06/18/22

Logic and Digital System Design (CS 303)
Term Project Report - Egg Timer

Group Members:
Hüseyin Alper Karadeniz
Enis Mert Kuzu

According to the configurations requested in the project document, we created the desired algorithmic state machines from scratch using Verilog, rather than the complex schematic circuit designs in Digital. In Verilog, we ensured that our functions run at the right moments through always statements. By using registers, we kept some variables that we want to hold as control variables. We arranged our outputs with the assign command.

Controller: We got six inputs for the "controller" as *rst*, *beginn*, *endd*, *reset*, *zero*, *clk*. We defined three different states (00, 01, 10) for our machine according to the respective status of *reset* and *beginn* inputs; and these states are used to ensure the functionality of the machine. We updated our state when *rst* or *clk* input came. We wrote an always statement that runs asynchronously on every state change:

When the state is statePreset (*reset* input is 1 and *beginn* input is 0), we stopped the counter by using the register controlling that we continue to count down in the next stages (countingReg) and made our load output 1 over the register (loadReg) so that sent an input to "downcounter" to load the given inputs from the user to the 7-segment displays.

When the state is stateStart (*reset* input is 0 and *beginn* input is 1), we made our start output 1 over the register (startReg) to start our "counter", also by using the register above (countingReg), we saved that we started the counter.

When the state is stateCountDown (both *reset* and *beginn* inputs are 0), while *endd* input is 1 and countingReg register is 1 so that we need to count down by 1, we gave an output of count down over register (countDownReg) so that triggered "downcounter" to count down the counter.

We have designed an always statement that works when *endd* input changes, and it likewise counts down one every time an *endd* input comes up.

Downcounter: We wrote a Verilog code that will update the 7-segment displays in each clock according to *load*, *count*, *rst*, *clk* and display inputs given by the user with switches. The function we have written updates the displays every time the clock (*clk*) hits in positive-edge, or *rst* input changes in negative-edge.

If *rst* input is 0, we reset the 7-segment displays that appears on the screen by making all displays 0.

If *rst* input is 1, and *load* input is 1, we loaded the inputs that the user gave through the switches to the displays, so the amount of time the user wanted to enter was reflected on the displays and here we wrote the code to reflect the inputs that the user will give as a maximum of 99:59 by logic.

If *rst* input is 1, *load* input is 0, and *count* is 1, we decreased the 7-segment display egg timer for one second by looking at the status of current displays from the registers so that counting down the egg timer. Also, when the 7-segment displays came to 00:00, hitZero register, which is checking if the egg timer is finished, becomes 1 meaning that the egg timer is finished; thus, *zero* output becomes 1 due to assign statement among them.

Note: The source codes of the projects are attached with this document.