LAB 2: COMBINATIONAL CIRCUITS (ANALYSIS)

This lab will focus on exploring the uses of various key combinational circuit components like encoders and decoders, multiplexers (MUX), and comparators, for a special use case: Tita Jo's seaside eatery.

As in Lab 1, please use Logisim-evolution or CircuitVerse when instructed to implement or design a circuit.

Part 1: Reading Orders

Tita Jo has 8 items on her eatery's menu: Sinigang, Lumpia, Bulalo, Manok Inasal, Lechon Kawali, Kanin, Taho, and Halo-Halo.

With lots of customers (and most of the waiters unavailable for most of the weekdays when things get busy), you will have to design a Customer Console system, with an "Ordering Machine" around three components:

Customers will get a *bubble sheet*, which they put into a *bubble card reader*. The bubble card readers can be considered 3-output machines, which return an output 0 for the corresponding output rail if the bubble is unfilled, and 1 if it does. *However, the bubble card reader can only read up to three bubbles at a time.*

These bubble sheet readers feed their outputs to an "Order Machine", which returns 8 possible outputs, connected to 8 different LEDs in the kitchen, assigned to turn on when an order is made.

The local electronics shop has completely run out of 3-bit decoders. As such, you will only have access to 2-bit decoders, AND, OR, and NOT gates. **With these resources, design and implement an "Ordering Machine."** [!]

[Note: Can associate each order w/ a 3-bubble combo, itself associated w/ a 3-bit input. And for half of these inputs, the MSB is 1 and the other 0. Can just use a 2-bit decoder for the first 4 and make the first LEDs turn on when AND(x', DECODER(y,z); the other 4 can be handled with another 2-bit decoder and turn the other 4 LEDs on when AND(x, DECODER(y,z).]

Part 2: Accepting by Availability

[Ingredient Availability indicators: 8 of them, fed to 8-to-3 binary encoder;]

inquiry rail for S:

S_i = 1 <iff> the customer has requested it

availability rail for D:

D_i = 1 <iff> the product is available

Availability of ingredients can vary greatly in the eatery from day to day, and *it is very important to ensure orders for unavailable dishes are not accepted*. In other words, if a customer orders a dish, but the ingredients are not available, we need the corresponding light indicator in the kitchen *not to turn on*. As such, the "Order Machines" implemented in Part 1 must receive an "Availability Mechanism" upgrade.

To help achieve this, inventory is taken, and 8 other inputs ($I_0, I_1, \dots, I_7$) are toggled on and off every day, with each I_j corresponding to one of the dishes, being 1 if the dish's ingredients are available, and 0 if not.

Your electronics shop, this time, has multiplexers (yay!), but supply is limited, and you only have enough for one per Order Machine. **Implement the "Availability Mechanism" upgrade to the Customer Console.**

[note to self: need to turn S inputs into 3-bits; order inputs must go to S; Availability goes to D]

Part 3: Payment

The last interaction left to automate is the customer payment. Each of the dishes cost an integer multiple of ₱5. (e.g., an order of Lumpia costs ₱5, an order of Lechon Kawali costs ₱10, and so on.) **Assign a price to each dish, and make sure each price is a unique multiple of ₱5, up to ₱40 at most.**

Tita Jo needs a new Payment Mechanism to be implemented into the Customer Console, as an additional check before an order can be accepted, in the same way the Availability Mechanism was a check before an order can be accepted (i.e., for the appropriate LED in the kitchen to switch on).

When paying, customers deposit their coins into a coin machine, where only ₱5 coins are accepted. This coin machine has a useful feature: it has 8 outputs ($C_1, C_2, \dots, C_8$), only one of which is "on" (i.e., equal to 1) at a given time. In particular, if the Customer pays j many ₱5 coins (totaling a payment of ₱(5*j)), then C_j=1, while C_k=0 for k ≠ m.

Implement the "Payment Mechanism" update to the Customer Console. You will only have access to AND, OR, and NOT gates, but if you can answer the question in the hints below, you can gain access to another component.

Money is also tight; the business cannot afford more than three additional components in this design.

(Hint: In Part 1, you were able to associate each of the 8 dishes with a three-bit input, which you then re-translated by mapping each unique three-bit to a unique element in a set of 8 outputs. What electronic component will help you to do this in reverse, mapping each element in a set of 8 inputs to a unique three-bit output?)