An Overview of Computer Architecture adventure into the imaginary world of Harry Potter Part 2

Anas Shrinah

November 24, 2020

In our last adventure, we successfully unlocked the door of Vault 713 in Gringotts Wizarding Bank. In the vault, there is a ruby-red stone called the Philosopher's Stone, fig. 1, a legendary substance with magical powers. It is believed the Stone can transform any metal into gold. It also produces the Elixir of Life, which will make the drinker immortal! This stone is sought after by Lord Voldemort (the most powerful and dangerous Dark Wizard of all time). Sorry, I meant You-Know-Who. I almost forgot that we are not supposed to say his name.

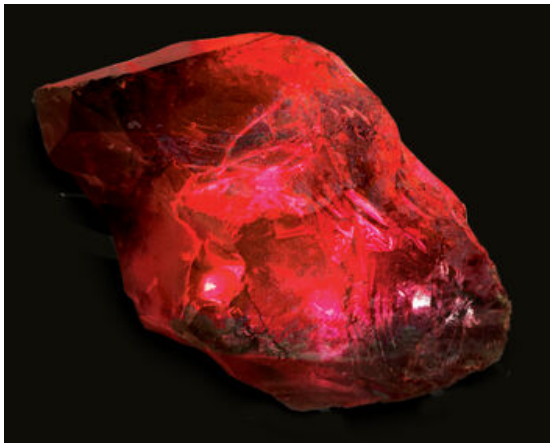


Figure 1: Philosopher's Stone [1].

Our mission is to protect the Stone by changing the logical circuit of the lock. Last time we solved the puzzle of the new secret code. Of course, we can use any 4-bit pattern as the new code, but the new secret code from the riddle in Vault 713 is magical. It is said it is extra hard; it even might be impossible, for You-Know-Who to crack it. Now it is time to design a circuit that will unlock the door upon receiving your new secret code.

1 State Transition Diagram

The specification of our lock is simple. The door should be unlocked after receiving the correct input sequence. Every time a wrong input is received, the circuit goes back to the initial state. From there, it will be ready to receive the next sequence of inputs from the beginning.

The first step is to draw a state transition diagram that captures the specification of the new lock.

1. How many states does our FSM have? 4
2. How many bits do we need to represent all states? 2
3. What does the output represent? lock or unlock

4. How many outgoing transitions does each state have? 2
5. Why does every state have an equal number of outgoing transitions? restart or go on
6. How many transitions does our FSM have? 8
7. How many incoming transitions does the initial state have? 5
8. Why does the initial state have at least one incoming transition from each other state? when output is 0, it must point to the initail one
9. Why does the last state have no self-lopping transition? fail or success

With the state transition diagram, now we can easily make the state transition table by recording all transitions.

2 State Transition Table

The state transition table is a tabular representation of the state transition diagram. We need it to make the next steps easier. Remember, the rows of the state transition table are the transitions of the state transition diagram. The columns of the state transition table are the binary coding of the current state, the input, the next state and the output. How many rows does the new lock state transition table have?

Good job preparing your state transition table! Now we can find the logical formulas of the next state bits and the output as functions of the current state bits and the input. Use the state transition table to find the Disjunctive Normal Form (DNF) formulas of the output and the next state bits. Next, we will use Karnaugh maps to find optimised formulas.

3 Karnaugh Maps

The Karnaugh map, or K-map for short, is a great method to simplify Boolean algebra expressions. Use K-maps to find simplified formulas for the output and the next state bits. With your team discuss and answer the following questions.

1. How many K-maps do we need to simplify the formulas of the output and the next state bits?
2. How many variables are there in each one of our K-maps?
3. Compare the formulas from the K-map and the ones derived directly from the state transition table.

4 Circuit Implementation in Logisim

Well done for finding the formulas for the output and the next state bits. As a team, implement these formulas in Logisim. You can use any logic gate with as many inputs as Logisim provides. Download and use the lock circuit template `Lock_FSM_template.circ`.

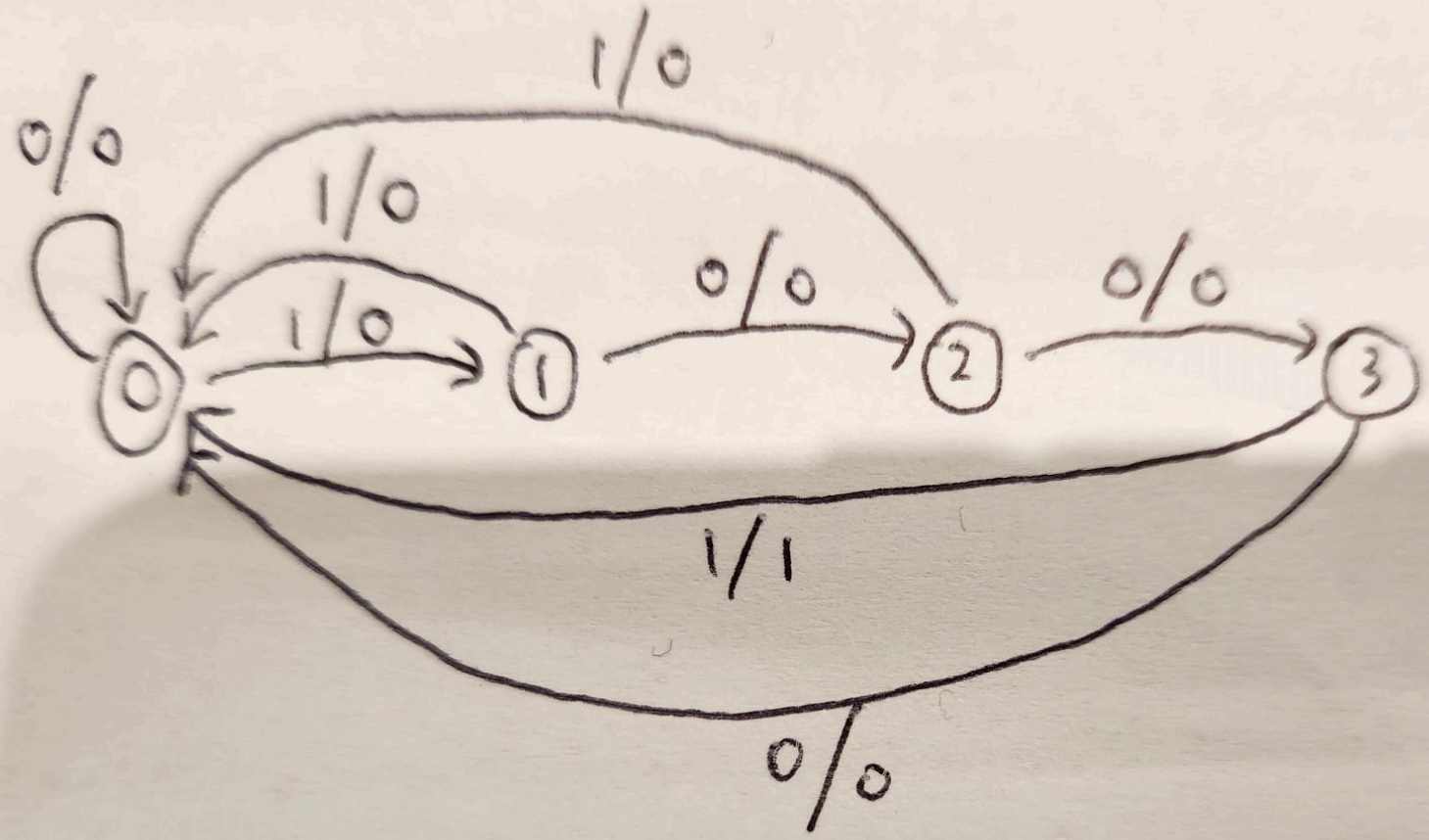
Discuss with your team at least two input sequences to test if your lock circuit works properly.

Good job! As an extra task, try to reimplement your design using two-input NAND gates only. Re-test your design to ensure the behaviour of your design did not change in the new implementation.

This brings us to the conclusion of our adventure. Hopefully you had fun while advancing your computer architecture skills.

References

[1] Harry Potter Wiki. Philosopher's stone. https://harrypotter.fandom.com/wiki/Philosopher%27s_Stone, 2020. Online; accessed 23-November-2020.


1/0
0/0
1/0
0/0
0/0
1/0
0
1
2
3
1/1
0/0

S_1	S_0	I	O	S_1'	S_0'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	0	0