**ECE 211: Digital Circuits I — Spring 2025**

**Lab 8: Sequential Counter**

By: Efe CIVISOKEN & Knycalus GREENJONES

16/04/2025

**Statement of Collaboration:**

During the lab, both collaborators worked together the achieve the goal of creating a 3-digit BCD counter, while Efe was more involved with writing the logic and submodular testing, Knyc was more involved with double-checking code syntax in real-time. Knyc was responsible for writing the *INTRODUCTION* and *DESIGN* parts Efe was responsible for writing the *IMPLEMENTATION*, *TESTING, RESULTS & DISCUSSION* parts as well as writing this *statement of collaboration.*

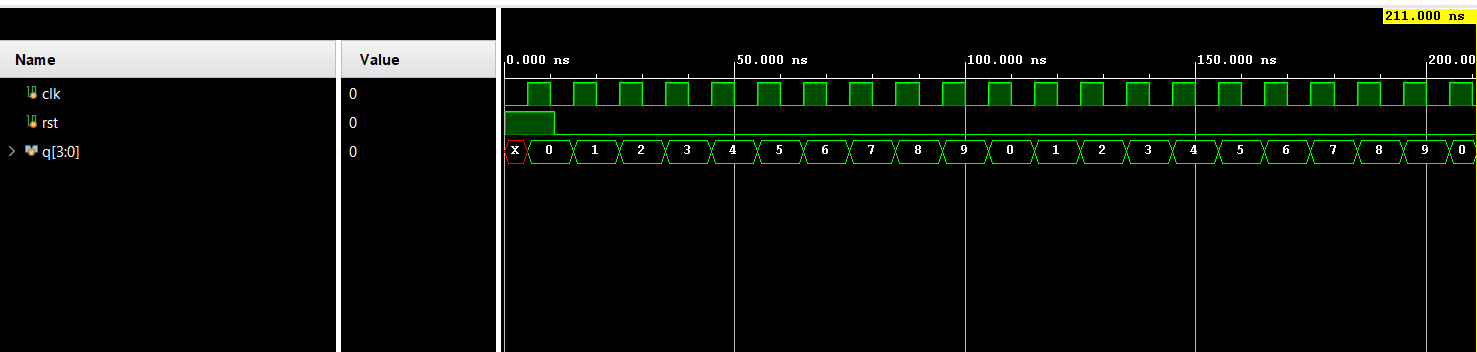
**Time Spent:** **2 hours & 45 minutes.**

**I. INTRODUCTION**

The purpose of this lab was to create, instantiate, and display a number counter using sequential logic on a seven segment display. The three-digit number will be dependent on the prior values in each placeholder, and the counter will reset to 0 when all three digits are 9. This, essentially, mimics a counting sequence that maxes out at the “whole” number 999.

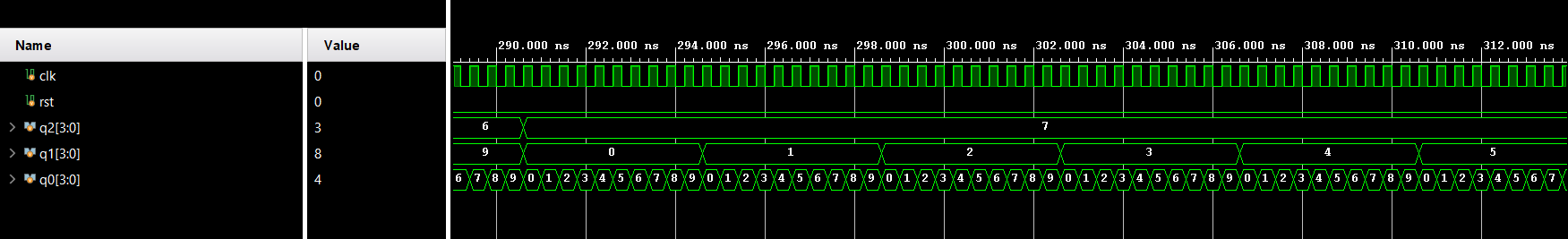
**II. DESIGN**

* Our design called for an implementation of “rst”, a function that resets the value of each index of “Q” (each digit) to 0 when its value is equal to 9. Otherwise, the digit continuously increases by 1. We called this module “counter\_4b”. Below is a figure of the waveform to give representation of said logic.



**Figure 1**: Waveform for counter\_4b

* Additionally, “Q” is a 3 bit input in this module; this is to account for the 3 “digits” that will be utilized. In the figure below, each value of Q is below 9. Q[2] is 3, Q[1] is 8, and Q[0] is 4. Because none of them are equal to 9, input “rst” has a value of 0. If any of them were **equal to** 9, rst would have a value of 1, and that given value would be reset to 0. Below is a figure of the waveform to give representation of said logic.



**Figure 2:** Waveform for three\_digit\_counter\_4b

**III. IMPLEMENTATION**

In our top-level module lab08\_top, we integrated the essential components to build a functioning 3-digit BCD (binary counted decimal) counter system with display output on a 7-segment display. The module receives a 100MHz base clock and control signals from the lower two switches, where SW[0] acts as the enable (EN) and SW[1] serves as the reset (rst). A derived clock is created using the clkdiv module, which reduces the 100MHz input to a lower frequency using a parameterized division factor set via .DIVFREQ(60). This slower clock is then used to drive the counter module, making changes on the display perceptible to the human eye.

The counting logic is handled by the three\_digit\_counter\_4b module, which receives the derived clock, enable, and reset signals. It produces three 4-bit outputs (q0, q1, q2), each representing one decimal digit of the total count in BCD format. These outputs are then passed into the provided counter\_display module, which handles digit multiplexing and BCD-to-7-segment conversion, allowing us to output the current count visually through the SEG and AN ports.

The design follows a modular approach, where each functional component is encapsulated within its own Verilog module. This not only ensures clarity and ease of debugging, but also follows best practices by avoiding the use of behavioral logic such as always\_ff directly within the top-level file. Instead, all stateful and combinational logic is cleanly encapsulated within submodules, making the system easier to simulate, verify, and reuse.

module lab08\_top(

input logic clk100MHz,

input logic [15:0] SW,

output logic [15:0] LED,

output logic [6:0] SEG,

output logic [7:0] AN

);

logic rst;

logic clk;

logic EN;

logic [3:0] q2, q1, q0;

assign rst = SW[1];

assign EN = SW[0];

clkdiv #(.DIVFREQ(60)) D0(.clk(clk100MHz), .reset(1'b0), .sclk(clk));

// Implement your lab here

three\_digit\_counter\_4b c(.clk(clk),.EN(EN), .rst(rst),.q0(q0),.q1(q1),.q2(q2));

counter\_display( .d0( q0 ),

.d1( q1 ),

.d2( q2 ),

.clk(clk100MHz), .seg\_n(SEG), .an(AN));

endmodule

**CODE 1:** The Implementation of Top Module lab08\_top.sv

**IV. TESTING**

Testing was carried out both at the submodule level using dedicated Verilog testbenches and at the system level through behavioral simulation and FPGA deployment. At the submodule level, we tested each module to ensure proper functioning. The counter\_4b and three\_digit\_counter\_4b modules were tested for correct BCD counting behavior, including incrementing from 000 to 999, resetting when the reset signal was high, halting when enable was low, and resuming correctly once enabled again.

Following successful testbench verification in submodular level, we proceeded with top-level behavioral simulation and real-time testing on the FPGA board. During initial attempts, we encountered issues due to using the 100MHz clock directly in the counter, which caused instability in the display and unexpected counting behavior. This was resolved by ensuring that the clock divider was correctly connected and used as the source clock for counting logic. Another early issue involved trying to implement always\_ff logic directly in the top module, which violated modular design principles and created some minor bugs. After refactoring this logic into appropriate submodules, the design became structurally sound and fully testable.

Once these bugs were resolved, the implementation performed correctly both in simulation and on hardware. On the FPGA board, the system consistently counted from 000 to 999, reset either when the reset switch was activated or when it naturally reached 999, and stopped counting when the enable switch was off. When the enable was turned back on, the counter resumed from its previous state, confirming that state was preserved correctly. This validated the proper functioning of all integrated modules and confirmed that the design operated as intended under all specified conditions.

**V. RESULTS & DISCUSSION**

The final implementation functioned as expected. Submodule-level testbenches confirmed the correctness of each component in isolation, and full-system behavioral simulation provided a complete picture of the system’s runtime behavior. Once deployed on the FPGA board, the system counted reliably from 000 to 999, displayed the values accurately on the 7-segment displays, and responded correctly to the enable and reset switches. The reset mechanism worked both manually through the switch and automatically upon reaching 999. Additionally, the enable switch properly paused and resumed counting from the saved state. These results demonstrated that the design was structurally correct, modular, and functionally robust across both simulation and hardware deployment.