CPE 324-03: Advanced Digital Logic Design Laboratory

Lab 1

Solderless Breadboard Prototyping and Verification of Small-Scale Digital Logic

Submitted by: Hannah Kelley

Date of Experiment: January 6, 2026

Report Deadline: January 13, 2026

Demonstration Deadline: January 13, 2026

1 Introduction

The purpose of this lab is to learn how to use rapid prototyping techniques using the DE2-115 lab board and the UAH ECE Breakout board. Two experiments were conducted: one using the DM74LS08 and DM74LS32 Quad 2-input AND and OR Gate chips to create a combinational logic circuit and the other using the DM74LS163 and DM74LS161 4-bit counters to create an 8-bit counter.

Using the circuit diagrams and pin maps given in the laboratory manual, experiment one was built using breadboarding to turn on LEDG0 only when the Boolean condition (I0 | I1) & (I2 | I3) & (I4 | I5) was true. Circuit diagrams, pin maps, chip diagrams, and breadboarding were used to implement an 8-bit counter from two separate 4-bit counters for experiment two.

2 Experiment Description

2.1 Theory

2.1.1 Breadboards

Breadboards are rapid prototyping tools that allow for the creation and testing of circuits without soldering. This allows for components and wires to be easily inserted and removed, allowing for quick modification and debugging where necessary. Breadboards feature two power rails, generally used for ground and Vcc, which can be connected to the breadboard's rows using wire. Each row on a breadboard internally resembles a separate strip of wire, allowing all components on a row to be connected to the same part of a circuit.

In this lab, the UAH ECE Breakout Board was interfaced with the Terasic DE2-115 FPGA to provide convenient access to the FPGA's input and output pins. The breakout board routes selected FPGA signals to clearly labeled header pins that can be easily connected to a breadboard using jumper wires. This configuration allows external components such as LEDs, resistors, and switches to be incorporated into the FPGA-based system without directly wiring to the FPGA board itself. By using the breadboard and breakout board together, the circuit could be assembled, modified, and tested efficiently while maintaining reliable electrical connections and minimizing the risk of damage to the FPGA hardware.

2.1.2 Combinational Logic

Combinational logic refers to digital logic circuits whose outputs depend solely on the present values of their inputs. Combinational circuits do not contain any form of memory and therefore do not retain information about previous input states. As a result, any change in the input signals produces a corresponding change in the output signals after a brief propagation delay inherent to the logic devices.

Combinational logic circuits are constructed using basic building blocks called logic gates. These gates include AND, OR, NOT, NAND, NOR, XOR, and XNOR gates. By combining these gates, Boolean expressions can be implemented to produce specific logical relationships between inputs and outputs.

In this lab, combinational logic was implemented using the DM74LS08 and DM74LS32 quad two-input AND and OR gate integrated circuits. These chips were mounted on a breadboard and interfaced with the Terasic DE2-115 FPGA through the UAH ECE Breakout Board to create and test the desired logic circuits.

2.1.3 Sequential Logic

Sequential logic refers to digital logic circuits whose outputs depend on both the current input values and the previous state of the system. Unlike combinational logic, sequential logic circuits contain memory elements that allow them to retain and utilize past circuit states. Sequential memory elements include flip-flops and latches, both of which store binary data and change state in response to clock signals and control inputs.

In this lab, sequential logic was implemented using the DM74LS163 and DM74LS161 4-bit counter integrated circuits. These ICs were interfaced with the Terasic DE2-115 FPGA through the UAH ECE Breakout Board to create and test the desired logic circuits.

2.2 Experiment 1 - Multi-IC Combinational Logic Design

For this experiment, I was tasked with implementing a combination logic circuit on a breadboard connected to the Terasic DE2-115 FPGA through the UAH ECE Breakout Board. The circuit I was tasked with implementing is shown in Figure 1.

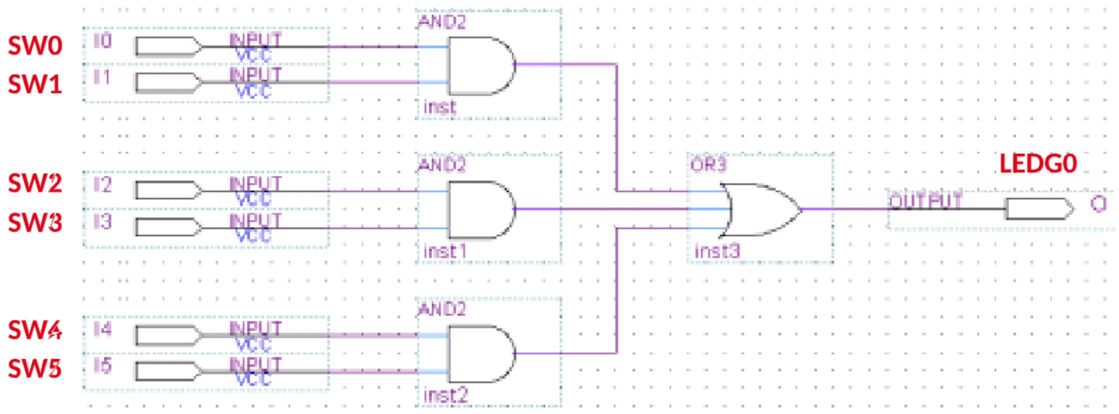

Figure 1 – Experiment 1 Circuit Schematic

2.2.1 Experiment 2 Procedure

In order to implement this circuit, the ICs and UAH ECE Breakout Board were connected as shown in Figure 2. As you can see, the 3-input OR gate shown in Figure 1 was implemented using two 2-input OR gates in series to achieve the same result.

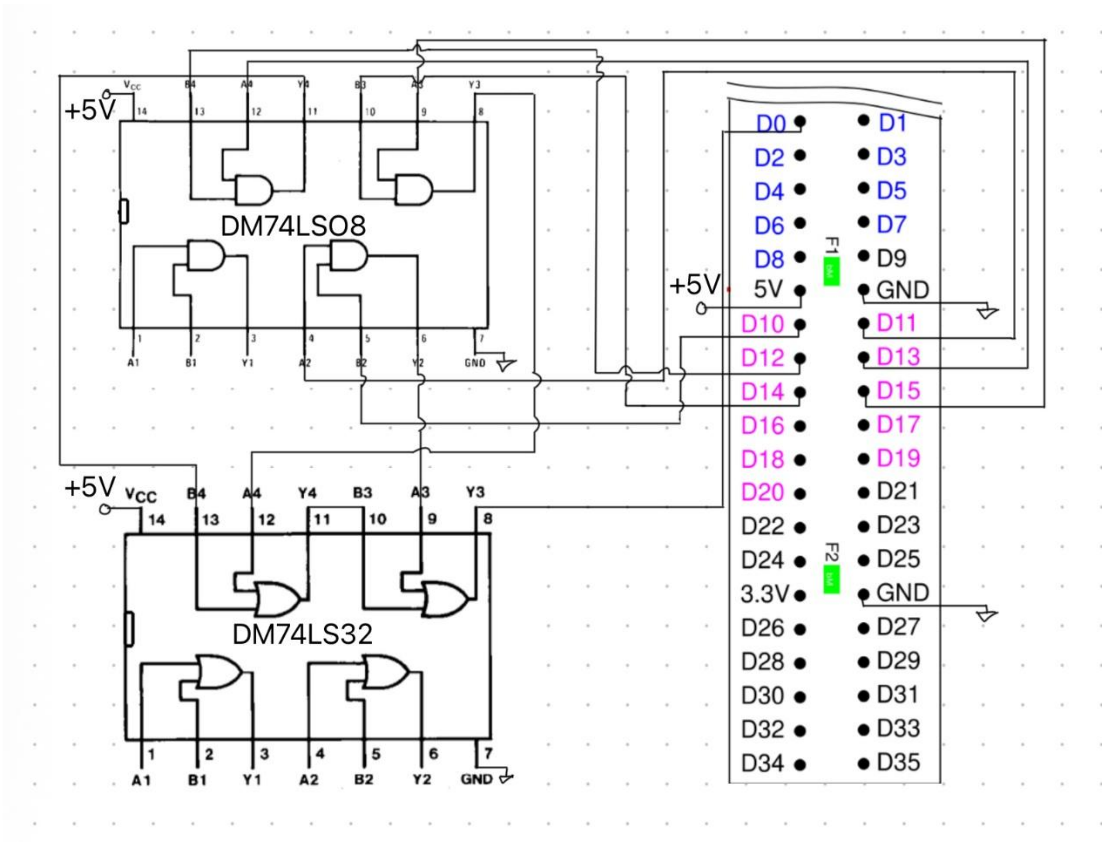

Figure 2 – Experiment 1 Physical Configuration

The final Boolean expression given by my configuration is:

$$O(I0, I1, I2, I3, I4, I5) = (I0+I1)(I2+I3)(I4+I5).$$

During testing, the input values were varied by toggling switches 0 through 5 on the DE2-115 FPGA, producing all possible 6-bit input combinations ranging from 000000 to 111111 in binary. The circuit output was displayed on LEDG0 on the FPGA. When LEDG0 was off, the circuit evaluated to false; when LEDG0 was on, the circuit evaluated to true.

2.2.2 Experiment 1 Results

The expected results were determined analytically using Boolean algebra. These theoretical results were then compared to the measured outputs obtained during testing. The expected results exactly matched the actual results for all tested input combinations. Both sets of results are presented in Table 1.

Inputs (Test Vectors)						Output, O	
I0	I1	I2	I3	I4	I5	Expected	Actual
0	1	0	0	0	0	0	0
1	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0
0	0	1	1	0	0	1	1
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
1	0	1	0	1	0	0	0
0	1	0	1	0	1	0	0

Table 1 - Partial Truth Table for Experiment 1

To completely and accurately verify the operation of this combinational logic design, all possible input combinations must be tested. Since the circuit contains six input variables, there are $2^6 = 64$unique switch combinations, ranging from 000000 to 111111 in binary. While only a subset of these combinations was tested during the experiment, full verification would require systematically applying every possible input pattern. The most efficient method to accomplish this would be to begin at the lowest binary value (000000) and incrementally count upward to the highest value (111111), observing and recording the output for each case.

Several challenges were encountered during the implementation of the circuit. Because this was the first time using the Terasic DE2-115 FPGA and the UAH ECE Breakout Board, initial difficulty was experienced in understanding the pin assignments and signal routing between the FPGA, breakout board, and breadboard. Additionally, interpreting the pinout tables provided in the laboratory materials required careful attention. These issues were overcome by closely referencing the documentation, verifying each connection against the pin assignment tables, and testing individual signal paths before integrating the full circuit. This systematic approach allowed the design to be successfully implemented and verified.

2.3 Experiment 2 - Multi-IC Sequential Logic Design

For this experiment, I was tasked with implementing an 8-bit binary counter using two synchronous 4-bit binary counters on the DM4LS161 and DM74LS163 IC chips. The 8-bit counter allows for a semi-customizable clock rate using the DE2-115 built-in clock settings and a programmable starting value. The clock rate can also be implemented manually using a switch and a button on the FPGA. The wiring diagram for this design is shown in Figure 3.

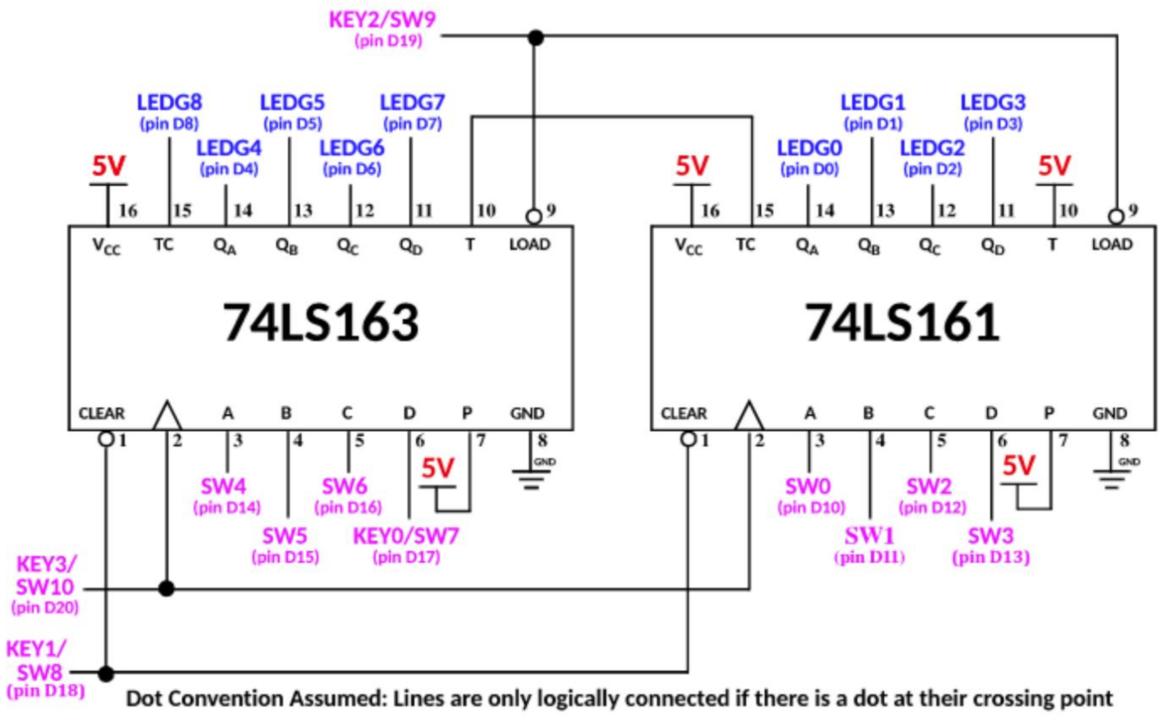

Figure 3 – 8-Bit Counter Diagram for Experiment 2

2.3.1 Experiment 2 Procedure

To implement this design, the wiring diagram shown in Figure 3 was followed to construct the 8-bit counter. Careful attention to wire placement was critical, particularly given the large number of connections required, to ensure reliable operation and prevent short circuits or miswiring.

The counter interfaces with several keys and switches on the FPGA. Switches 0-7 can be used to load the desired starting value into the counter when the LOAD signal present in switch 9 is inactive. The desired value is represented by the states of switches 0-7 with a logic low representing a binary 0 and a logic high representing a binary 1. Switch 8 represents the CLEAR signal. The clock signal found in pin D20 can be toggled in between manual and automatic mode using switch 17. When in manual mode, switch 10 and key 3 can be used to manually toggle the clock. When in automatic mode, the speed is determined using the signals in switches 15 and 16.

2.3.2 Experiment 2 Results

To test the design, a series of steps found in the Lab 1 manual were executed. The LOAD and CLEAR signals are assumed to be inactive at the start of step 1.

1. **Load an initial count value of 00001110 into the counter. Discuss how this was done and when it took effect. Return the LOAD signal to its inactive state.**

To load a value into the counter, set switches 0-7 to the desired states assuming the corresponding LEDs represent a binary number. To represent 00001110, switches 1-3 should be on and the remaining should be off. LEDs 1-3 should light up as a result. The new number will be loaded into the counter once the LOAD signal is returned to a low value and the clock signal experiences its next rising edge.

2. **With the count value of the counter set initialized to 00001110, generate a manual clock by pressing the KEY3 button for each clock cycle. Record what happens to the count value and the ripple counter (also called the Terminal Count, TC) output as you clock your design for three clock cycles.**
   To run the clock in manual mode, switch 17 should be turned off. Once in manual mode, each press of key 3 corresponds to a clock cycle. Pressing key 3 three times results in 3 clock cycles passing. The counter increments with every clock cycle, so the number now displayed in the counter is 00010001. Additionally, every clock cycle is visually represented by the TC value shown in LED8. While key 3 is pressed, the clock is in a high state and LED8 is on. Once the key is released, the clock enters a low state and LED8 turns off.
3. **Temporarily rewiring the input P on Pin 7 of the 74LS161 IC so that it is connected to ground instead of 5V. What happens when you manually advance the clock signal five clock cycles? Return the connection of input P on Pin 7 of the 74LS161 so it is at 5V. What happens when you manually advance the clock signal five clock cycles?**
   Temporarily rewiring the input P to ground breaks the counter. Since the P and T inputs must both be high to increment the count, this is expected. When input P is returned to 5V, the counter functions as expected again. Advancing the counter 5 times in this states results in 5 clock cycles passing and the counter to increment 5 times. Assuming the counter started at 00010001 as seen at the end of step 2, the new counter value is 00010110.
4. **Using an automatically generated but slow speed clock from the DE2-115 demonstrate a forward count starting from 00001100 to at least 00011111.**
   When the counter is set to a slow automatic speed, the counter functions as intended. The incremented count is visible on LEDs 0-7 at the rising edge of each clock cycle.
5. **Run the clock with IC in automatically generated clocking mode with clock set to a high speed (SW17, SW16, SW15 all in the up position). What happens to the output?**
   When the clock is set to high using the automatically generated clock speeds, the change is too quick for the human eye to detect. The LEDs appear to always be one. No flicker or increment is visible.

6. **Return the clocking mode to manual. Initialize the count value to 11111111. CLEAR input without advancing the clock. What happens to the output? Now with the CLEAR input still activated advance the clock one cycle. What happens then to the output?**
   When the count is initialized to 11111111 and the CLEAR signal is activated, no change occurs at first. When the clock is advanced by pressing key 3, the clear signal is recognized on the rising edge and the counter is cleared to 00000000.
7. **What happens if you use the SW10 key to manually clock your design instead of KEY3? Explain why the design behaved in the manner you observed.**
   When switch 10 is used to manually clock the counter, the same functionality is observed as incrementing the clock manually using key 3. LED8 turns on as long as the switch is in a high logic state. The counter increments on the rising edge. When the switch is returned to a logic low nothing happens. This is because the counter is designed to trigger events on a rising edge only.

The counter functioned as expected throughout the experiment. Some initial issues arose due to faulty or inconsistent wiring, but these problems were resolved immediately once the wires were replaced. Additionally, some difficulty was encountered in determining which switch positions corresponded to specific outputs, but careful testing and observation clarified these relationships.

3 Conclusion

The experimental results closely matched the expected outcomes from Boolean analysis, confirming the accuracy of the circuit design. Minor issues, such as intermittent signals, were traced to faulty or loosely connected wires and resolved through careful troubleshooting.

This experiment demonstrates practical techniques for implementing and testing digital logic circuits, including the use of ICs, breadboards, and FPGA interfaces. Systematic testing of all input combinations proved essential to verify correct operation. These skills are directly applicable to real-world engineering, where reliable prototyping and debugging are critical. Overall, the lab reinforced both theoretical and practical aspects of digital design, providing a foundation for building more complex systems in later labs.