Assignment 01

Implementing HDL for a 1-digit 7-segment display

This document and code are also available on my GitHub repo and made with the VScode "md to pdf" extension

Information

This small system verilog project shows how to implement the first step of a 1-digit 7-segment display as close to hardware as possible.

> *"it doesn't get deeper than this..."*

About sevenseg

Here the central I/O module gets initialized. It contains the LUT used to make decisions about the visual display of digital data. Since the wiring of the 7SD is manmade, it follows no particular mathematical rule, which means that an automation is impossible. Therefore, a 1:1 LUT is the best way to achieve a direct mapping between a 4bit interger input and a 7bit display. Additionally, it creates the negated output in every case. This makes the code long but readable and the numbers 0 to 15 only get mentioned once.

About tb_sevenseg

The testbench stimulates the module with all requested numbers (hex 0 to F). For this, it maps all I/Os to the testbench I/Os and displays them in text form (and of course as waves in the model-sim). To fill the input, a for-loop is utilized.

About sim_tb_sevenseg

This is the starting script for the model-sim software. In here standard procesdures are followed through according to the lecture of 14.10.21. Calling it from the model-sim software causes the simulation to start and waveforms to be displayed.

Results

Two kinds of results can be obtained from the simulation:

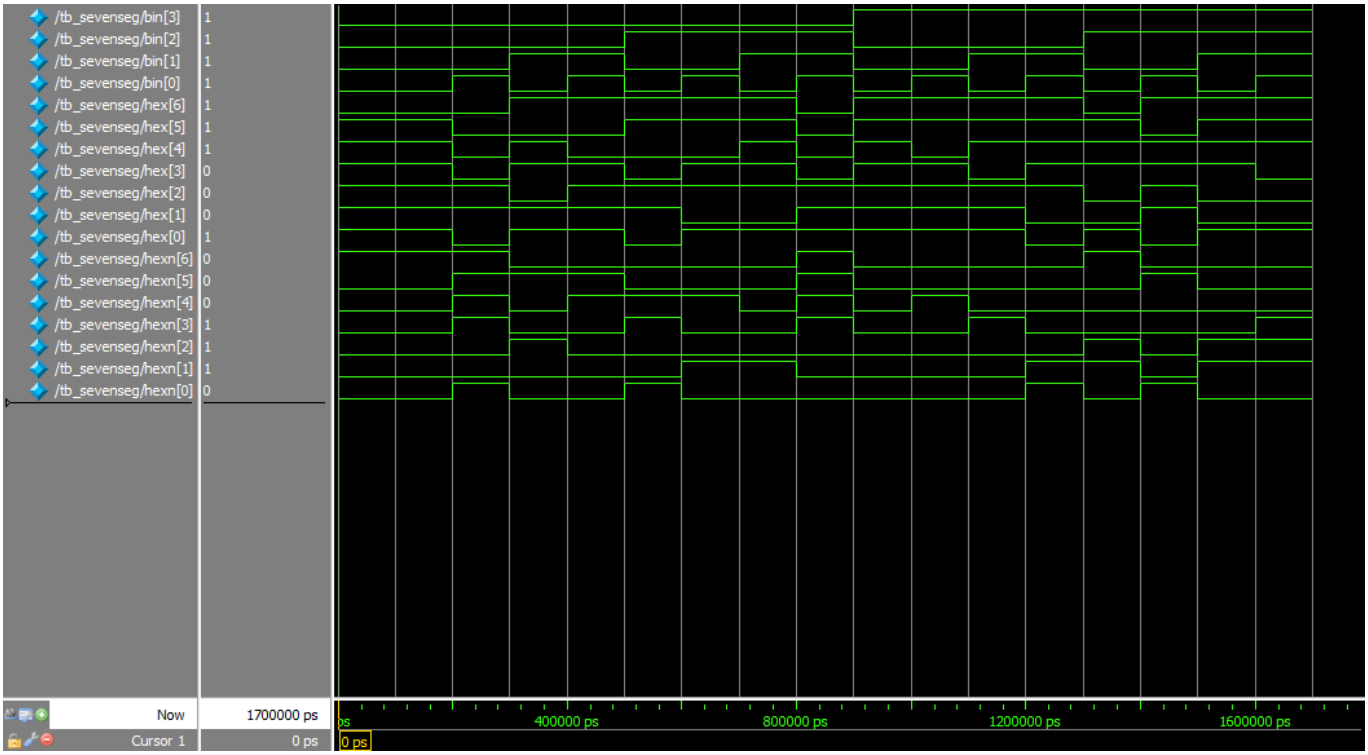
1. The coded output via `$dispay`
2. The waveforms from the simulation

It has to be stated that these waveforms do not visually tell much about the 7SD since they are not arranged like a 7SD display. The verification of data is therefore a quite on-the-nose one and has to be checked by a human.

Console out:

```
# ***********************
# Benching sevenseg...
# ***********************
#  0 --> 3f != 40
#  1 --> 3f != 40
#  2 --> 06 != 79
#  3 --> 5b != 24
#  4 --> 4f != 30
#  5 --> 66 != 19
#  6 --> 6d != 12
#  7 --> 7d != 02
#  8 --> 07 != 78
#  9 --> 7f != 00
# 10 --> 6f != 10
# 11 --> 77 != 08
# 12 --> 7c != 03
# 13 --> 39 != 46
# 14 --> 5e != 21
# 15 --> 79 != 06
# ***********************
# sevenseg benching end.
# ***********************
```

Waveforms:



The waveforms correspond to the desired output and negated output with the segment 0 (on top of the digit) being the LSB and the center segment the MSB. However, the console statement seems to be offset by one iteration. The second hex value should not be `3f` but `06`. It is unclear how this piece of code can produce this offset:

```
for(int i = 0; i < 16; i+=1) begin

    bin = i;

    $display("%d --> %x != %x", bin, hex, hexn);
    #100ns;

    end
```