CSSE2010 Assignment 1

Jay

April 14, 2022

1 Design

The final digit in my student number is 2. So, given the combination 10010 the correct order of input for me is $YYYXX$, for two inputs X and Y. Shown visually in *Figure 1.*

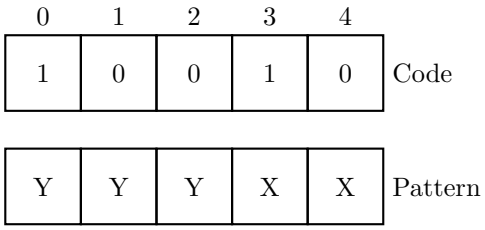


Figure 1: **Pattern Diagram**

The given combination can then be converted into a state diagram as shown in *Figure 2.* This diagram conveys the correct sequence of input $Y\overline{Y}Y\overline{X}\overline{X}$ to unlock the digital lock at S_5, the **Unlocked** state. It also shows that the device will return to S_0, the **Reset** state, when it is first initialised, when the wrong combination is entered, or when the reset button is pressed. As per the requirements given in the *Design Task.* The other S_n states are in sequential order of correct inputs to decode the lock. The three outputs given D_2, D_1, and D_0 are arbitrary in *Figure 2* below.

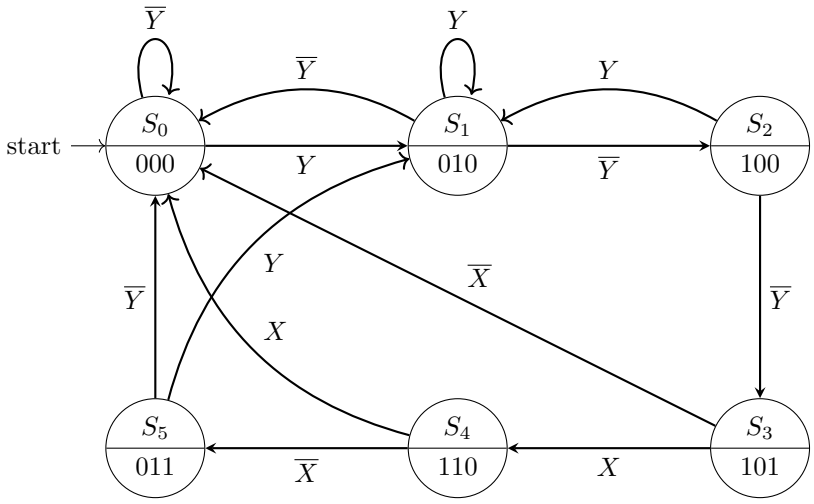


Figure 2: **State Diagram of the Unlock Sequence**

As seen from *Figure 2*, there are six valid states the lock can be in. Given the formula 2^m for flip-flops' states, the device will need a minimum of three flip-flops as 2^2 is insufficient. Also from this figure, it is now trivial to draw a state table of the connections from one state to another and the outputs associated with them. *Table 1* such a table.

Table 1: State Table 1

State	Input				Output		
	$\overline{X}\overline{Y}$	$\overline{X}Y$	$X\overline{Y}$	XY	L_2	L_1	L_0
RESET	RESET	S_1	RESET	S_1	0	0	0
S_1	S_2	S_1	S_2	S_1	0	0	0
S_2	S_3	S_1	S_3	S_1	0	0	0
S_3	RESET	RESET	S_4	S_4	0	0	0
S_4	UNLOCK	UNLOCK	RESET	RESET	0	1	0
UNLOCK	RESET	S_1	RESET	S_1	1	0	1

From *Table 1*, it is clear that though there are four possible inputs at any given time, there will only every be two outcomes, dependent on whether the X or the Y input is being checked. Given the circuit requires three flip-flops, which is $2^3 = 8$ states, there are two states which are invalid. These *'undefined'* states will be labelled U_1 and U_0 respectively. If either state is reached, the device will return to the **Reset** state S_0. With these details now considered, two encodings have been developed; *Grey Encoding* and *1061 Encoding*

The *Grey Encoding* only allows for one bit to change between states. Referring back to *Figure 2*, following the steps of the state diagram the following grey encoding can be derived.

$$000 \rightarrow 001 \rightarrow 101 \rightarrow 111 \rightarrow 011 \rightarrow 010$$

Starting from the **RESET** state at 000 to the **UNLOCK** state at 010, each state is reached by changing a single bit. *Table 2* is the 1-dimensional state table that is the result of the given 1-bit-flip transition from one state to the next.

The concept behind the *1061 Encoding* goes against the conventions given by *CSSE2010*. Instead of stating at **False** states and moving down to **True** in a truth table, this begins at **True** and moves down to **False** states. This is to test *why* the conventions differ between Computer Science and Mathematics. Unlike *Grey Encoding*, this encoding derives the following sequence from its state table, *Table 3*.

$$111 \rightarrow 110 \rightarrow 101 \rightarrow 100 \rightarrow 011 \rightarrow 010$$

This sequences starts at **RESET** at 000 to **UNLOCK** at 010. The structure of the *1061 Encoding* necessitates a simplification encoding for both the inputs and the outputs. This however was not achieved, as will be discussed. It quickly becomes clear why the two disciplines differ in their conventions.

Table 2: Grey Code

Current State			Input		Next State			Output		
Q_2	Q_1	Q_0	X	Y	D_2	D_1	D_0	L_2	L_1	L_0
0	0	0	X	0	0	0	0	0	0	0
0	0	0	X	1	0	0	1	0	0	0
0	0	1	X	0	1	0	1	0	0	0
0	0	1	X	1	0	0	1	0	0	0
0	1	0	X	0	0	0	0	1	0	1
0	1	0	X	1	0	0	1	1	0	1
0	1	1	0	Y	0	1	0	0	1	0
0	1	1	1	Y	0	0	0	0	1	0
1	0	0	X	Y	0	0	0	0	0	0
1	0	0	X	Y	0	0	0	0	0	0
1	0	1	X	0	1	1	1	0	0	0
1	0	1	X	1	0	0	1	0	0	0
1	1	0	X	Y	0	0	0	0	0	0
1	1	0	X	Y	0	0	0	0	0	0
1	1	1	0	Y	0	0	0	0	0	0
1	1	1	1	Y	0	1	1	0	0	0

From this state table, it is now possible to utilise Boolean algebra to derive the most efficient method of wiring the inputs and outputs of the circuit.

Grey Code Input Boolean Algebra

$$D_2 = \overline{Y}(\overline{Q}_2 \cdot \overline{Q}_1 \cdot \overline{Q}_0) + \overline{X}(Q_2 \cdot \overline{Q}_1 \cdot Q_0)$$

$$D_1 = \overline{X}(\overline{Q}_2 \cdot Q_1 \cdot Q_0) + \overline{Y}(Q_2 \cdot \overline{Q}_1 \cdot Q_0) + X(Q_2 \cdot Q_1 \cdot Q_0)$$

$$D_0 = Y(\overline{Q}_2 \cdot \overline{Q}_1 \cdot \overline{Q}_0) + \overline{Y}(\overline{Q}_2 \cdot \overline{Q}_1 \cdot Q_0) + Y(\overline{Q}_2 \cdot \overline{Q}_1 \cdot Q_0) + \overline{Y}(Q_2 \cdot \overline{Q}_1 \cdot Q_0) + Y(Q_2 \cdot \overline{Q}_1 \cdot Q_0) + X(Q_2 \cdot Q_1 \cdot Q_0)$$

Grey Code Output Boolean Algebra

$$\begin{aligned} L_2 &= Y(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) + \overline{Y}(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) \\ &= \overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0 \end{aligned}$$

$$\begin{aligned} L_1 &= X(\overline{Q}_2 \cdot Q_1 \cdot Q_0) + \overline{X}(\overline{Q}_2 \cdot Q_1 \cdot Q_0) \\ &= \overline{Q}_2 \cdot Q_1 \cdot Q_0 \end{aligned}$$

$$\begin{aligned} L_0 &= Y(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) + \overline{Y}(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) \\ &= \overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0 \end{aligned}$$

Table 3: 1061 Encoding State Table

Current State			Input		Next State			Output		
Q_2	Q_1	Q_0	X	Y	D_2	D_1	D_0	L_2	L_1	L_0
1	1	1	X	1	1	1	0	0	0	0
1	1	1	X	0	1	1	1	0	0	0
1	1	0	X	1	1	1	1	0	0	0
1	1	0	X	0	1	0	1	0	0	0
1	0	1	X	1	1	1	1	0	0	0
1	0	1	X	0	1	0	0	0	0	0
1	0	0	1	Y	0	1	1	0	0	0
1	0	0	0	Y	1	1	1	0	0	0
0	1	1	1	Y	1	1	1	0	1	0
0	1	1	0	Y	0	1	0	0	1	0
0	1	0	X	1	1	1	0	1	0	1
0	1	0	X	0	1	1	1	1	0	1
0	0	1	X	Y	1	1	1	0	0	0
0	0	0	X	Y	1	1	1	0	0	0

1061 Encoding Input Boolean Algebra

$$\begin{aligned} D_2 = {} & Y(Q_2 \cdot Q_1 \cdot Q_0) + \overline{Y}(Q_2 \cdot Q_1 \cdot Q_0) + Y(Q_2 \cdot Q_1 \cdot \overline{Q}_0) + \overline{Y}(Q_2 \cdot Q_1 \cdot Q_0) + Y(Q_2 \cdot \overline{Q}_1 \cdot Q_0) \\ & + Y(Q_2 \cdot \overline{Q}_1 \cdot Q_0) + \overline{X}(Q_2 \cdot \overline{Q}_1 \cdot \overline{Q}_0 + X(\overline{Q}_2 \cdot Q_1 \cdot Q_0) + Y(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) + \overline{Y}(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) \end{aligned}$$

$$\begin{aligned} D_1 = {} & Y(Q_2 \cdot Q_1 \cdot Q_0) + \overline{Y}(Q_2 \cdot Q_1 \cdot Q_0) + Y(Q_2 \cdot Q_1 \cdot \overline{Q}_0) + Y(Q_2 \cdot \overline{Q}_1 \cdot Q_0) + X(Q_2 \cdot \overline{Q}_1 \cdot \overline{Q}_0) \\ & + Y(Q_2 \cdot \overline{Q}_1 \cdot \overline{Q}_0) + X(\overline{Q}_2 \cdot Q_1 \cdot Q_0) + \overline{X}(\overline{Q}_2 \cdot Q_1 \cdot Q_0) + Y(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) + \overline{X}(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) \end{aligned}$$

$$\begin{aligned} D_0 = {} & \overline{Y}(Q_2 \cdot Q_1 \cdot Q_0) + Y(Q_2 \cdot Q_1 \cdot \overline{Q}_0) + \overline{Y}(Q_2 \cdot Q_1 \cdot \overline{Q}_0) + Y(Q_2 \cdot \overline{Q}_1 \cdot Q_0) \\ & + X(Q_2 \cdot \overline{Q}_1 \cdot \overline{Q}_0) + \overline{X}(Q_2 \cdot \overline{Q}_1 \cdot \overline{Q}_0) + X(\overline{Q}_2 \cdot Q_1 \cdot Q_0) + \overline{Y}(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) \end{aligned}$$

Following on, the simplification of the output is as follows.

1061 Encoding Output Boolean Algebra

$$\begin{aligned} L_2 &= Y(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) + \overline{Y}(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) \\ &= \overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0 \end{aligned}$$

$$\begin{aligned} L_1 &= X(\overline{Q}_2 \cdot Q_1 \cdot Q_0) + \overline{X}(\overline{Q}_2 \cdot Q_1 \cdot Q_0) \\ &= \overline{Q}_2 \cdot Q_1 \cdot Q_0 \end{aligned}$$

$$\begin{aligned} L_0 &= Y(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) + \overline{Y}(\overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0) \\ &= \overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0 \end{aligned}$$

With both encodings defined, the circuit diagrams below are made. Discussion on considerations will be made after.

2 Simulation

Figure 3: Grey Encoding

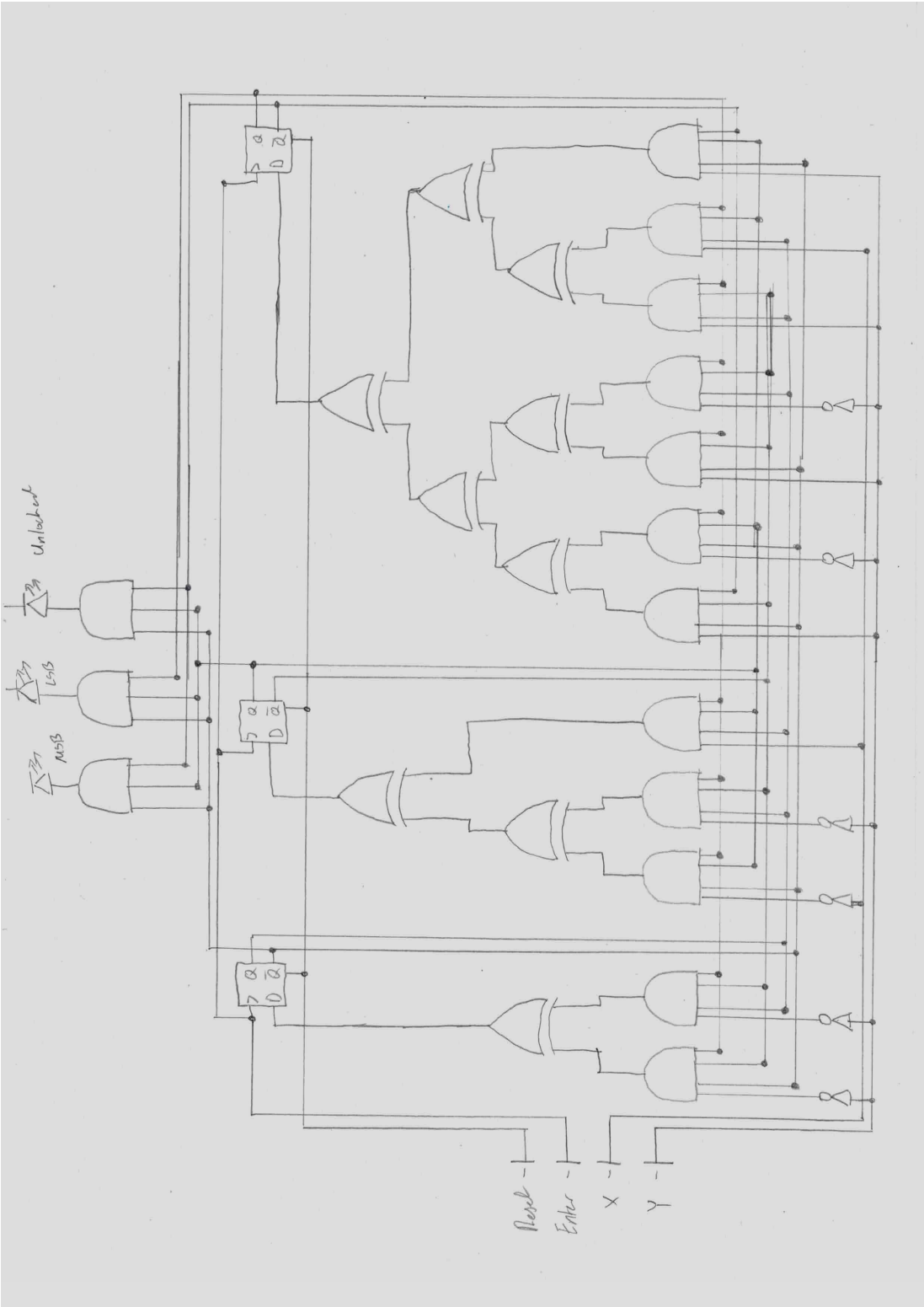
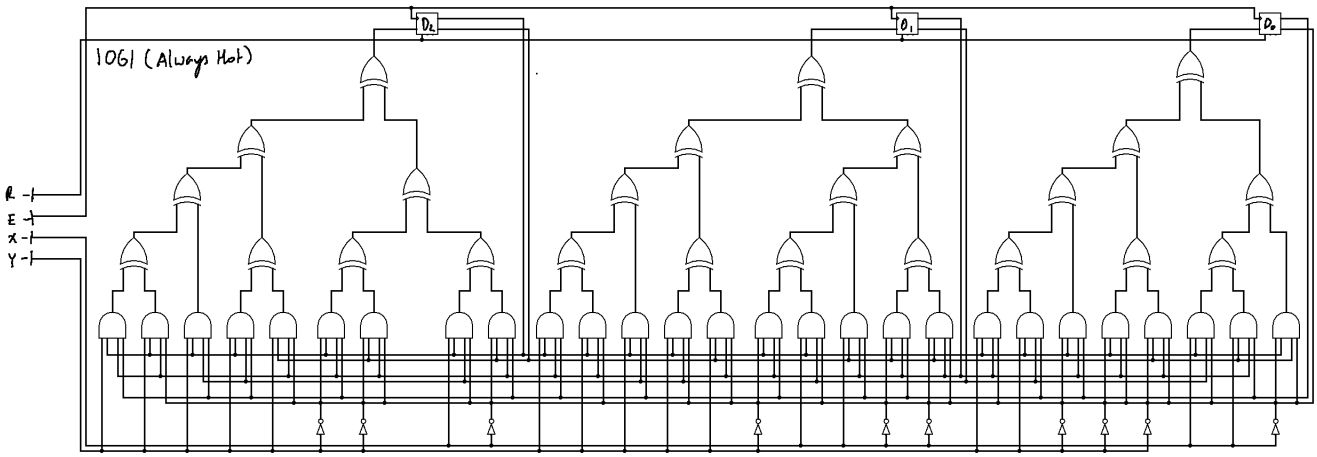


Figure 4: 1061 (Always Hot) Encoding



3 Questions

Question 1

The *1061 (Always Hot)* encoding was not made with the intention of being a serious encoding. It was developed to provide personal insight to why conventions differ between the disciplines of Computer Science and Mathematics. The partially completed circuit in *Figure 4* conveys visually *one* of the main problems with using a *always hot* method. There are 51 total gates shown in *Figure 4*, when compared to *Figure 5*, for *Grey* encoding having 22 total gates for both inputs and outputs, the choice of which to simulate becomes clear.

It was an interesting *but painful* experiment. A critical issue was found regarding undefined states for the *1061 (Always Hot)* encoding. If in one of the two undefined states, the circuit would get stuck in 000 for Q_2, Q_1, Q_0, which itself was an undefined state. Fixing this error would be a waste of time and effort which was instead put into the diagram presentation and labelling. If it were not for consistent errors in my Boolean algebra, which prevented accurate simplification of the encodings, I would have been satisfied. If it were not for consistent errors in my Boolean algebra, which prevented accurate simplification of the encodings, I would have been fully satisfied with my findings. The *Grey* encoding was simplistic to design and implement on the other-hand. Without my Boolean algebra errors, the circuit could have been simplified to roughly 16 total gates.

Question 2

As mentioned, the *1061 (Always Hot)* encoding would get stuck in 000 for Q_2, Q_1, Q_0, which itself was an undefined state. No further discussion of this encoding will be made due to this. The *Grey* encoding however, proved to be successful. It took four iterations, starting from a design based on the simplification of inputs. The final iteration did not include simplification of inputs as it was not required by the spec and my simplifications lead to critical errors. This iteration was tested to ensure that if the circuit *was* in an undefined state, it would return to its reset state. It is in the functional *Logisim* circuit only. This came from testing edge cases. The final AND gate was so the final step will continue as mentioned in the spec. Attempts at brute-forcing the last three steps were made, but were not successful. Overall, I believe this circuit to be a good example of the *Grey* encoding and fully functional to the details given in the spec.