
        i. For Windows, download logisim-evolution-3.9.0-x86.msi
        a. Open the downloaded .msi file. You will likely get a warning that "Windows protected your PC". Click "More Info" and then "Run anyway". Follow the installer prompts.
        b. Test the installation worked by hitting the Windows key and start typing "logisim". The newly-installed program should appear in the list, click it. Click **deny** if asked about access to the network (it does not need or request network access).
        ii. For MacOS, download logisim-evolution-3.9.0.dmg

a. Open the downloaded .dmg file and then drag the Logisim-evolution icon into the Applications icon.
b. Test the installation worked by opening Spotlight (CMD + spacebar) and start typing “logisim”. The newly-installed program should appear in the list, click it.
   i. Depending on your MacOS version, you may get a warning that Apple cannot check the app for malicious software. If so, open the Applications list in Finder and *Control-click* Logisim-evolution.app and click *Open*. Clicking Open will add an exception that allows the app to open.
iii. For Linux, let the instructor or TAs know if you need help compiling and/or installing.

2. In Logisim-evolution, click Help→ Tutorial and follow the *Beginner’s tutorial* steps *0-4*. In the box below, include a screenshot showing the final circuit with both inputs set to 1.
   a. Note: when “Ctrl” is mentioned in the tutorial, on a Mac you will need to use “Command”

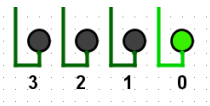
Screenshot showing completion of Logisim-evolution tutorial steps 0-4



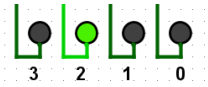
Designing a Tabulator/Display Decoder System

Your assignment is to design a system with 3 inputs that counts how many of the inputs are set to 1 and sets 1 of 4 outputs to a 1 to indicate this. Assume the inputs are connected to switches where on/1 is when the switch is flipped up. Assume the outputs are connected to a line of LEDs – as the switches are flipped, the LEDs will light up accordingly. For example:

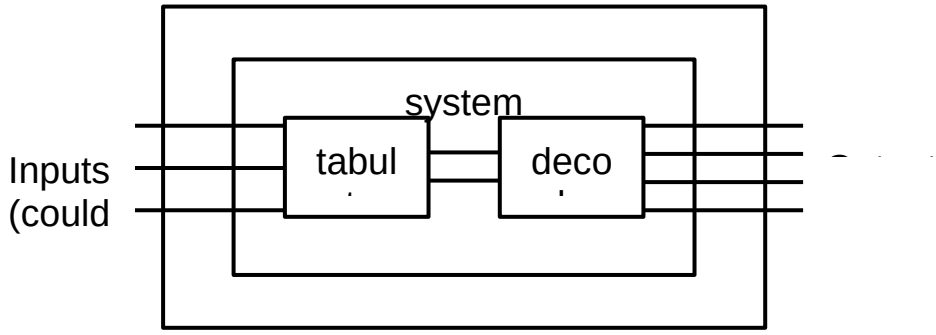
0 switches flipped: LEDs:



2 switches flipped: LEDs:



The system has two main components as illustrated below:



Tabulator: A 3-to-2 tabulator has 3 1-bit inputs and produces a 2-bit binary number as output. It counts the number of inputs that are 1s and produces the count as the output. **Note how this description matches what you did in Part 1, Problem 5.**

Decoder: A 2-to-4 decoder takes a 2-bit binary number as input and has 4 1-bit outputs. For a given input combination, exactly 1 of the outputs is set to a 1 and the rest are set to 0 (00 on the input enables output 0, 01 on the inputs enables output 1, etc.)

The **tabulator** and **decoder** should each be implemented in the same Logisim-evolution schematic. You will build them one at a time, testing each as you go, and then wire them together to test the behavior of the entire circuit.

You will follow a sequence of steps to design the system. Read the following summary first, and then work through each section in this document to complete these steps in order, showing your work in the boxes provided.

-Construct truth tables for the tabulator and decoder and translate the truth tables to Boolean equations. You have done this for the tabulator already in Part 1, Problem 5.

-Complete the circuit for the tabulator in Logisim-evolution and test its behavior.

-Complete the circuit for the decoder in Logisim-evolution and test its behavior.

-Wire up the tabulator and decoder and test the complete circuit.

-Break the circuit to see common sources of errors.

-Optional: connect inputs to switches and outputs to LEDs.

Problem 2: Truth Tables and Boolean Equations

Tabulator Boolean Equations

You can represent the outputs as the sum-of-minterms or minimized forms from Part 1, Problem 5. Copy your chosen equations into the box below so it is clear what approach you are taking:

```
out1 =i2i1+i2i0+i1i0
out0 = i2'i1'i0+i2'i1i0'+i2i1'i0'+i2i1i0
```

Decoder Truth Table

in1	in0	out3	out2	out1	out0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Decoder Boolean Equations

```
out3 = i1i0
out2 = i1i0'
out1 = i1'i0
out0 = i1'i0'
```

You can represent the outputs as a sum-of-minterms, or minimize them, it is up to you. If you minimize, **show your work or explain your reasoning** (e.g., if it matches a circuit we have seen before).

Problem 3: Implement and Test the Tabulator

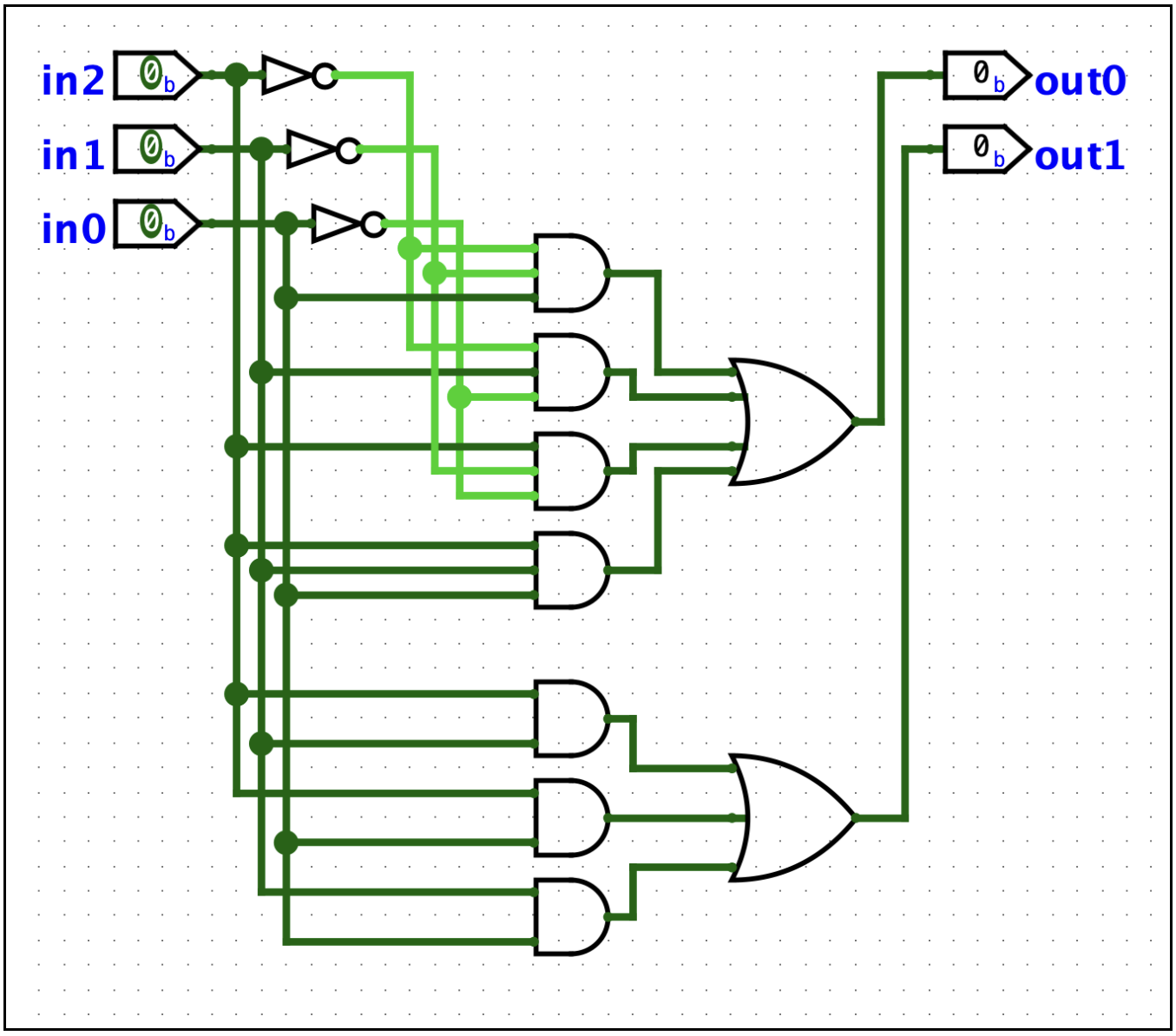
You will implement the tabulator in Logisim-evolution by adding gates and wires. Add input ports to the tabulator named `in2`, `in1`, and `in0` to match the behavior of your circuit to the truth table. Add output ports named `t_out1` and `t_out0`. These output ports will be used for testing the tabulator, but eventually you will remove them to connect the tabulator to the decoder.

Schematic and Simulation for Tabulator

Take a screenshot of your tabulator schematic and paste it in the box below. In addition, save the current circuit in a file called "tab_iso.circ" (click File → Save as). Copy `tab_iso.circ` to *one group member's* student dropbox[1] in our course directory, and list the full path to that directory here. Here is a link to a tutorial on how to do this.

[1] The courseware dropbox, one exists for each student's netid in `/escnfs/courses/fa24-cse-20221.01/dropbox`

tabulator schematic

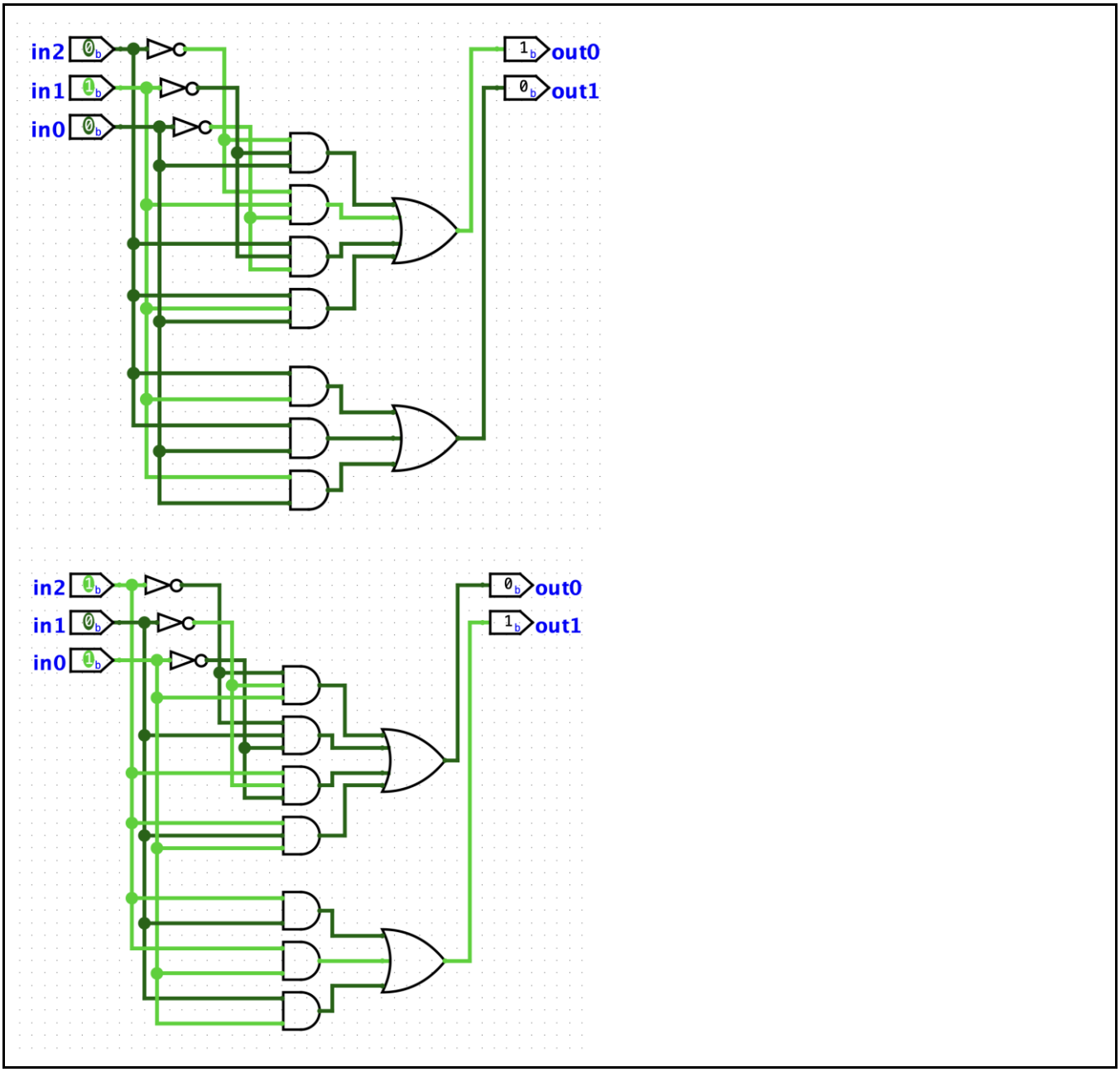


tabulator dropbox full path

```
/escnfs/courses/fa24-cse-20221.01/dropbox/awang27
```

Simulate using the poke tool to test all input combinations for `in`. Ensure the output is correct for all input combinations (tip: compare to your truth tables). Copy screenshots showing input combinations 010 and 101 into the box below.

screenshots of 2 tests using poke tool



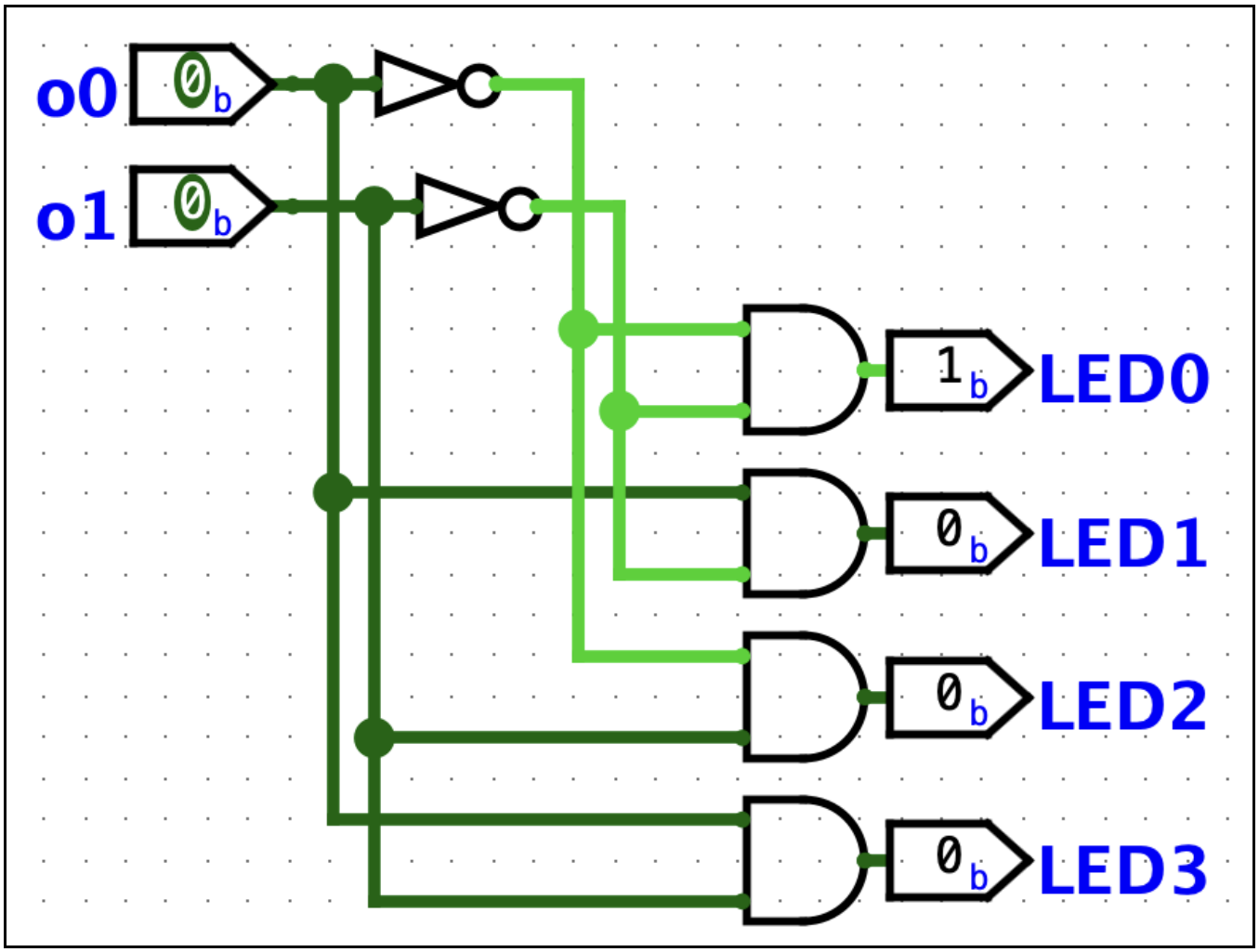
Problem 4: Implement and Test the Decoder

Implement the decoder in Logisim-evolution by adding gates and wires *to the same schematic as in Step 2*. Do **not** connect the decoder to the tabulator yet. Add input and output ports to the decoder. Name your input ports `d_in1` and `d_in0`; name your output ports `out3`, `out2`, `out1` and `out0` to match the behavior of your circuit to your Boolean equations. Eventually you will remove the input ports to connect the tabulator to the decoder.

Schematic and Simulation for Decoder

Take a screenshot of your decoder schematic and paste it in the box below. In addition, save the current circuit in a file called "tab_dec_iso.circ" (click File → Save as). Copy `tab_dec_iso.circ` to *one group member's* student dropbox in our course directory, and list the full path to that directory here.

decoder schematic

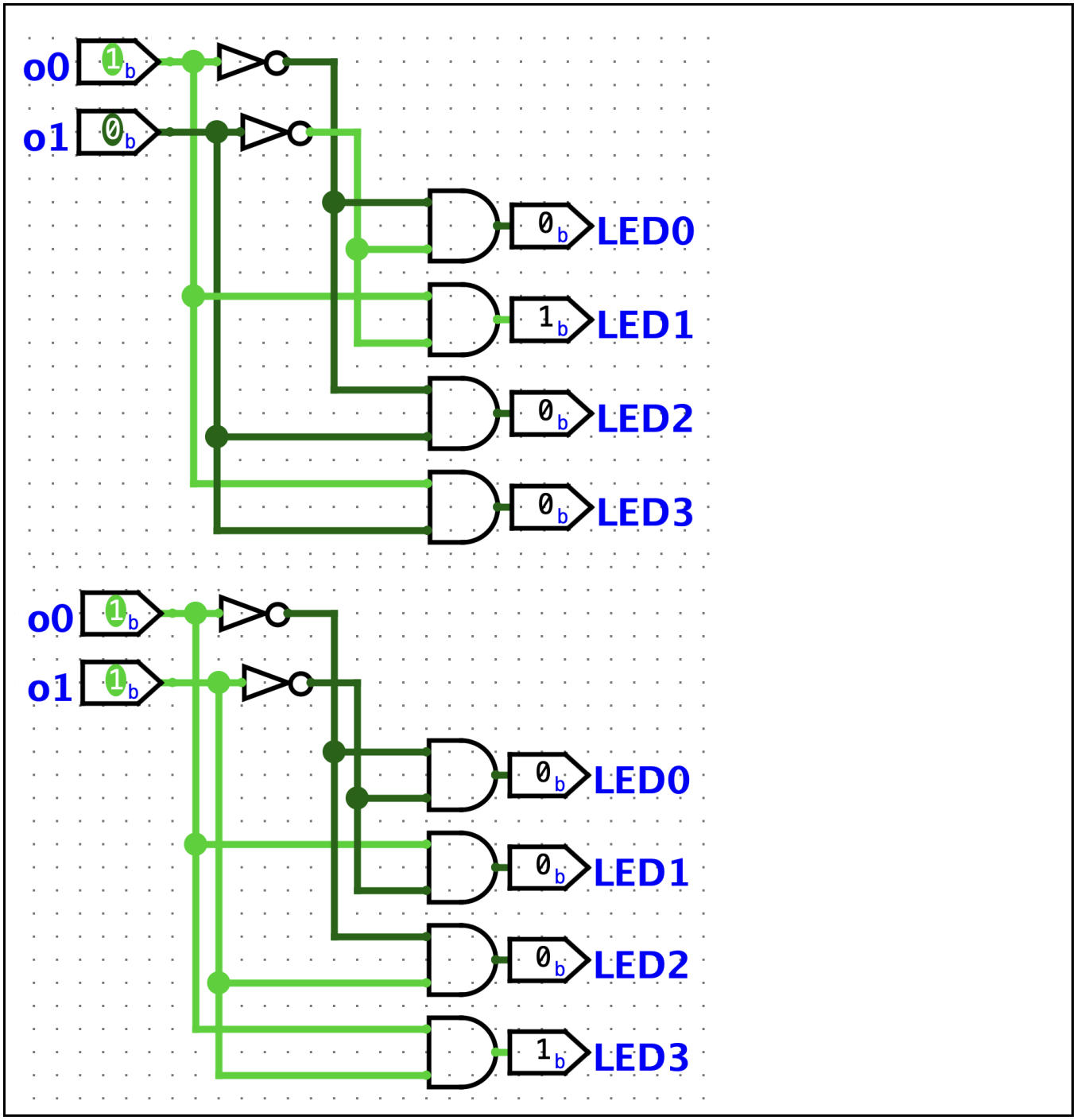


dropbox full path

```
/escnfs/courses/fa24-cse-20221.01/dropbox/awang27
```

Simulate using the poke tool to test all input combinations for `t_in`. Ensure the output is correct for all input combinations (tip: compare to your truth tables). Copy screenshots showing input combinations 11 and 01 into the box below.

screenshots of 2 tests using poke tool



Problem 5: Connect Circuits and Test

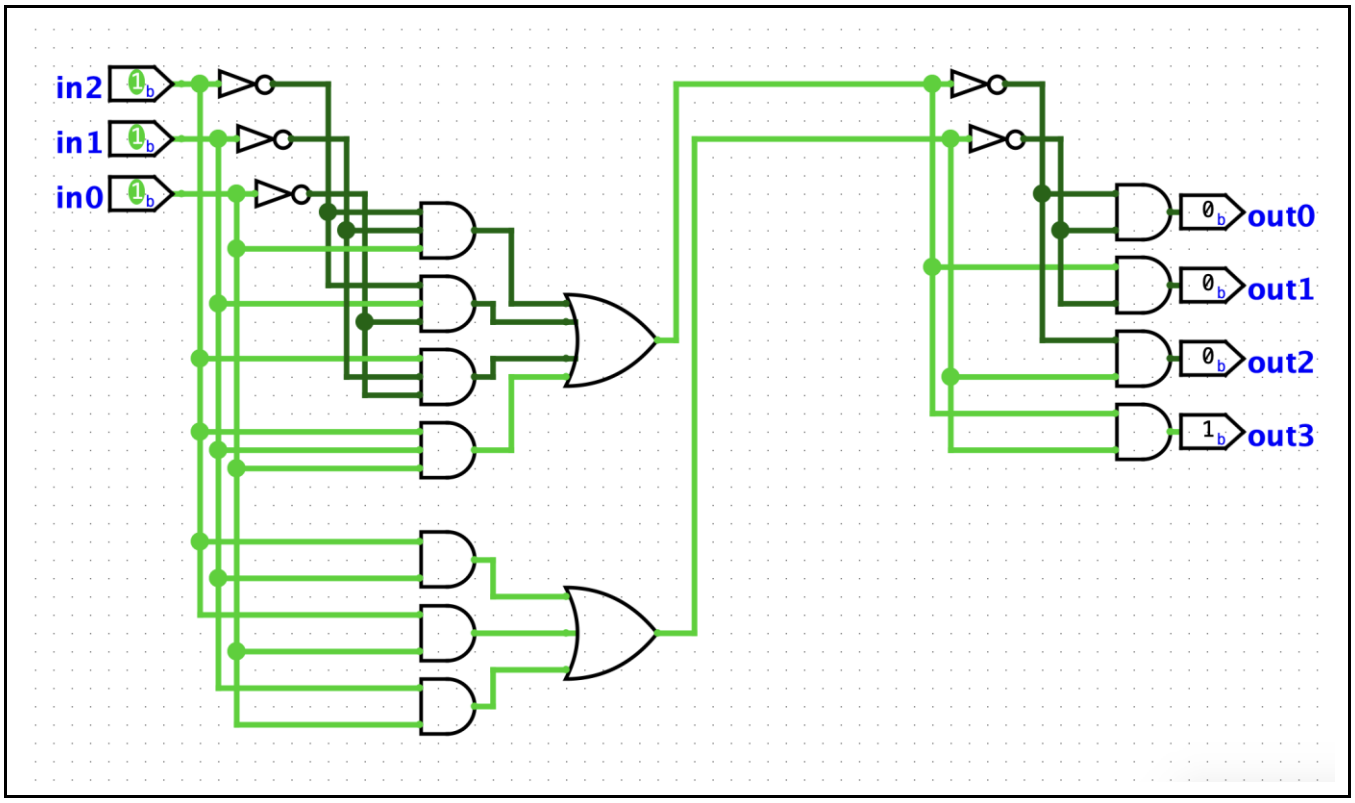
Now that you have verified the individual behavior of the two circuits, you can connect them together into a "system" component and test the behavior of the entire circuit. Remove the tabulator output ports and the decoder input ports and connect the tabulator to the decoder using wires.

Schematic and Simulation for System

Take a screenshot of your entire schematic and paste it in the box below. In addition, save the current circuit in a file called "`tab_dec_system.circ`" (click File → Save as). Copy `tab_dec_system.circ` to

one group member's student dropbox in our course directory, and list the full path to that directory here.

system schematic



dropbox full path

(Note: you likely copied the same path 3 times, but this is very helpful to the graders because of the way Gradescope is organized. This will only be done for HW04 to help graders give feedback as you learn to use the tool; in later assignments you will post only complete/final circuits to the dropbox.)

```
/escnfs/courses/fa24-cse-20221.01/dropbox/awang27
```

Simulate using the test vector tool to test all input combinations for `in`. To do this, download `test_vector.txt` from the HW04 folder in Google Drive. Click Simulate[2] → Test Vector… to open the Test Vector window. Then, click Load Vector and navigate to the file you downloaded. Open the `test_vector.txt` file and watch as it runs through the 8 input combinations. If any tests fail, check your circuit and test with the poke tool to fix any errors. Re-test using the given test vector until all tests pass. Copy a screenshot of the Test Vector tool showing all tests have passed into the box below.

[2] The Simulate menu option at the top of the window (Windows) or top of the screen (MacOS), between Project and FPGA. (Not the Design/Simulate tab on the left of the screen.)

screenshots of Test Vector pass

Passed: 8 Failed: 0

Status	in2	in1	in0	out3	out2	out1	out0
pass	0	0	0	0	0	0	1
pass	0	0	1	0	0	1	0
pass	0	1	0	0	0	1	0
pass	0	1	1	0	1	0	0
pass	1	0	0	0	0	1	0
pass	1	0	1	0	1	0	0
pass	1	1	0	0	1	0	0
pass	1	1	1	1	0	0	0

Problem 6: Break the Tabulator/Decoder System to Observe Symptoms of Common Mistakes
Now, you will introduce errors into your completed design to see how Logisim-evolution behaves. This will help you to identify errors in future assignments. For i-iii, introduce the error listed and answer each question in the boxes below.

i. Remove one of the internal wires connecting your tabulator to your decoder. Does the program visually indicate anything is wrong, and if so, how? What happens when you use the poke tool to test each input combination? What about the test vector?

```
Removing one of the internal wires connecting my tabulator to my decoder causes
the subsequent wired to turn blue, or red if they subsequently pass through a
logic gate.

When using a poke tool, the output becomes E for the two outputs that the wire is
connected to.

With test vector it also says E for those outputs and fails.
```

ii. Undo the changes you made in 6i. Change one of your AND gates to an OR gate (or vice versa). Does the program visually indicate anything is wrong, and if so, how? What happens when you use the poke tool to test each input combination? What about the test vector?

```
The program does not visually indicate anything is wrong.

However, when you use the poke tool, the truth table no longer matches as you
have changed the equation.

Upon opening test vector, some of the cases fail because the equation/circuit has
changed
```

iii. Undo the changes you made in 6i and 6ii. Rename one of your outputs to “I_AM_ERROR”. Does the program visually indicate anything is wrong, and if so, how? What happens when you use the poke tool to test each input combination? What about the test vector?

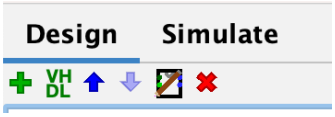
```
There are no issues until test vector, where it says the test does not match the
circuit because “out0 has no matching pin.”
```

Optional: Connect to Switches and LEDs

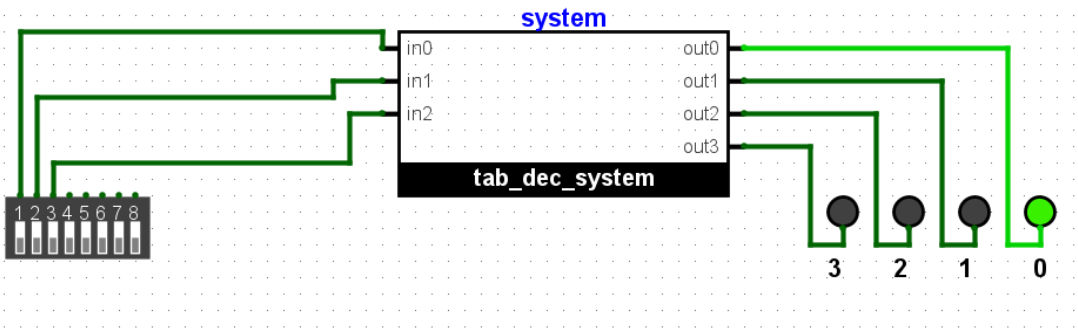
Complete this step If you would like a preview for future assignments where we build larger circuits from sub-circuits.

Rename your current circuit from `main` to `system`.

Click the + icon to add a circuit and name it *main*. Right click *main* and click “Set as Main Circuit”.



Now, you can connect the 3 inputs to a dip switch and the 4 outputs to LEDs. Both can be found in the Input/Output category. You can rename the sub-circuit by double-clicking it.



Part 1 - 65 points

Problem 1 (20 points)

	Deliverable	Points
1a.	Truth Table	6
1b.	Sum-of-minterms	4
1c.	K-map and minimized equation	5

1d.	Two circuits	5

Problem 2 (10 points)

	Deliverable	Points
2a.	Repeated applications of De Morgan's Laws	5
2b.	Convert to minimized sum-of-products	5

Problem 3 (5 points)

	Deliverable	Points
3.	Circuit to single gate	5

Problem 4 (15 points)

	Deliverable	Points
4.	K-maps, **3 in total**	5 points each

Problem 5 (15 points)

	Deliverable	Points
5a.	Three Button Circuit truth table	3
5b.	Sum-of-minterms from truth table	2
5c.	K-maps and minimal logic	5
5d.	Logic circuits	5

Part 2 - 55 points
Problem 1 (1 point)

	Deliverable	Points
1.	Screenshot showing completion of tutorial steps 0-4.	1

Problem 2 (10 points)

	Deliverable	Points
2a.	Tabulator Boolean equations (just copy from Part 1, Problem 5)	0
2b.	Decoder truth table	5
2c.	Decoder Boolean equations	5

Problem 3 (5 points)

	Deliverable	Points
3a.	Tabulator schematic (“copied” from Part 1, Problem 5)	0
3b.	Tabulator screenshots using poke tool and file in dropbox	5

Problem 4 (15 points)

	Deliverable	Points
4a.	Decoder schematic	10
4b.	Decoder screenshots using poke tool and file in dropbox	5

Problem 5 (15 points)

	Deliverable	Points
5a.	Schematic of tabulator and decoder connected into a complete system	10
5b.	System screenshots using test vector and file in dropbox	5

Problem 6 (9 points)

	Deliverable	Points
6.	Explanation of 3 mistakes (3 pts each)	9