Lab 1: Building Circuits

Tyseer Toufiq

January 16, 2024

The preparation exercises for this lab involve the design of digital circuits using AND, OR and NOT gates that implement a specified behaviour. However, to make this work in the real world, these gates are built into the 7400-series chips as described in the lab handout. So your circuit diagrams also need to specify how many of these chips your design needs, and how the pins on the chips you use correspond to the pins on your circuit diagram. Luckily, Logisim Evolution includes this information.

Your task is to design the circuits in Parts I and II using **only 74LS04 (NOT), 74LS08 (AND) and 74LS32 (OR) chips**. In Logisim Evolution, these correspond to 7404, 7408, and 7432, respectively, under `TTL`. Choose the actual pin numbers of the chips that you will use when you build your circuit and show them on your circuit diagram; this makes the construction of your physical circuit easier.

For each logic function, show all of the steps required to go from the specification to the final circuit. This includes:

- Deriving a truth table
- Drawing a schematic of the final circuit
- Assigning names to input and output wires on your schematic
- Deriving a simplified logic expression, when applicable

You can and should use Logisim Evolution to help you. The `Poke` tool helps you debug your schematic. The `Analyze Circuit` option helps you compare your own truth table derivation with what Logisim infers from your schematic. When your schematic is complete, use Logisim's `File -> Export Image` so that you can include it in your report. You can also export the contents of `Analyze Circuit` into Latex, which includes the truth table. An example is demonstrated on the next page. You can use it as a guide for Parts I and II.

Example

Consider the Boolean function: $f = ab + (c + b)$. Perform the following steps.

1. Write out the truth table for the function.

a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

2. Draw a schematic that implements the function using only the allowed chips.

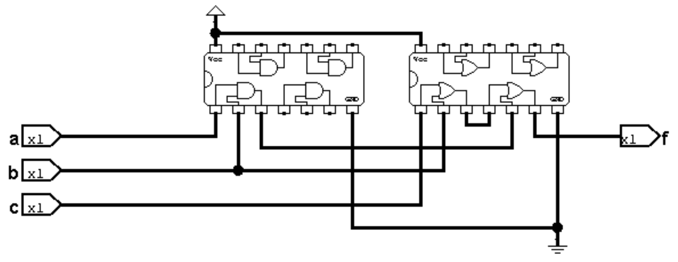

Figure 1: A schematic of the example function.

3. Is there a cheaper implementation for your design? Explain why or why not. If there is, include a schematic and discuss how much cheaper it is.

   Yes. We start by minimizing the original equation.

$$\begin{aligned} f &= ab + (c + b) \\ &= ab + c + b \\ &= ab + b + c \\ &= b + c && \text{(Covering)} \end{aligned}$$

   The schematic for the minimized equation would only require a single 74LS32 chip (Figure 2), which is 1 chip (and 2 gates) less than the original schematic (Figure 1).

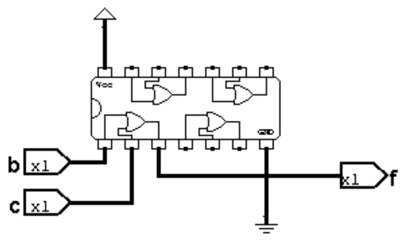

Figure 2: A schematic of the example function, with reduced cost.

Part I

Consider the Boolean function: $f = x\overline{s} + ys$ (a 2-to-1 multiplexer). Perform the following steps.

1. Write out the truth table for the function.

X	S	Y	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

2. Draw a schematic that implements the function using only the allowed chips.

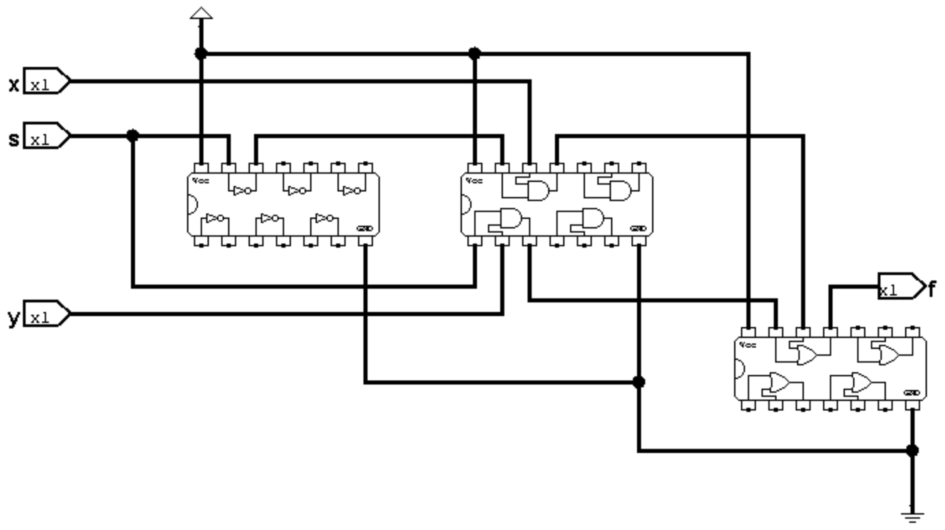

Figure 3: A schematic of a 2-to-1 multiplexer.

3. Is there a cheaper implementation for your design, assuming you are still limited to using the same three chip types? Explain why or why not. If there is, include a schematic and discuss how much cheaper it is.

I believe that we cannot simplify this further since the original boolean expression is in its simplest form. The original function consists of two distinct product terms that do not share a common factor, and no Boolean algebra identities or laws (such as absorption, distribution, or De Morgan's laws) can be applied to reduce the expression further. Furthermore I used the minimum number of chips in my implementation, thus this is the cheapest way to complement this circuit.

Part II

Consider the Boolean function: $f = \overline{a+b} + c\bar{b}$. Perform the following steps.

1. Write out the truth table for the function.

a	b	c	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

2. Draw a schematic that implements the function using only the allowed chips.

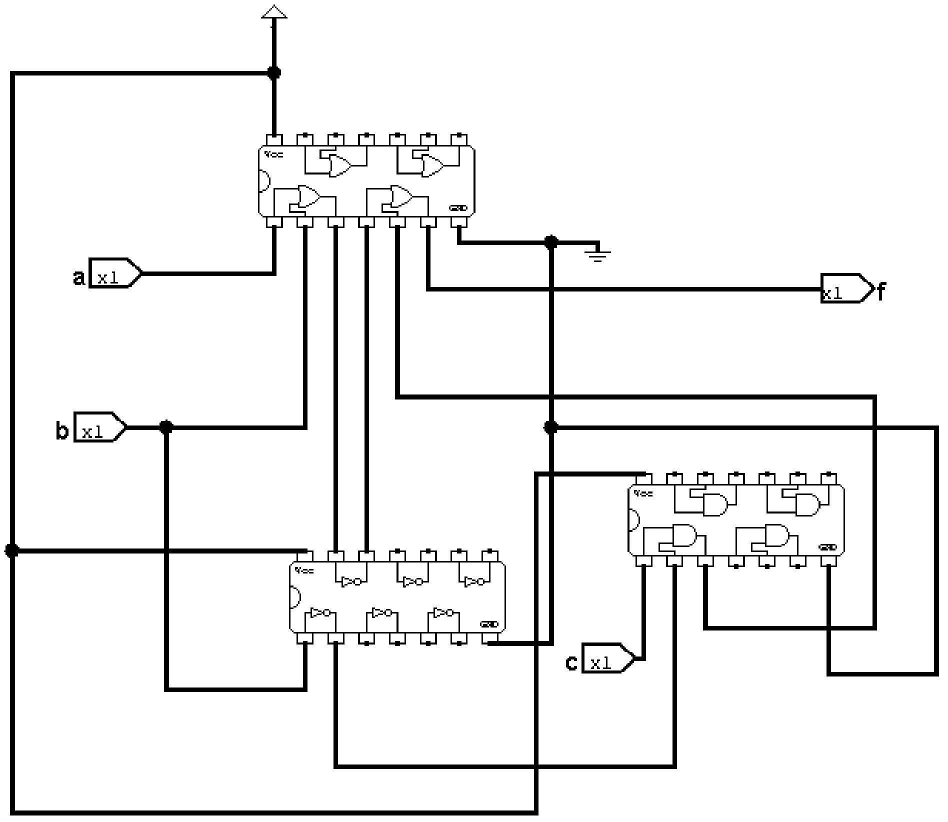

Figure 4: A schematic of the function from Part II.

3. Is there a cheaper implementation for your design, assuming you are still limited to using the same three chip types? Explain why or why not. If there is, include a schematic and discuss how much cheaper it is.

Notice that if we attempt to simplify the previous equation we have $\overline{a+b} + c\bar{b} = \bar{a}\bar{b} + c\bar{b} = (\bar{a}+c)\bar{b}$. By DeMorgan's Theorem, we can see that the new implementation would require fewer gates than the previous thus yielding a cheaper implementation. The last example will require a total of 5 gates, where as this one require 4 as seen below.

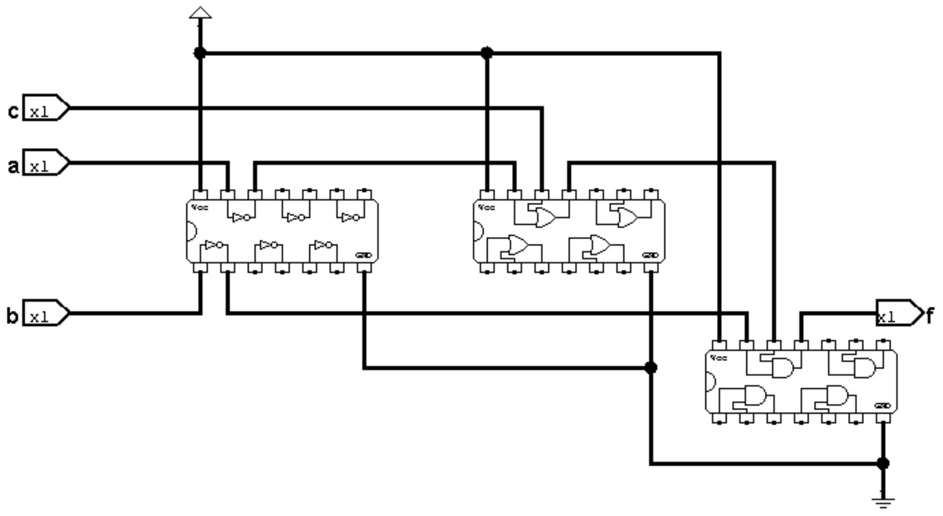

Figure 5: Cheaper Implementation for Part II.

Part III

Draw a schematic of the 2-to-1 mux, but instead of using the `TTL` chips, use the NOT, AND, and OR gates directly (i.e., under `Gates`).

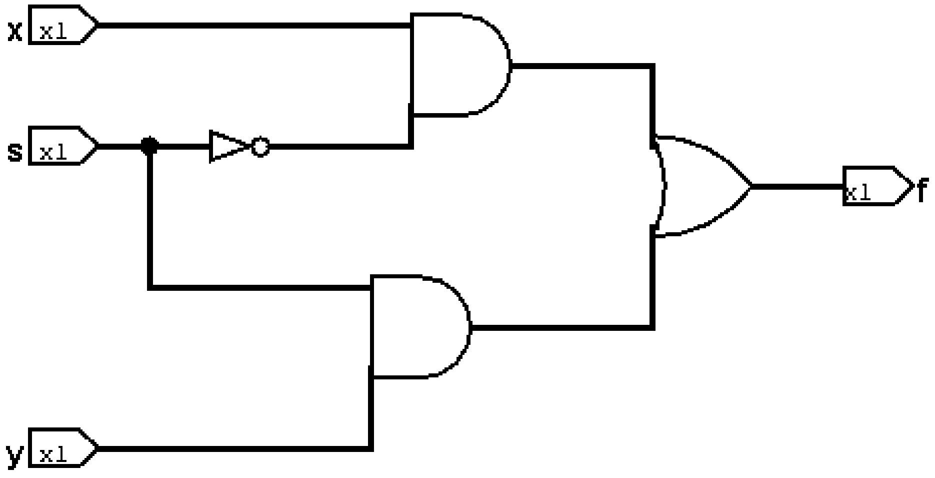

Figure 6: A schematic of the 2-to-1 multiplexer from Part III.