Finite State Machines

In this exercise we specify a finite state machine (FSM) to be implemented later on an FPGA.

Hand-in instructions: Prepare a small report with your solutions (detailed). Submit your report as a PDF through the lecture moodle per the moodle submission deadline.

System Description

The goal is to model an electronic lock system, as shown in Figure 1, which might be used in an access control system or a safe with an electronically controlled door.

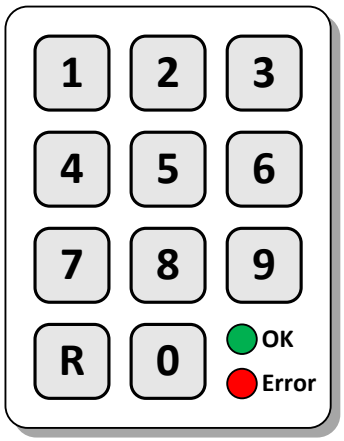

Figure 1: Electronic Lock System

The input of the FSM are two signals, the *KeyValid* signal, which is asserted (='1') when a key is pressed, and the *Key* signal, which reports the value of one of the pressed numerical keys. Once the key is released, *KeyValid* is de-asserted (='0'). The possible values of these two signals are:

- *KeyValid* $\in \{0, 1\}$ (key pressed)
- *Key* $\in \{0, 1, 2, 3\}$ (key value)

The interface electronics are designed in a way that if two numerical keys are pressed at the same time, *KeyValid* is not asserted (i.e., both keys are ignored). If *KeyValid* is asserted, *Key* always has a defined value $\in \{0, 1, 2, 3\}$.

The outputs of the FSM are two signals controlling a red and a green LED:

- *GLED* $\in \{0, 1\}$ (green LED: OK & door open)
- *RLED* $\in \{0, 1\}$ (red LED: Door closed or wrong key)

The signal for the green LED also controls the door opening mechanism of the system.

Basic Lock (Single Cycle Open)

Initially, the door is closed (*RLED* asserted). Once any key is pressed, both LEDs turn off. After three keys have been pressed, the door is either open (*GLED* asserted) if the code was correct or closed (*RLED* asserted) if the code was wrong. Once the door is open (for a single clock cycle), it closes automatically (transitions into the closed state in which the *RLED* asserted).

Our lock has a 3-digit code: "0,2,1" that must be entered in the correct order. Make sure to check after each key press, that the key has been released again, before waiting for the next key press. This mechanism in practice avoids multiple detections of the same key, due to it being held down for a long time.

Task 1: State Diagram of Basic Lock

Draw the state machine diagram that implements the above-described lock.

Lock with Extended (Multi-Cycle) Opening

The lock defined in the previous Task has the problem that it remains open only for a single clock cycle. On the FPGA, our FSM is clocked with a clock of $125\,\text{MHz}$, which renders the opening time very short if we just use a single clock cycle.

Modify your FSM such that the lock stays open for approximately 2 seconds (green light) and then it closes (red light). Any new key input can be ignored during the 2 seconds where the lock is open. The red light stays on until the first key is pressed to open it again (the light is not shown while a new password is being entered). To this end, you are given two new FSM outputs to help control the time the lock stays open:

- *CountEn* $\in \{0, 1\}$ (counter enable)
- *CountClear* $\in \{0, 1\}$ (counter clear)

These outputs control an up-counter, which increases its value by one in every cycle whenever *CountEn* is asserted. The *CountClear* signal allows the counter to be reset to zero.

The current value of the up-counter is given by the following FSM input:

- *Count* $\in \{0, 1, 2, \dots\}$ (counter value)

This input is used to control when the FSM leaves the open state.

Task 2: State Diagram of Lock with Extended Opening

Draw the state machine that implements the lock described above.