Lab 3 – State Machine Design and Simulation

Introduction

The objectives of these sessions are to:
− Analyze the partitioning and mapping of a complex system its sub components
− Learn the design, simulate, and implement a State Machine
− Experience a full design cycle of a system similar to an ECTE250 project

3.1 System Functionalities and Partitioning

The system to be designed included the following input and outputs:
− One active-high switch (input)
− UDP messages from the Internet (input and output)
− One LED (output)
− LCD Display (output)

The messages displayed on the LCD and sent via UDP are identical. Therefore, we can summarize that such a system present two inputs and two outputs.

It is reasonable to use Arduino to handle input and output UDP messages, and the LCD display. Switch and LED can be directly interfaced to the State Machine. Moreover, you can add additional LEDs (one per flip flop) to display the current state of the of the State Machine. In this case use different colors for the output LED (e.g. red) and for the LEDs indicating the current state (e.g. green).

The system should provide the following functionalities:

− At startup (or at reset) the LED is off and nothing is displayed on the LCD (assume the switch is OFF).

− When the switch is ON, the LED goes ON. If no message from the internet is received while the LED is ON, and if the switch goes to OFF, also the LED goes OFF (going back to a state identical to the startup state).

− While the LED is ON, if the system receives the message ‘YES’ from the internet, the LED stays ON, and the LCD displays ‘HELLO’.

− While displaying ‘HELLO’ on the LCD, and the LED is ON, any change on the switch is ignored.

− While displaying ‘HELLO’ on the LCD, and the LED is ON, if the system receives the message ‘NO’ from the internet the display is cleared.

− When the LCD display changes from clear to ‘HELLO’, the message hello is also transmitted over the internet (no message has to be sent when the display changes from ‘HELLO’ to clear).

The above description is already “Moore” State Machine friendly.
Use Arduino to receive UDP messages from the internet, and associate one digital output line with the received message (e.g. output a high voltage after receiving ‘YES’, and a low after receiving ‘NO’).

The State Machine has 2 inputs, one coming from the active high switch, and one coming from Arduino. The State Machine has two outputs, one going to Arduino (to command the display of ‘HELLO’), and one driving the LED.

Arduino is interfaced to the Internet, it drives the LCD display, and it has two line (one input and one output) towards the State Machine.

5.2 State Machine Design, Implement, and Test

As describe in 5.1 the State Machine has 2 inputs and 2 outputs. To achieve the functionalities described above, 3 states are likely to be sufficient.

− Design a state diagram (on paper) for the state machine, following the guidelines presented in the related lecture.

− If the state diagram appears reasonable and all inputs, outputs, states, transitions has been defined, implement the state diagram on Boole.

− Simulate the State Machine in Boole (use Results → Advanced Interactive Simulation, or use the Advanced Batch Simulation).

− Test your State Machine, implemented on Boole at state diagram level, against any possible scenario. If the behavior is not as expected/described in 5.1 amend your design. If all scenarios work as expected, get the State Machine equations and circuit from View Circuit → With D flip-flops.

Task: Implement your state machine (your project, not this example) using Boole dusto and then implement it using the logical gates on Tinker cad.