

EXPERIMENT

4

SEQUENTIAL LOGIC CIRCUITS – Part 1

VERSION F05

In this experiment, you will design, implement and verify two different synchronous counters. After performing this experiment, you should be able to:

- 1) Understand the difference between the behavior of a latch and a flip-flop,
- 2) Design sequential circuits for implementation in an FPGA,
- 3) Use hierarchy in circuit design, and
- 4) Use an FPGA-prototyping system to verify sequential circuit function.

In this lab, there are two principal circuits that you will design. The first is a synchronous binary-coded-decimal (BCD) counter with asynchronous reset and a clock enable input to permit individual counters to be cascaded into multi-digit synchronous counters. You will design this counter from individual flip-flops and input logic. You will also convert this circuit to a hierarchy block to make it simpler to create multiple instances of the counter. To test your counter, a schematic sheet acting as a test platform is provided for you. The second circuit to be designed is a divide-by-N counter, where N is a specified constant. However, in this case, you will implement this counter using a counter primitive that provides a 16-bit counter with synchronous reset, and simply add logic to force a counter reset at the desired count.

In synchronous design, our efforts are premised on the assumption that the clock is occurring simultaneously at all flip-flops. While this is simple to assume, actually getting the clock to all flip-flops with an acceptable delay is not always so easy. To facilitate this, the FPGA has dedicated pins, buffers and routing channels specifically designed to be used for clock distribution. When doing synchronous design, we need to ensure that our clocks are in fact using these resources.

In the lab, you will also do a study of the difference in time-based behavior between a latch and a flip-flop. The bit file for this part of the experiment will be provided. As preparation, review the subsection on the D Latch and the section on Flip-Flops in Mano and Kime.

4-1 PRELAB

For the prelab, you will design and test (in simulation) a cascadable BCD counter and a divide-by-N counter.

BCD COUNTER DESIGN AND SIMULATION

Design a BCD counter using D flip-flops. In particular, you will use the Altera primitive **DFFE**, which is a D flip-flop with clock enable (ENA) input and asynchronous preset/clear inputs. You will design your counter in the manner illustrated in Mano & Kime 3rd edition, pages 337-339. A state table and K-maps are provided for your use

at the end of this document. When you create your hierarchy block, you will name it **BCD_CNTR** and provide the terminals listed below. When completing the state tables, remember that we have chosen flip-flops that include inputs for clock enable and asynchronous clear, so these two functions do not need to be implemented in the D-FF input forming logic.

Inputs	CLK	Clock
	CE	Clock Enable
	CLR	Clear (asynchronous)
Outputs	Q8, Q4, Q2, Q1	Counter value (Q8 = most significant bit)
	TC	Terminal count output for cascading (high only when count = 9 and CE = 1)

BCD COUNTER DESIGN AND SIMULATION

- 1) Complete the state table and flip-flop inputs provided at the end of the document with the required information for this counter design.
- 2) Fill in the K-maps provided at the end of this document with the information from the state table and flip-flop inputs. Extract the required input logic equations and note them adjacent to the corresponding K-map.
- 3) Create a new Quartus project named **BCD_CNT** in a new directory LAB4\BCD_CNT. Import the standard ECE 351 pin assignments.
- 4) Create a new schematic diagram file named **BCD_CNTR**. You will design the BCD counter logic in this file, and then use it to create a block.
- 5) Place four (4) DFFE flip-flops in your **BCD_CNTR** schematic file, add input/output pins named as shown above, and add the required input and output logic. Since this flip-flop has clock enable implemented in the primitive, we do not need to incorporate it into our input logic (which would complicate things more than a little bit). Instead, we can simply tie together all of the flip-flop ENA inputs and use that as the clock enable input to the counter.
- 6) Functionally simulate your counter and verify its functionality, debugging and modifying as necessary. To do this, first set **BCD_CNTR** as the top-level entity by clicking on the **Hierarchy** tab, the right click on **Compilation Hierarchy**, select **Settings...**, and under **General**, set the top level entity to be **BCD_CNTR**.
- 7) Compile your project, then open the Simulator Tool. Set it to **Functional**, and click **Generate Functional Simulation Netlist**. Then, select **Open** to open the waveforms window, and use Node Finder [set the Filter to Pins: (unassigned)] to select all of the **BCD_CNTR** input/output pins. Then, set **CE** to a 1 for the duration of the simulation, set **CLK** to a 10ns clock, and set **CLR** to 1 for the first 10ns then 0 afterwards. Save the waveforms, and run the simulation.
- 8) When you believe your counter is functioning properly, use it to create a hierarchy block named **BCD_CNTR**.
- 9) Print the schematic of your BCD counter for submission with your prelab report.
- 10) Download the schematic sheet **CNTR_TEST** from the course web page. You will use this schematic (shown below) to test your counter blocks in a cascaded connection. Add the schematic sheet to your project, and set it as the top level entity. Compile and then devise a simulation waveform set that will test the counters through a complete 0 to 99 and back to 0 counting cycle. (Organize the waveforms so that the outputs of each counter are grouped together.) Run your simulation, and if necessary, go back and make the required changes to you counter design. Once you are satisfied with it, print a single page of your simulation trace (at a suitable scale) that shows the operation of the least-significant counter through one complete cycle (0-9). Change the scale appropriately, and print a single page of your trace that shows the operation of the most-significant counter through one complete cycle.

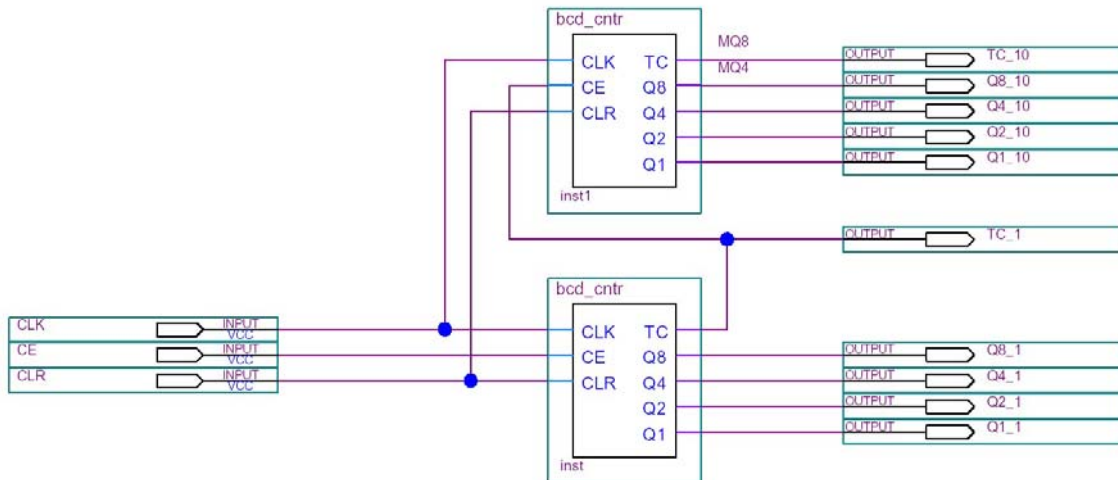


Figure 1 Cascaded BCD Counters Test Set-up

DIVIDE-BY-2000 COUNTER DESIGN AND SIMULATION

Here, you will design a divide-by-N counter, where $N = 2000$. (This value will be very useful in the next lab for generating a 1ms delay based on the FPGA boards 2.00 MHz clock.) However, instead of completely designing the flip-flop logic as you did for the BCD counter, you will use a pre-designed 16-bit counter block (COUNT16) to simplify the implementation. Specifically, you will design and implement combinational logic to force a reset of the counter after the count reaches 1999. (Note that a handy simplifying assumption is that the counter will never get higher than 1999.) This same signal used to reset the counter will also be used as the output (TC – terminal count) of the divide-by-2000 counter.

DIVIDE-BY-2000 COUNTER DESIGN AND SIMULATION

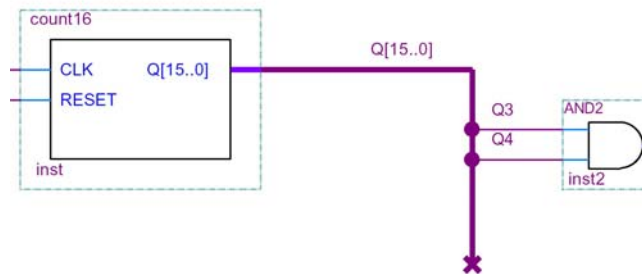
- 1) Create a new project named **DIV2000** in a new directory LAB4\DIV2000. Import the standard ECE 351 pin assignments. Create a new schematic sheet **DIV2000**. (Note that this will automatically be selected as the top-level entity in your project.)
- 2) Download the file **COUNT16.ZIP** from the course web page and extract the two files in it (**COUNT16.V** and **COUNT16.BSF**) to your project directory. These are the files that contain the design and symbol for the **COUNT16** block.
- 3) Place one **COUNT16** counter block by right-clicking in the schematic, select **Insert symbol...**, and browse to find **COUNT16.BSF**. This is a 16-bit counter with a synchronous reset input.
- 4) Add a single input, a single output, and the necessary logic to force a reset as required to implement a modulo-2000 counter. This reset signal will also be used as the output of your counter since it will be active for only one clock cycle out of the full count. See the box below for information on connecting to buses.

Connecting to buses: The output of the **COUNT16** counter block is a 16-bit bus. Connecting single nets to a bus is a little bit different from what you have done to this point, and requires us to draw what are called 'bus taps'. In order to get at the individual lines in a bus, we must do two things.



First, to run a bus from the output of the counter to your logic, you use the "Orthogonal Bus tool" button pictured here. The operation of the bus drawing button is vary similar to drawing nets. Select the button, and starting at the counter block, run the bus out. End the bus, the right click it and set its name to match the counter outputs, i.e. Q[15..0].

With that done, you can now connect whatever logic you have to the bus. Draw a regular wire from the desired pin on the gate and connect it to the bus. You then name this net as the bus name followed by the desired number. So for example, name the net Q15 to connect to the counter's most significant bit. The below exampleshows bits Q3 and Q4 being connected to an AND gate.



- 5) Assign your counter input to **CLK0** (which is the FPGA board's 2.00MHz oscillator), and assign your counter output to **BAR1_1**.
- 6) Functionally simulate your design using a 100MHz clock to stimulate the CLK input, and submit a single page of the simulation trace that shows one complete cycle of the counter output. To get an accurate time measurement, insert a second Time Bar (right-click and **Insert Time Bars...**), and place the Time Bars at the appropriate points to show the time required for a single cycle of the counter output. (Note: To do this accurately, you will need to zoom in on the trace to set one end of the measurement, and then scroll to the other end of the measurement. Then, zoom out so that both ends of the measurement are visible.)

Changing the Simulation Length: To change the default length of the vector waveform files used for simulation, select **Tools→Options** from the Quartus menu, then select **Waveform Editor** to see the waveform file options. You can set the "**default file end time**" here; the setting will affect all new VWF files that you create. Also, the VWF files are simply ASCII text files, so you can open them using **File→Open...** (but specify "**Text**" instead of "Auto" in the **Open as:** box at the bottom of the open file dialog) and then edit the file directly. If you look at a simple VWF file (i.e. just a clock signal), it's pretty obvious how the basic syntax works.

- 7) Print your schematic for submission with the prelab report.

PRELAB REPORT

Your report on 8.5 by 11 inch pages should consist of the following:

- 1) a cover with the experiment number, experiment name, your name, name(s) of the other team member(s), and your instructor's name, and your lab section,
- 2) Your state table and K-maps for the BCD counter,
- 3) Your annotated simulation traces and schematic diagram for the BCD counters.
- 4) Your annotated simulation trace and schematic diagram for the divide-by-2000 counter.
- 5) All questions in the question box below with your individual answers.

1. If we did not have a clock enable input on our BCD counters, we would have to cascade them by connecting the TC output of one to the clock input of the next most significant. What would be the effects (good and/or bad) of connecting them this way, as compared to using the clock enable input?
2. You implemented a divide-by-N counter, where N was a fixed value based on the logic driving the counter reset. What changes would you need to make to the divide-by-N counter so that the value of N could be an arbitrary value stored in a register so that it can be changed when desired (i.e. likely under control of some other logic)? Clearly describe your approach – a detailed design is not required but a block diagram may be very useful in conveying your methodology.

4-2 LAB WORK

WARNING: All lab results and all answers to questions or discussion are to appear in the lab reports of individual students. All tangible lab results are to be identical (unless indicated otherwise). When a printout of results is specified, a copy should be made for each team member. All answers to questions or discussion are to be the work of individual students, not the lab team. Evidence of collaboration on these aspects of a report within or between teams will be noted and is subject to University disciplinary action.

EQUIPMENT NEEDED

In addition to the equipment already on the lab bench, your instructor will have you check out a plastic tray containing:

- 1) an FPGA prototyping board,
- 2) a power supply module for the prototyping board,
- 3) two scope probes,
- 4) a logic analyzer probe,
- 5) four black logic analyzer extension wires in a plastic bag, and
- 6) some blue interconnect leads.

WARNING – LAB EQUIPMENT HANDLING: Much of the lab equipment is small and delicate. In particular, this applies to the XESS prototyping boards, scopes, scope probes, and logic analyzer probes. So please be careful and handle the equipment with a light and careful touch and do not use the scope probes with the grabbers removed. Perform wiring on the board ONLY with the power disconnected.

LATCHES AND FLIP-FLOPS

The circuit you will be exploring, given in Figure 2, consists of two nearly identical circuits, one of which contains a D flip-flop and the other of which contains a D latch. The bit file for this design is provided for you, you don't need to do the actual design.

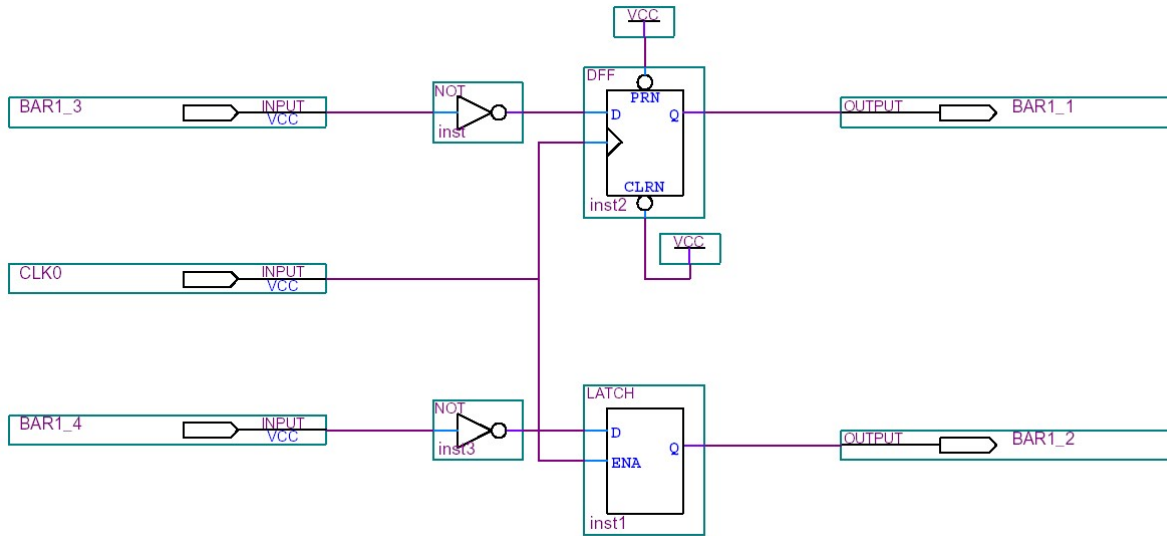


Figure 2 Latch and Flip-Flop

The two circuits you will investigate consist of a storage element, in one case, a flip-flop, and, in the other, a latch. In each case, the D input is driven through an inverter by the respective Q output. In the circuit containing the flip-flop, in response to each positive edge of the clock signal CLK0, the flip-flop output Q toggles to the opposite value, behaving like a one-bit binary counter of clock pulses. In order to compare this predicted behavior with that of the D latch's response to a positive level ($G = 1$), proceed as listed in the steps below.

- 1) Perform a POWER DOWN sequence.
- 2) Using a blue lead, connect BAR1_1 and BAR1_3 together on JP8. This connects the flip-flop Q output to the inverter driving the flip-flop D input.
- 3) Using a second blue lead, connect BAR1_2 and BAR1_4 together on JP8. This connects the latch Q output to the inverter driving the latch D input.
- 4) Attach logic analyzer leads DO, D1, and D2 to CLKIN (on JP12, this is the CLK0 input signal), BAR1_1, and BAR1_2 (respectively) on the FPGA board.
- 5) Perform a POWER UP sequence on the FPGA board.
- 6) Download the file latch_vs_ff.sof, which is available from the course web page. Use the Programmer tool to configure the FPGA with it.
- 7) Set up the oscilloscope to observe the values on CLKIN, BAR1_1, and BAR1_2. Include one or two clock periods in the display, whichever allows the flip-flop and latch behaviors to be best observed.
- 8) Stop the display to increase the stability and then print the oscilloscope display.
- 9) Perform the POWER DOWN sequence on the FPGA board.
- 10) Remove the two blue jumper leads that you installed in steps 2 and 3.
- 11) Answer following questions.

1. Describe the behavior of output BAR1_1 with respect to input CLK0. Is this the expected behavior?
2. Describe the behavior of the output BAR1_2 with respect to input CLK0. Explain this behavior and why it differs from the behavior on BAR1_1 in terms of the concept of transparency of the latch.
3. Is the output of the D latch for $ENA = 1$ dependent on or independent of circuit delay? In terms of what you observe and what you believe would occur for different delays around the loop from Q back to D, explain your answer.
4. BONUS QUESTION: While observing the oscilloscope, place your fingers tightly on the lead going from BAR1_2 to BAR1_4. Can you explain the changes you see?

12) Include the printout and the answers to the questions in the Report for Experiment 4 as indicated.

TESTING THE CASCADED BCD COUNTERS

TESTING SETUP

- 1) Open your BCD_CNT project, and open the top-level design sheet that you used for simulation (CNTR_TEST.BDF).
- 2) You will need to edit this sheet to use the input/output connections shown in Figure 3 on the following pages. Be sure that you use the proper input/output net names.
- 3) We want to use the BCD decoders that you designed previously so we can view the BCD counter output on the LED displays. To do this we just use the Symbol tool, and browse to your Lab 3 directory and select bcd_7seg.bsf. (If the symbol file does not exist, create one using **File→Open**, navigate to your BCD decoder schematic (bcd_7seg.bdf), and **Open** it. Then, select **File→Create/Update→Create Symbol Files for Current File**.)
- 4) Add two of your BCD-to-7segment decoders so that the counters will drive the LED displays. Note that the LED displays on the FPGA board are active high, so your decoder will be able to drive them correctly.
- 5) Implement your design. Verify that the actual pin assignment was correct by viewing the Pin-Out File.

CHECKPOINT: Show your instructor your schematic and your Pin-Out File verifying the correct pin assignments.

PERFORMING TESTS AND RECORDING RESULTS

- 1) Perform the POWER UP sequence.
- 2) Download your bcd_cnt.sof file to the prototyping board.
- 3) Verify the proper operation of your circuit by changing the values of S1_1 (CE), S1_2 (CLR), and pressing PB1 to clock the circuit. Negating CE should inhibit the counter, while asserting CLR should cause an immediate reset of the counter. Observe the output of the counters on the two LED displays on the FPGA board to verify their operation. Do any necessary debugging and revisions to get an operational counter.

CHECKPOINT: Have your instructor observe the operation of your counters.

- 4) Answer following questions.

4. Clock routing is a critical issue in synchronous design. What problems could arise if the clock signal was delayed as it fanned out to the various flip-flops in our synchronous circuit?
5. In the design of the BCD counter, we did not even include the states 10-15. Was this a bad idea? Explain why or why not.

TESTING THE DIVIDE-BY-2000 COUNTER

TESTING SETUP

- 1) Perform the POWER DOWN sequence.
- 2) Remove the blue lead you installed previously.
- 3) Connect the oscilloscope to observe the clock input (CLKIN) on D0 and the counter output (BAR1_1) on D1. Trigger the oscilloscope on the counter output.

PERFORMING TESTS AND RECORDING RESULTS

- 1) Perform the POWER UP sequence.
- 2) Download your div2000.sof file to the prototyping board.
- 3) Obtain a suitable display on the oscilloscope, and use the cursors to accurately measure the period of counter output. Print the screen for submission with your report.

CHECKPOINT: Have your instructor observe the operation of your counter.

4-3 REPORT

Your individual report on 8.5 by 11 inch pages should follow the report format available on the course web page and consist of:

- 1) A cover with the experiment number, experiment name, your name, name(s) of other team member(s), and your instructor's name, your lab section, the number on the equipment tray used, and name on the PC used.
- 2) Description of Laboratory Exercise and Conclusions as shown in the report format,
- 3) Your individual answers to all of the questions given in the green-bordered boxes (online version has color), and
- 4) Copies of all printouts for the lab properly annotated and in the order produced.
 - a. printouts for the Latches and Flip-Flops portion of the experiment.
 - b. printout of the Divide-By-2000 counter test.

REFERENCES

1. MANO, M. M AND C. R. KIME. *Logic and Computer Design Fundamentals*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2003.

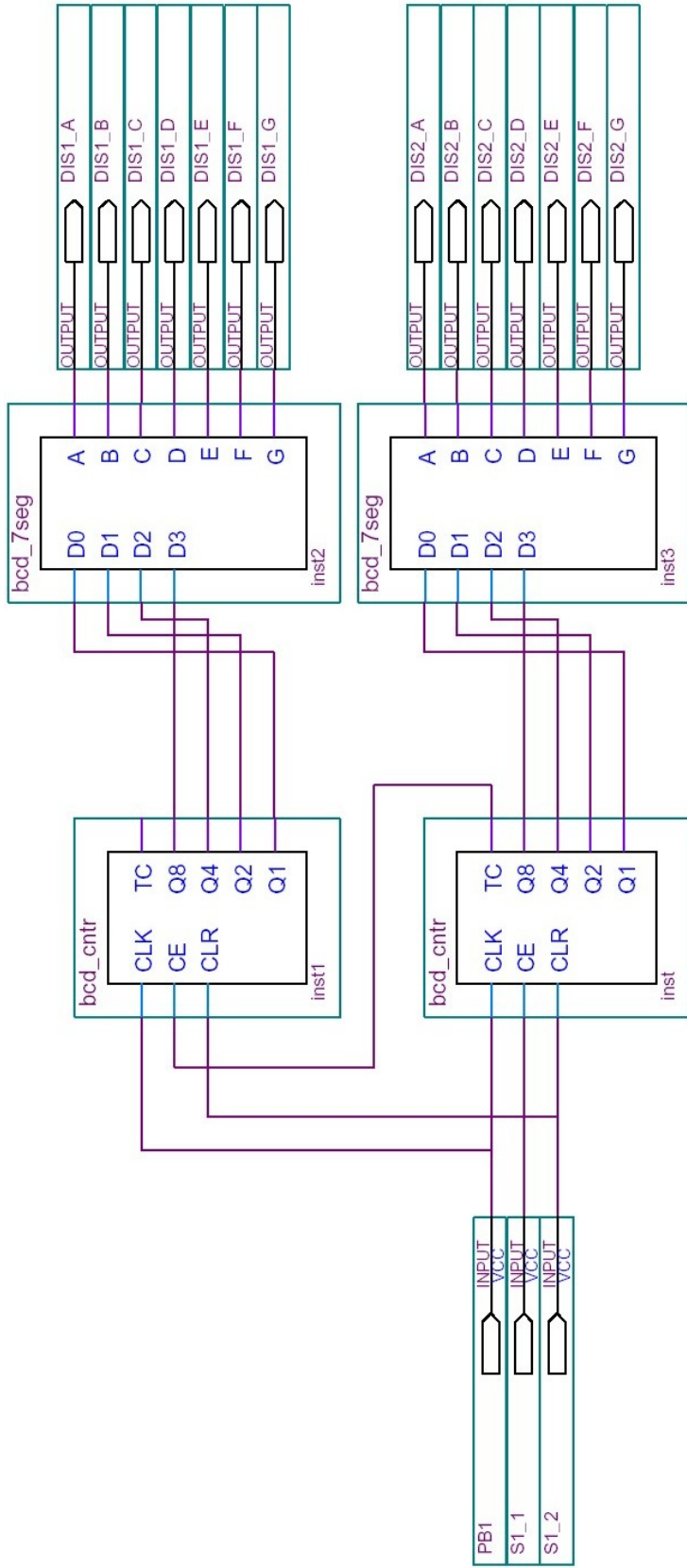


Figure 3 In-Lab Test Set-up

BCD COUNTER STATE TABLES AND INPUT VALUES - SUBMIT WITH PRELAB REPORT

<u>Present State</u>				<u>Inputs</u>		<u>Next State</u>				<u>Output</u>	<u>Flip-Flop Inputs</u>			
Q8	Q4	Q2	Q1	CE	CLR	Q8	Q4	Q2	Q1	TC	D8	D4	D2	D1
0	0	0	0	1	0									
0	0	0	1	1	0									
0	0	1	0	1	0									
0	0	1	1	1	0									
0	1	0	0	1	0									
0	1	0	1	1	0									
0	1	1	0	1	0									
0	1	1	1	1	0									
1	0	0	0	1	0									
1	0	0	1	1	0									
X	X	X	X	0	0									
X	X	X	X	X	1									

KARNAUGH MAPS FOR D FLIP-FLOP INPUTS - SUBMIT WITH PRELAB REPORT

K-map for FF Input D8

	$Q_2 Q_1$	00	01	11	10
$Q_8 Q_4$	00				
01					
11					
10					

K-map for FF Input D4

	$Q_2 Q_1$	00	01	11	10
$Q_8 Q_4$	00				
01					
11					
10					

K-map for FF Input D2

	$Q_2 Q_1$	00	01	11	10
$Q_8 Q_4$	00				
01					
11					
10					

K-map for FF Input D1

	$Q_2 Q_1$	00	01	11	10
$Q_8 Q_4$	00				
01					
11					
10					

K-map for Output TC

	$Q_2 Q_1$	00	01	11	10
$Q_8 Q_4$	00				
01					
11					
10					