Lab 5: Digital Logic

EG-UY 1003 Y1B

Jason Yao

Team members: Alice (Nanda) Ross, Jeremiah Nofrada

Date of experiment: 20th July 2015

Date due: 22th July 2015

**Abstract**

The objective of this lab was to design a combinational logic circuit that would activate only under specific conditions, and to implement it in both the LabVIEW software and the NI-ELVIS II+ hardware prototyping board. The circuit was successfully designed in LabVIEW and built on the NI-ELVIS board correctly in an efficient manner. This lab was significant because it demonstrated how a problem that was solvable with Boolean logic could be simplified down from a more complex answer into a simpler form, which allows for efficient use of resources in everything from computers to ATM machines to alarm systems.

**Introduction**

Boolean logic is defined as a form of “symbolic logic” according to the EG Lab Manual (2015). Its usage is in the implementation of every computing device in the world, from computers to alarm systems to the toaster, and thus impacts the world greatly.

With Boolean logic, there are only the two values of true and false, which are represented as the integers 1 and 0 respectively. When logic is performed on those inputs of 1s and 0s, a single output is produced.

In the case of digital circuitry, Boolean logic is represented by logic gates, perform some operation upon a number of inputs to produce a single output. These gates can be linked or “wired” together to form a combinational digital logic circuit, allowing for complex calculations to be performed.

Three of the majour gates in digital logic include the AND, OR and NOT gates. For the AND gate, it takes in any number of inputs, and produces a 1 signal if **all** of the input signals are 1s as well. If even a single input signal is a 0, it outputs a 0. The OR gate can be thought of as the complement to the AND gate, in that it also takes in any number of inputs, and outputs a 1 if **any** of the input signals are a 1. The NOT gate is an inverter, taking in only one signal, and producing the opposite of that signal, so an input of 1 would produce an output of 0.

As an example problem, the functionality for an ATM may be implemented using logic gates, with functionality including printing a statement (P), withdrawing money (W), and depositing money (D). The output of the digital logic circuit is whether or not the ATM session has a cost (C).

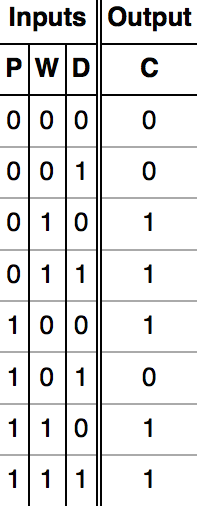
 Every possible combination of inputs and outputs can be represented in the form of a **truth table**, with the sample problem’s truth table shown in Fig. 1.

Fig. 1: Truth table

These tables produce an equation that represents when the ATM should mark the current transaction as having a cost, shown in Fig. 2.

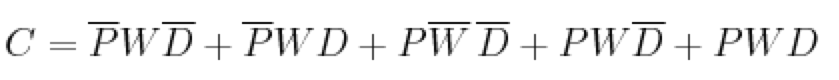


Fig. 2: Boolean expression

A K-map is a two-dimensional representation of the truth table that identifies only the conditions that are a component of the solution, removing extraneous parts of the equation, as shown in Fig. 3.

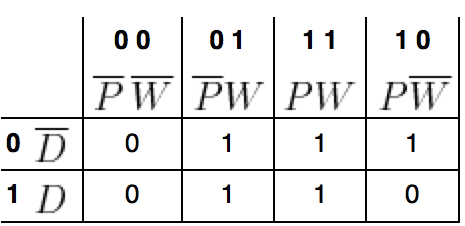


Fig. 3: K-map

The Boolean equation found from the output of a K-map is simpler than the original equation, as shown in the example in Fig. 4. Because of this simplified expression, the actual circuitry required to express this Boolean equation is also simpler. This is incredibly important, especially in embedded systems in which the space available for circuitry is limited.

Fig. 4: Simplified Boolean expression

The objectives of the lab were to design in the LabVIEW software an alarm system for a farm owner. This farm owner has two barns, a hen, and a supply of corn. The alarm should sound if the fox and then hen are in the same barn, or the hen and the corn are in the same barn. The objectives of the lab were achieved, and a simplified and functional alarm was built.

**Procedures**

The first step was to design a truth table from the given inputs of a hen (H), a fox (F), a pile of corn (C), with the output as whether the alarm goes off (A). This resulted in the truth table as shown in Fig. 5.

|  |  |  |  |
| --- | --- | --- | --- |
| **H** | **F** | **C** | **A** |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |

Fig. 5: Truth table for alarm

From this truth table, the Boolean expression in Fig. 6 was found to represent the alarm system, though it could still be simplified.



Fig. 6: Boolean equation for alarm

By using the Boolean equation from above and running it through a K-map, as shown in Fig. 7, the simplified Boolean equation could be derived.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 1 | 1 | 1 | 0 |
|  | 1 | 0 | 1 | 1 |

Fig. 7: K-map for alarm

The simplified equation, shown in Fig. 8, was then translated to digital logic through the use of logic gates.



Fig. 8: Simplified Boolean expression for alarm

This digital logic was then programmed into a LabVIEW program, as shown in Fig. 9.

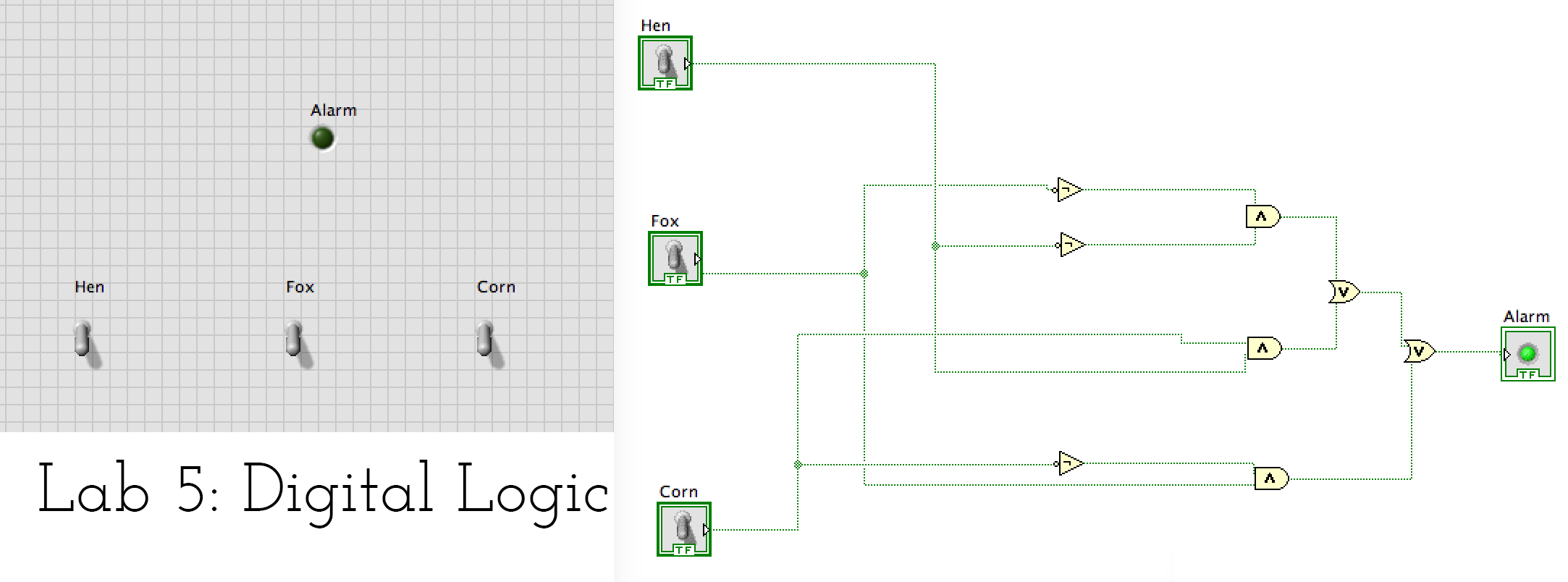


Fig. 9: Simplified digital logic of alarm system

Once the digital logic was confirmed to work, it was then translated into hardware through the use three logic gate integrated circuits (ICs), with the final circuit for the alarm system shown in Fig. 10.

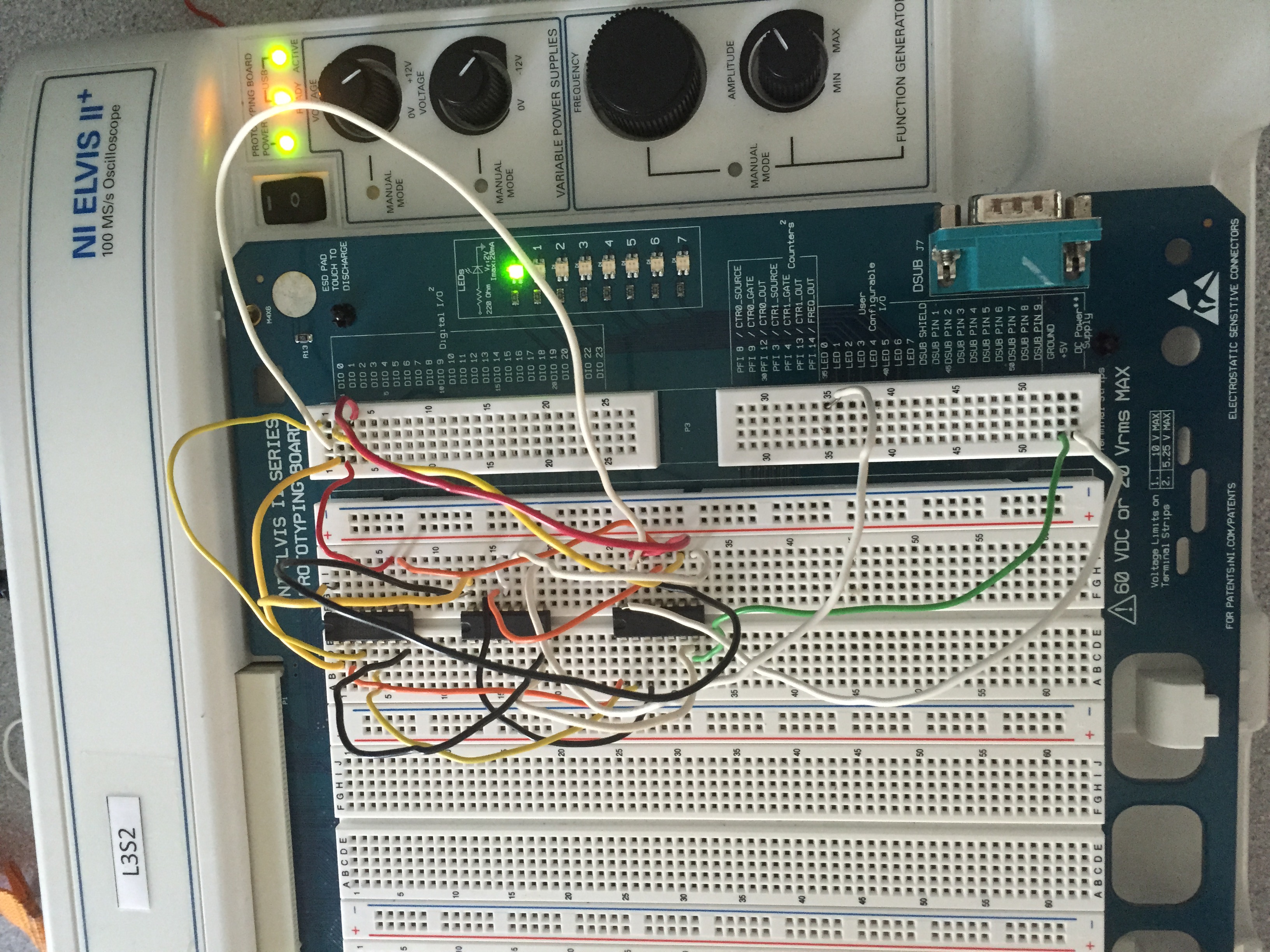


Fig. 10: Alarm system in hardware form

**Data/Observations**

When building the alarm system in hardware utilizing the AND, OR and NOT gate ICs and wiring, the NOT gate was accidentally reversed, causing current to flow through in the wrong direction, overheating the chip and causing the internals of the chip to melt. This was then fixed with the introduction of a new NOT gate IC, which allowed the project to continue, and complete successfully in the time allotted.

**Discussion/Conclusions**

The experiment was a success, with the alarm system implemented in its simplest form both digitally and physically while still covering all possible scenarios.

There is always a need for minimization of a logic design, since it directly lowers the amount of unnecessary hardware usage, which directly lowers cost of materials and cost of labour. Minimized logic design also has the advantaged of being able to be manufactured into smaller spaces, allowing for more complex logical operations in a smaller given space.

Combinational logic circuits are critical to advances in technology, since they allow for robust (deals with all possibilities) solutions to any given problem that can be expressed in Boolean equations. This allows for solutions to a very wide range of problems.

There would not be any design changes necessary if one of the barns used a bell as an alarm, and the other used a horn, since the digital logic itself remains unchanged on whether the alarm should sound, regardless of what the alarm is. This allows for **modularity** in hardware, meaning that you could take and reuse

the digital logic circuit however you wished, and attach whatever you’d like to the output of the circuit.

In this lab, the digital logic representation of the circuit in LabVIEW functioned correctly, responded quickly, and covered all possible cases. When translated into hardware, the physical circuits also functioned correctly, responded immediately, and covered all possible input cases. Thus this experiment was a success.

A way to improve the circuit in the future is by converting it from its prototype phase on the NI-ELVIS board into an IC chip, allowing the circuit to become minimized, modular, and easy to utilize. This would allow anybody to hook inputs to the chip in order to receive some output that they could transform into an alert.

**Works Cited**

NYU Polytechnic School of Engineering. 2015. “Lab 5: Digital Logic”. EG 1003 Online Lab Manual. Accessed 20 July 2015 from manual.eg.poly.edu.

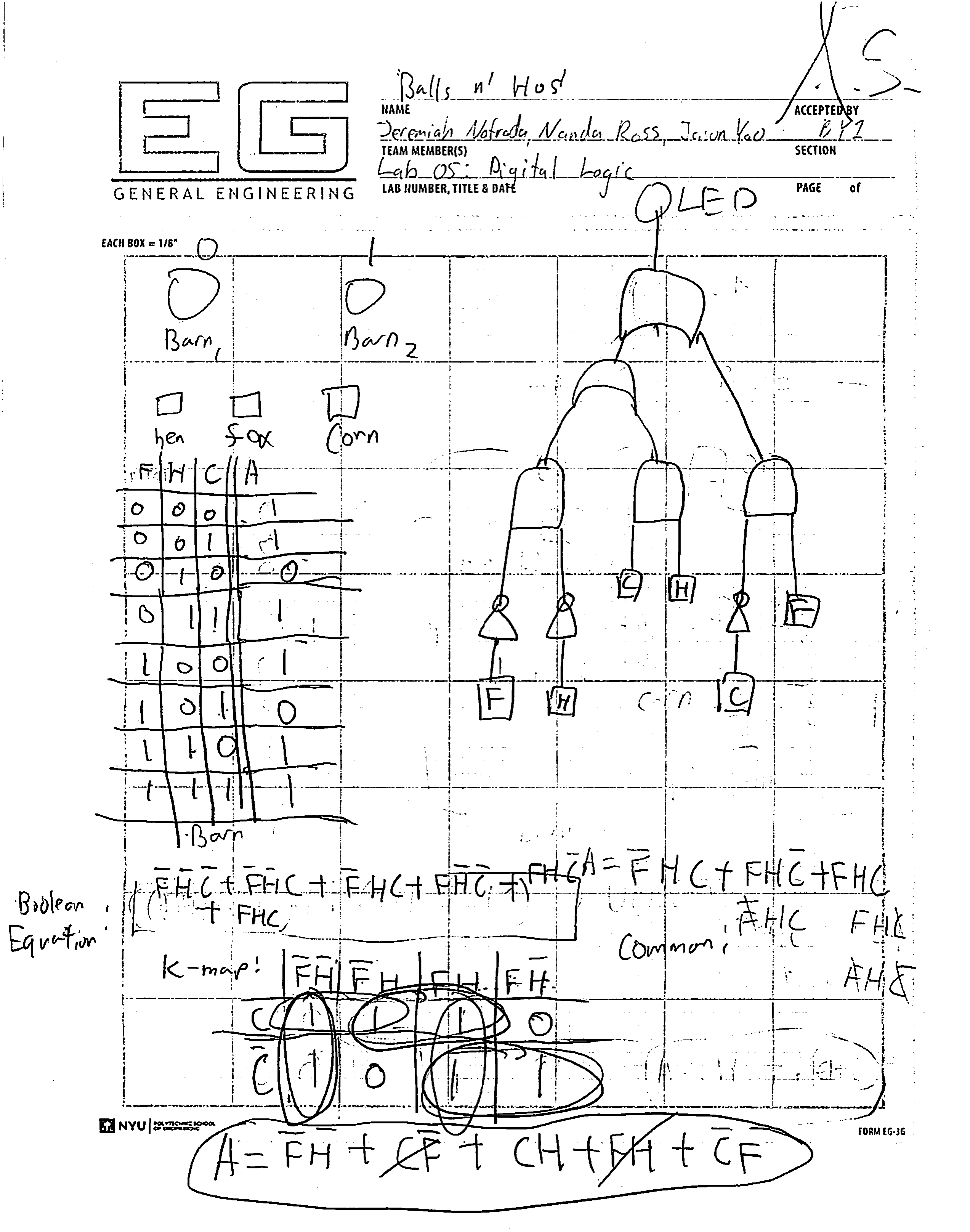


Fig. 11: Initial design sketch