Tyler Swenson

EE457

Lab Grad-Lab Report

4/9/2025

EE457 The Dizzy Snake

# Introduction

In this lab, we will be designing a state machine that simulates a snake of various lengths that’s trapped inside the DE10-Lite board. The snake, “Morton”, will continue spinning around the 7-segment display at 1 second increments. Morton will also be able to change the direction he’s moving depending on the switch being on or off. He will also change lengths that will be controlled by switches 1-4 using binary data for his link length.

# Theory of Operation

## Requirements

1. The design will reset when the reset signal is asserted low.
   1. Resetting will set all 15 segments when pressed.
2. The link length will be controlled by switches 1-4 using binary hex data.
   1. A link length of 0 is allowed and is completely off.
3. The snake will spin around the display at 1 increment per second.
4. Switch 0 will determine the direction of movement.
   1. Down/Off will make him move clockwise.
   2. Up/On will make him move counterclockwise.
5. All inputs and resets will be synchronized.

## Description of the Design

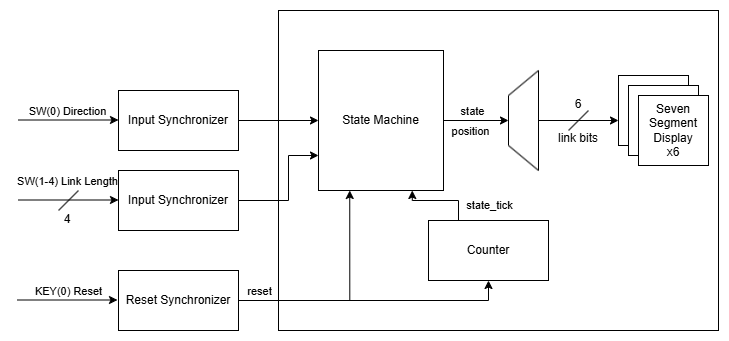


Figure 1: Design block diagram

In this design, the synchronized inputs for direction, link length, and reset will be fed into the state machine, which will determine the position and which direction the links lead. A counter will be used to ensure that the state proceeds in 1 second intervals by generating a tick. The snake will rotate around the outside of the 6 HEX displays with the length that is determined by the link length of switches 1-4 which will use binary hex. The snake will then rotate clockwise while the switch is off, and counterclockwise while the switch is on. The snake will continue to change when any input change is given. When KEY(0) is pressed, this will be the design reset and will set all 15 segments on while it is being pressed, and upon release the design will go back to the link length that is inputted. The state machine will generate the position of the lead link and will then be decoded to the remaining links that are on, and this will finally be fed into the hex displays.

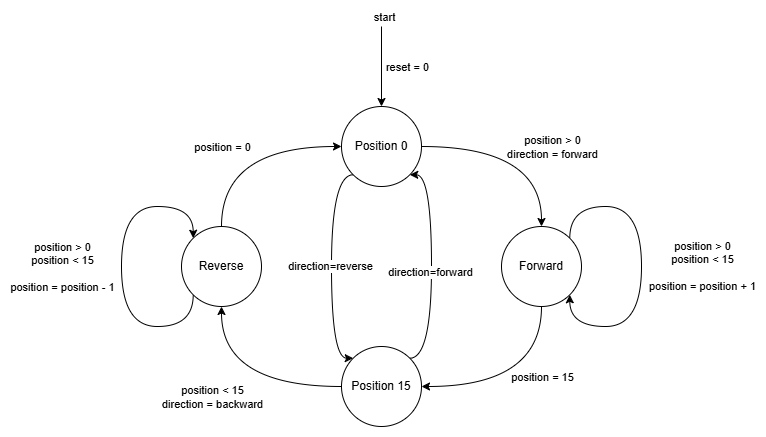


Figure 2: Design state machine

# Verification

## Test Plan

To test the design, a testbench will test both directions to ensure that all operations are valid in either direction. Various cases will test situations such as:

1. Regular operation (no matter the length).
2. Switching from a small to a large length.
3. Switching from a large to a small length.
4. The maximum link length (15).
5. The minimum link length (0).
6. Switching directions.
7. Resetting the design.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 3: Test conditions that will be tested in both directions

## Test Bench

### Forward Direction

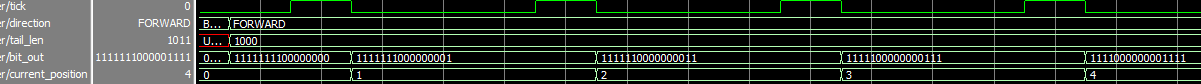


Figure 4: Test 1 – A noticeable length for the snake to move (8)

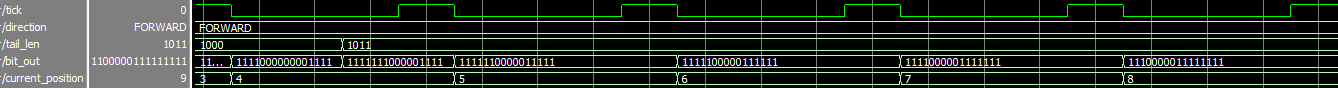


Figure 5: Test 2 – A larger link length (11)

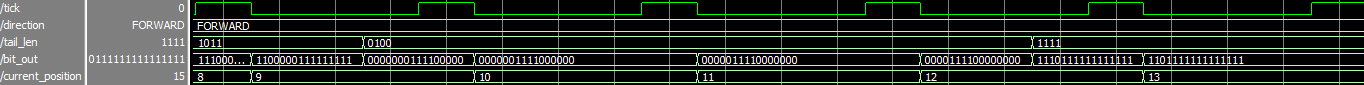


Figure 6: Test 3 – A small link length (4)

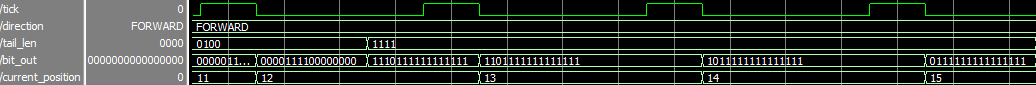


Figure 7: Test 4 – Maximum link length (15)

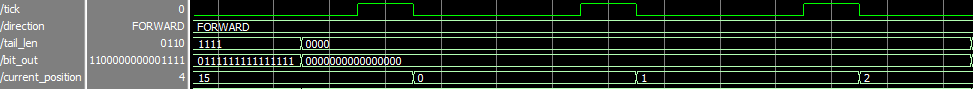


Figure 8: Test 5 – Minimum link length (0)

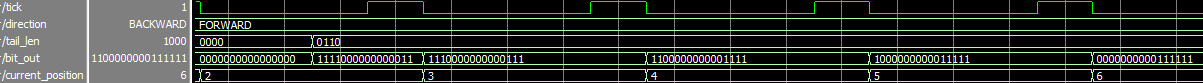


Figure 9: Test 6 – Set back to a normal length (6)

### Reverse Direction

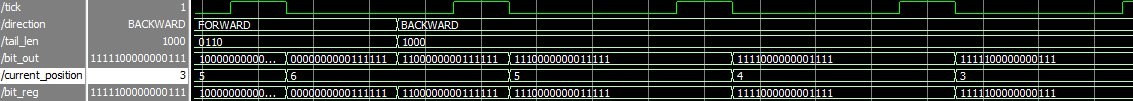


Figure 10: Test 1 – A noticeable length for the snake to move (8)

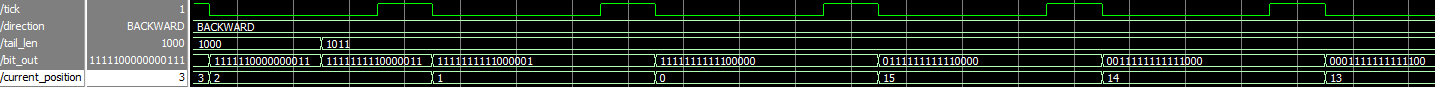


Figure 11: Test 2 – A larger link length (11)

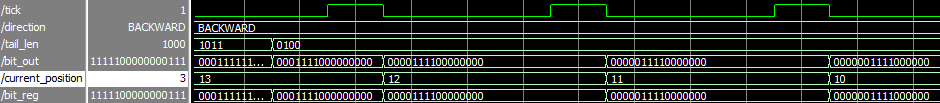


Figure 12: Test 3 – A small link length (4)

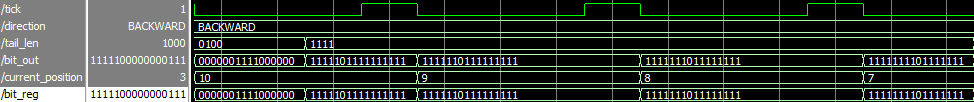


Figure 13: Test 4 – Maximum link length (15)

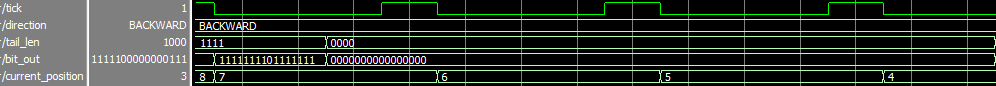


Figure 14: Test 5 – Minimum link length (0)

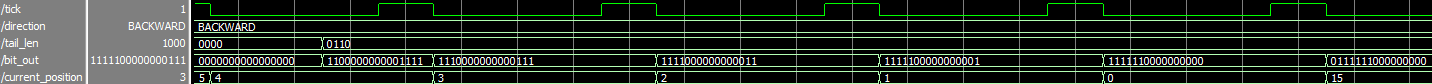


Figure 15: Test 6 – Set back to a normal length (6)

# Conclusion

In this lab, I learned to take what seemed like a fairly complex state machine design for the snake to rotate around the display board, into a working design that made sense to me. The challenge of this lab was to not hard code every single possible state that could be possible, as that would take too much effort and time to painstakingly write out every single state and each input and output. Instead, I was able to make a solution that was able to decode a simple position state machine, into the remaining bits for the snake to use. This also adapted well into being able to be used for both directions, which was a pleasant surprise.