PX2505 - Electronics Lab Report - Haunted Doll Factory

PX2505 - Electronics Lab Report - Haunted Doll Factory

Sidney Pauly
52104132

Declaration
I certify that this report has been written by myself, except where otherwise indicated

University of Aberdeen
Scotland
UK

https://github.com/sidney-pauly/papers

Contents

1 Introduction **1**

2 Experiment **1**

2.1 Aim 1

2.2 Theory 2

2.3 Verifying the theory 4

2.4 Schematic Diagram 4

2.4.1 Input 5

2.4.2 Logic 5

2.4.3 Output 5

2.5 Results 5

3 Conclusion **5**

1 Introduction

The electronics section is on digital logic or TTL (Transistor-transistor logic). Broadly speaking the aim digital logic is to perform logic operations to perform various practical tasks. The most simple case is to directly evaluate logical conditions electronically. An example for this could be a trafic light assembly: The light for the pedestrians should only be set to green if the request button is pressed and no car is currently on the intersection, but not if it was already green in the last 30s. Those conditions can be translated into formal logic and further into a digital circuit which then performs these operations. In principle digital logic circuits can execute any higher order algorithms, which is what computers do, However this experiment only focuses on simple logic like in the example.

The "Haunted Doll Factory" is exactly about such a application of basic logic. The doll has four inputs and four outputs and the task is to have the motors (outputs) only turn on when specific inputs are active. Using simple logic gates (AND, OR, XOR, etc.) these requirements can be implemented both as an actual circuit as well a simulation one (within the Multisim simulation tool)

A further practical goal when building digital circuits is to perform the task with as little components as possible. This is because fever components reduce the amount of failure points as well as the energy consumption of the device. To reduce the amount of components while still achieving the same logical behavior analytical methods can be used. Boolean algebra is a mathematical description of logic and can therefore be used to manipulate logical statements and to reduce them to their simplest form. This technique was used in this experiment to greatly reduce the number of required logic gates.

2 Experiment

2.1 Aim

The Lab Manual sets out the following scenario: We are tasked with designing an electronic board, that controls the behavior of (haunted) doll. The doll has four inputs (those could be buttons or switches) and four outputs (motors that can either be turned on or off). The desired logical behavior of how these inputs and outputs are related is given by (Table 1):

INPUT				OUTPUT			
S_1	S_2	S_3	S_4	M_1	M_2	M_3	M_4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	1
1	0	1	0	1	0	1	0
1	0	1	1	0	0	0	0
1	1	0	0	1	1	0	0
1	1	0	1	0	0	0	0
1	1	1	0	1	1	1	0
1	1	1	1	0	0	0	0

Table 1: Desired Behavior of the Doll (Mcphearson Page 22)

S_{1-4} specify the four inputs, while M_{1-4} specify the four outputs (motors). If a given input shows 0 its off, if it shows 1 its on, the same applies to the motors.

2.2 Theory

At first glance the behavior of the doll looks quite complex. It is not immediately clear what sort of simple logical expression could describe the behavior of the doll in its entirety. Luckily there are tools to analyses such a table and thus arrive at a simplified expression.

The first thing that could be asked is for what inputs is an individual motor turned on? This could than be translated into logical language and those logical expression could then be reduced. A simple example is Motor 2, it is on if input 1 is on, input 2 is on, input 3 is off and input 4 is off. Alternatively it is also on if input 1 is on, input 2 is on, input 3 is on and input 4 is off. For all other cases the motor is off. This can also be represented in boolean algebra:

$$M_2 = (S_1 * S_2 * \bar{S_3} * \bar{S_4}) + (S_1 * S_2 * S_3 * \bar{S_4})$$

Translated to natural language multiplication means "and", addition "or" and a bar over the respective value means "not". The nice thing about this representation is that relevant algebraic rules hold, i.e. transitivity, commutativity, etc.(Mcphearson Page 15).

Thus the above expression can be simplified to:

$$M_2 = (S_1 * S_2 * \bar{S_4})(S_3 + \bar{S_3})$$

as something OR not something $(X + \bar{X})$ is always true the expression further simplifies to:

$$M_2 = S_1 * S_2 * \bar{S_4}$$

Through the same technique similar expressions for the other motors could be derived. Then the expressions could be compared and similar operations could then be implemented with the same physical components to reduce their overall count.

However in case of the doll there is a quicker and more effective way: converting the binary input and output signals to decimal. As we are more used to decimal (and because decimal represents more information in fever digits) it is sometimes more intuitive to understand and to infer patterns. Table 2 shows such a decimal translation. The representation treats every input as a digit of a four digit binary number. The same is done for the output.

Input	Active Motor	Output
0		0
1	M_4	1
2	M_3	2
3		0
4		0
5		0
6		0
7		0
8	M_1	8
9	M_1, M_4	9
10	M_1, M_3	10
11		0
12	M_1, M_2	12
13		0
14	M_1, M_2, M_3	14
15		0

Table 2: The behavior translated to Decimal

Doing this translation immediately makes it obvious that there are just two cases:

1. The output is 0 (all motors are off)
2. The input is the same as the output (the motors belonging to the respective input are on)

The result of this realization is that the task can be rephrased: we need a circuit that is only on for the binary representations of the numbers 1, 2, 8, 9, 10, 12 and 14. The most direct (but also most inflated) boolean representation for the same looks like this:

$$O = (\bar{S_1}\bar{S_2}\bar{S_3}S_4) + (\bar{S_1}\bar{S_2}S_3\bar{S_4}) + (\bar{S_1}\bar{S_2}S_3\bar{S_4}) + (S_1\bar{S_2}S_3S_4) + (S_1\bar{S_2}S_3\bar{S_4}) + (S_1S_2\bar{S_3}\bar{S_4}) + (S_1S_2S_3\bar{S_4})$$

Through a lot of algebraic manipulation one can end up with the following expression:

$$O = S_1\bar{S_4} + \bar{S_2}S_3\bar{S_4} + \bar{S_2}\bar{S_3}S_4$$

This expression can be further simplified by using the fact that $X\bar{Y} + \bar{X}Y = X \oplus Y$, i.e. X and not Y or not X and Y is the same as only X or only Y (exclusive OR):

$$O = S_1\bar{S_4} + \bar{S_2}(S_3 \oplus S_4) \tag{1}$$

This simple expression now distinguishes the two cases laid out previously. If the expression is false, no motors should be on. If it is true the motors with that have their corresponding input on should be on as well. Logically this can be expressed as

$$M_n = S_nO$$

Note that this expression basically acts as a filter or mask on the inputs. Because of this the expression will from on be referred to as the logic filter.

2.3 Verifying the theory

Before spending a lot of time on building circuits it useful to know that the discoverd logic expression does indeed recover the desired logical behavior. The following script runs through all different input combinations and applies the logical expression to them. The result is then printed out.

```

import pandas as pd

result = [0] * 16

for i in range(0, 16):

    s_1 = (8 & i) > 0
    s_2 = (4 & i) > 0
    s_3 = (2 & i) > 0
    s_4 = (1 & i) > 0

    O = (s_1 and not s_4) or (not s_2 and (s_3 ^ s_4))

    result[i] = [s_1, s_2, s_3, s_4, O and s_1, O and s_2, O and s_3, O and s_4]

print(pd.DataFrame(result))

```

The output of this script matches table 1 exactly, confirming that the simplification process worked

2.4 Schematic Diagram

The first step is to draw a schematic diagram of the resulting circuit. The following schematic diagram was drawn and then simulated with multisim:

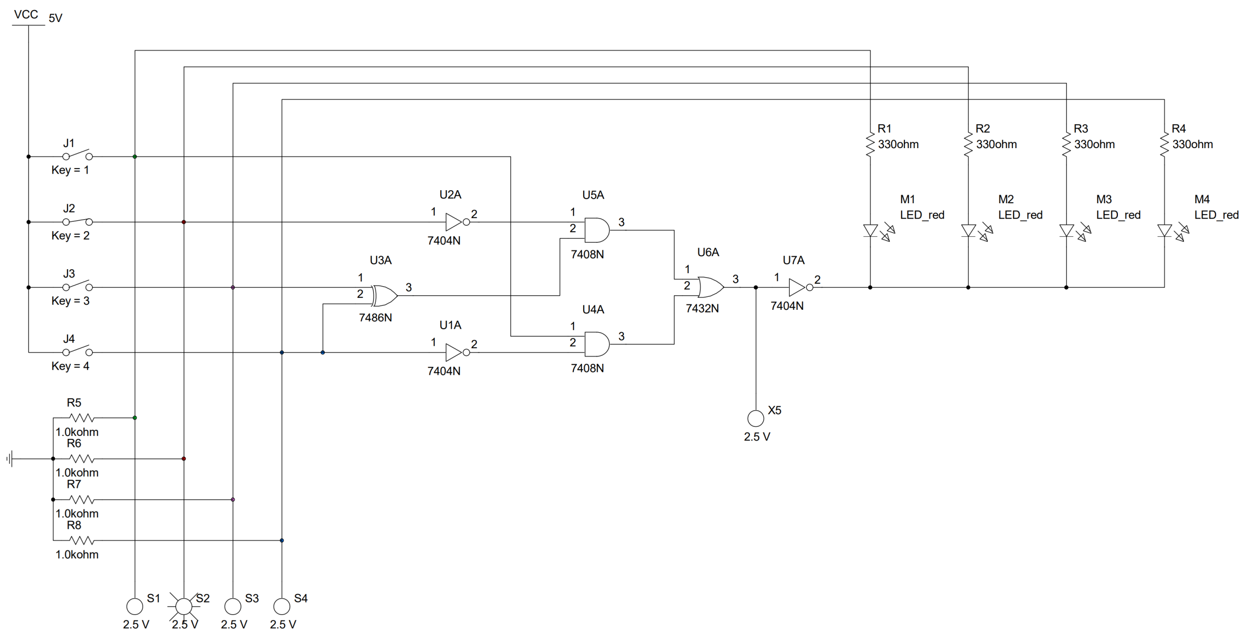

Figure 1: Schematic diagram of the "Haunted Doll" circuit

The resulting circuit consists of three main sections. Each section has some additional features and optimizations.

2.4.1 Input

The input section consists of four switches. Each of the switches corresponds to one of the inputs S_{1-4}. In the diagram they are labeled with J1-J4. If the input switches are closed the corresponding wire will be at VCC voltage ($5V$). This voltage (also referred to as HIGH) corresponds to an input value of 1. A value of 0 is represented by $0V$ (also called GND). Note that the input wires are always connected to GND through a $1k\Omega$ resistor. This connection ensures that when the input switches are open that the input wire is at a valid logic value of GND. If there would be no connection in that case the corresponding input would have an undefined value, which could lead to undefined behavior. Connecting the input through a $1k\Omega$ resistor ensures that in case the swtich is closed the input will be at VCC voltage, as the wire without resistance "overpowers" the connection with resistor.

In the schematic additional logic probe indicator lights are attached to the input cables (S1-S4). Their purpose is to quickly and easily determine the state of the given input.

2.4.2 Logic

The logic section takes in the four input signals and proccesses them. The implemented logic corresponds exactly to the logic filter O (equation 1). The logic section ends at the X5 logic probe indicator light. This light is there to quickly determine the value of O (the logic filter value). The subsequent inverter (label U7A) is not considered part of that section. It belongs to the output section and will be explained subsequently.

To implement equation 1 the corresponding 74xx series TTL logic components are used:

1. 7404N Inverter (NOT)
2. 7486N XOR
3. 7408N AND
4. 7432N OR

The layout corresponds to the equation 1:1. This can be seen easiest going backwards from the OR gate (label U6A). It's lower input corresponds to the left component of equation 1 and it's upper input to the right component. Consequently the lower AND gate (label U4A) has S_1 and $\bar{S_4}$ as inputs (S_4 gets inverted through inverter U1A). Equally the upper AND gate (label U5A) is connected to to $\bar{S_1}$ (inverter U2A) and the exclusive or gate (label U3A) that combines S_3 and S_4.

2.4.3 Output

2.5 Results

3 Conclusion

Works Cited

Mcphearson, Dr. Ross. PX2505 Practical Optics & Electronics Electronics Manual.