Tyler Swenson

EE457

Lab 6 Report

3/10/25

EE457 Sliding Sign

# Introduction

In this lab we will be designing a sliding sign that will display EE457 as well as a number corresponding to the current state the sign is in. Depending on whether the switch is up or down, the EE457 will then slide left or right each second. This will occur until either a reset button is pressed, or the hold button is pressed to keep the current position of the EE457. This will continue while the design is on.

# Theory of Operation

## Requirements

1. The design will reset when the reset signal is low.
2. Upon resetting, the design will default to State\_0 State.
3. If the directional signal is high, then the state machine will move forward, otherwise it will move backward.
4. When the hold signal is low, the state machine will halt until hold signal is high.
5. The state machine will tick at a rate of once per second, or another time specified.
6. Signals from switches and buttons will be synchronized
   1. SW(0) will be synchronized to the direction signal.
   2. KEY(0) will be synchronized to the reset signal.
   3. KEY(1) will be synchronized to the hold signal.

## Description of the Design

A diagram of a machine

AI-generated content may be incorrect.

Figure 1: Proposed block diagram from lab manual

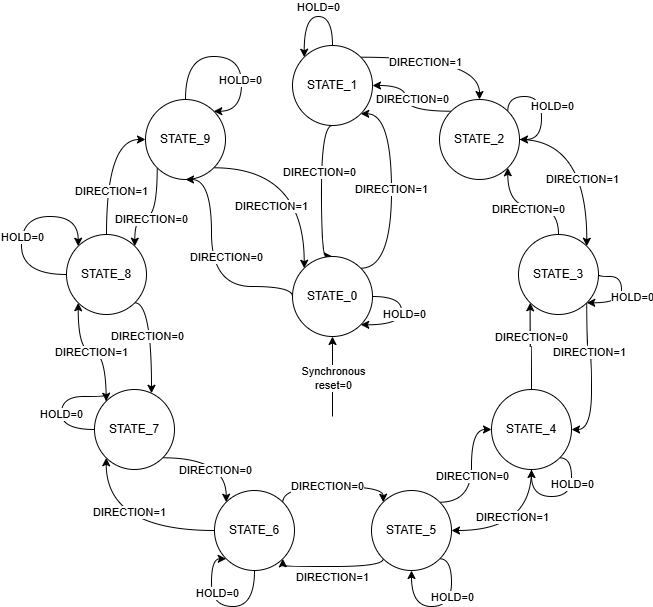


Figure 2: Sliding Sign State Machine

The user will be able to provide inputs from KEY(0) for the reset signal, KEY(1) for the hold signal, and SW(0) for the direction signal. The design will move forward through the states when SW(0) is high, and backward through the states when SW(0) is low. The states through the forward process will have 5 seven segment displays show EE457, and “slide” each signal to the right as the states progress. The sign will cause a segment to be blank when any part of EE457 does not occupy it, until a state where each segment is blank. The sign will then start to shift each character back in from the leftmost display until it is back in the starting state of EE457 being displayed. Another seven-segment display will then display which state is currently being shown using a number 1-9. This will continue until the reset signal is held low, or the hold signal is held low. Additionally, the states will start to move in reverse if the SW(0) is switched to low. The reset signal will cause the state machine to default to STATE\_0, or display EE457 1, until the reset signal returns to a high value. The hold signal will cause the state machine to stay at its current state until the signal returns to a high value. The states themselves will transition at a default tick value of 1 tick per second, or a variable rate that can be changed. This will have to be determined from the 50 MHz clock that is provided.

# Verification

## Test Plan

To test the design, a testbench will test the various uses of the sliding sign such as: the forward direction, reverse direction, resetting the state machine, and holding the current state.

To test the forward direction, the test bench will go through at least the number of states in the state machine plus a couple extra to ensure it properly cycles:

-- Step through states in the forward direction

sw(0) <= '1';

for i in 0 to (number\_of\_states + 2) loop

-- wait for slow\_clk\_wait;

wait for (clk\_per \* slow\_clk\_counts\_per\_tick);

end loop;

The hold button will then be tested to ensure that the state machine remains on the current state when held:

-- Hold test

key(1) <= '0';

wait for (clk\_per \* slow\_clk\_counts\_per\_tick \* 4);

key(1) <= '1';

The reverse direction will then be tested by flipping the switch signal to ensure that the states work properly in that cycle plus some extra:

-- reverse direction

sw(0) <= '0';

for i in 0 to (number\_of\_states + 2) loop

-- wait for slow\_clk\_wait;

wait for (clk\_per \* slow\_clk\_counts\_per\_tick);

end loop;

sw(0) <= '1';

The reset will then be tested to ensure that the synchronization of the reset signal from the button press works correctly:

-- reset test

wait for 100 ns;

key(0) <= '0';

wait for 50 ns;

key(0) <= '1';

Then finally a few more states will be transitioned through to ensure the state machine cycles properly after resetting.

-- more state transitions after reset

for i in 0 to 5 loop

-- wait for slow\_clk\_wait;

wait for (clk\_per \* slow\_clk\_counts\_per\_tick);

end loop;

## Test Bench

A black screen with green lines

AI-generated content may be incorrect.

Figure 3: Forward direction moving through the states and displaying HEX

A screen shot of a computer screen

AI-generated content may be incorrect.

Figure 4: Hold to maintain current state

A screen shot of a computer

AI-generated content may be incorrect.

Figure 5: Reverse direction to move backward through the states and displaying HEX

A screen shot of a computer

AI-generated content may be incorrect.

Figure 6: Reset back to STATE\_0 and starting cycle over

# Conclusion

In this lab, I learned to be able to design a simple state machine to move forward and backward through the various states for the specified design. This was able to show more of what we previously learned to implement a state machine, but moreover how to design one to accomplish the task and being able to handle various other signals such as reset and hold to modify the normal process of the state machine. What I would do differently if I were to start over with this lab, is create the testable components first and see that they work as expected, so that I could use them for proper debugging, as I had not initially done this, and trying to see it 1’s and 0’s are working as expected is a lot harder than looking at the type that ModelSim is able to provide.