ECE 501 Digital Systems Laboratory
Experiment 11 – Register Files

Seth So
Lab Partner: Martin Klena
Station 2

Date Performed: 3-26-18 to 3-27-18
Date Written: 3-28-18

....

Introduction:

Memory is an integral part of many electronic devices. Registers are used to store local, temporary memory that will be quickly processed and rewritten over. The goal of Laboratory 11 was to use various integrated circuit chips provided in the course lab kit to construct a 4x4 register file that has the ability to write to and read four bits, or one hexadecimal character, from four different addresses and display them on a seven segment LED. In order to assist with design, a parts list was given, as well as a skeleton structure, shown below in *Figure 1*:

Figure 1: 4x4 Register File Skeleton Design

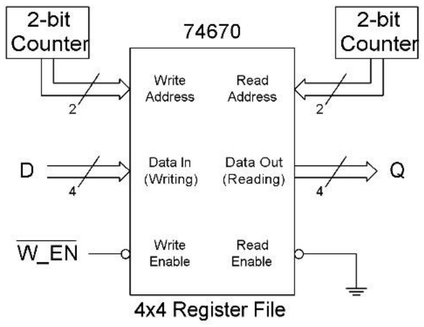


Figure 1 also acts as design specifications: The addressing should be handled by two different counters, the 74LS670 IC chip should process four bits of data both in and out and Write Enable should be enabled when a button is pressed, while Read Enable should always be enabled.

After design in Altera Quartus II, the circuit was physically made on the proto-boards, with the LED outputs being 7-segment LED displays. See the *References* section to see how the LED outputs were translated to display form (note that the "old set" of LEDs was used).

The following report details the procedure followed and results obtained while making the circuit.

Part I: Quartus Simulation

Purpose:

The purpose of part 1 was to use knowledge of digital logic to design the schematic for a 4x4 register file with the design specifications listed in the Introduction. The designing was to be done in Altera Quartus II software on the Graphic Editor, so that functionality of the design could be confirmed in the Waveform Editor before actual construction of the circuit.

Procedure:

1. The first step in designing the register file was to examine and understand the different components that would be used to make the register file. In addition to the skeleton design provided in *Figure 1,* a components list was also provided, shown in *Figure 2* in

the results section. Using this parts list and *Figure 1*, conceptual planning was made for what parts would be connected to what to achieve a functional design.

2. With the basic understanding and initial planning, the circuit was designed in Quartus II. Several important notes about connections:
   a. On the 74LS670 chip, GWN and GRN are the enable pins, which are low-true, meaning that GRN should be grounded for Read Enable to always be enabled, and GWN should have an inverted switch as its trigger to toggle Write Enable on and off.
   b. On the 74LS193 chip, DN and LDN are forced high so that the counters function as up counters. Inputs A-D and outputs CON, BON, $Q_{2,}$ and Q_3 would not be used and so were left open. Unlike previous chips, CLR was not low-true, so it was grounded to avoid constantly clearing the counter. Only Q_0 and Q_1 were used to make up the four addresses.
   c. On the 7-segement LED display, Dummy output had to be connected to an output pin to the simulation to correctly read the display.
   d. On the 74LS247 chip, BIN, RBIN, and LTN had to be forced HIGH for correct operation.

   See *Figure 3* in the results section for the final diagram schematic.

3. With the schematic built in the Graphic Editor, it was simulated in the Waveform Editor to test function of the 7-segment display. All periods were appropriately based off an inputted clock period of 100μs to cover all combinations of inputs $D_3D_2D_1D_{0,}$ $Q_3Q_2Q_1Q_{0,}$ Read Enable (grounded because it is low true), and Write Enable. The resulting outputs A-G were then observed to verify that the LED display was outputting the correct displays. See this waveform in *Figure 4* of the results section.

4. With verification that all inputs and connections were correct, the reading and writing functionality of the register were tested. *Figure 5* below details a simple procedure that sampled various conditions and outputs to verify that the expected results were observed.

Figure 5: Read and Write Functionality Test

Step:	*Action:*	*Step:*	*Action:*
0	Reset	*6*	Write 15 to address 1
1	Write 0 to address 1	*7*	Read address 1
2	Read address 1	*8*	Read address 2
3	Write 1 to address 2	*9*	Read address 3
4	Write 3 to address 3	*10*	Read address 0
5	Write 7 to address 1	*11*	Write 4 to address 1 and Read

Each step was performed sequentially by creating a custom clock signal that would trigger according to this procedure. *Figure 6* below in the results section shows the output waveform. This procedure was designed to test the functions of writing to each register address, reading stored values, and overwriting values.

Results:

Figure 2: Provided Parts List

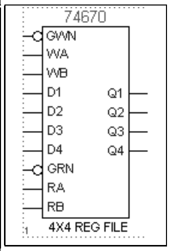
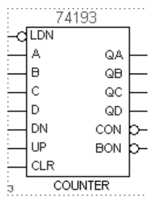
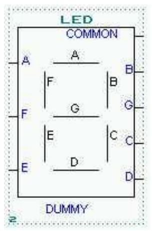
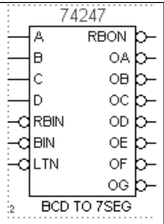
 74670 GWN WA WB D1 D2 D3 D4 GRN RA RB Q1 Q2 Q3 Q4 4X4 REG FILE	74LS670: 4 x 4 register file. This is the register chip itself. D1-D4 are the inputs (writing) and Q1-Q4 are the outputs (reading). [W_b,W_a] select the addresses for writing, and [R_b,R_a] select the address for reading. GWN and GRN are low-true inputs for write-enable and read-enable, respectively.
 74193 LDN A B C D DN UP CLR QA QB QC QD CON BON COUNTER	74LS193: 4 bit synchronous up/down counter (2 copies). These are counters that will be used to select addresses, one for reading and the other for writing. Each should be configured as an up counter, by forcing the DN and LDN inputs high. Inputs A-D and outputs CON and BON will not be used. The CLR input must be low for the counter to count. In this configuration, the counter will count up on the rising edge of the UP input. Finally, since there are only 4 addresses, we will use only the two lowest-order output bits [Q_b,Q_a].
	7-segment LED display (2 copies). These are used to display the data being written to or read from the register. The COMMON+ input should be connected to +5V, and a given segment is illuminated by setting the associated pin to a **low** voltage. **The DUMMY output must be connected to an output pin in order for the device to work.** You can obtain the Quartus II files for this display from the CoE server. The files you need are N:CoE501/led.sym and N:CoE501/led.gdf ***If you have problems with the simulation, remove these devices from your schematic.***
 74247 A B C D RBIN BIN LTN RBON OA OB OC OD OE OF OG BCD TO 7SEG	74LS247: 7-segment display driver (2 copies). These are decoders that take a 4-bit input and generate the appropriate signals for the 7-segment display. The BIN, RBIN, and LTN inputs should all be forced high to cause the decoder to operate. (You should have output ports from these display drivers to show that the LEDs work)

Figure 2 above is the provided parts list. The key information is summarized in step 2 of the procedure above. From step 1 of the procedure, a brief outline of how the parts worked together was formed: two 74LS193 up-counters would be used to output select lines for Read address and Write address on the 74LS670 file register chip. Based on the address, the 74LS670 file register chip would then output read and/or write instructions to their respective 74LS247 7-segment display drivers. These in turn would decode the instructions and output voltages to the correct pins the 7-segment LED display, making the proper combination of LEDs.

Figure 3: Final Schematic of 4x4 Register File

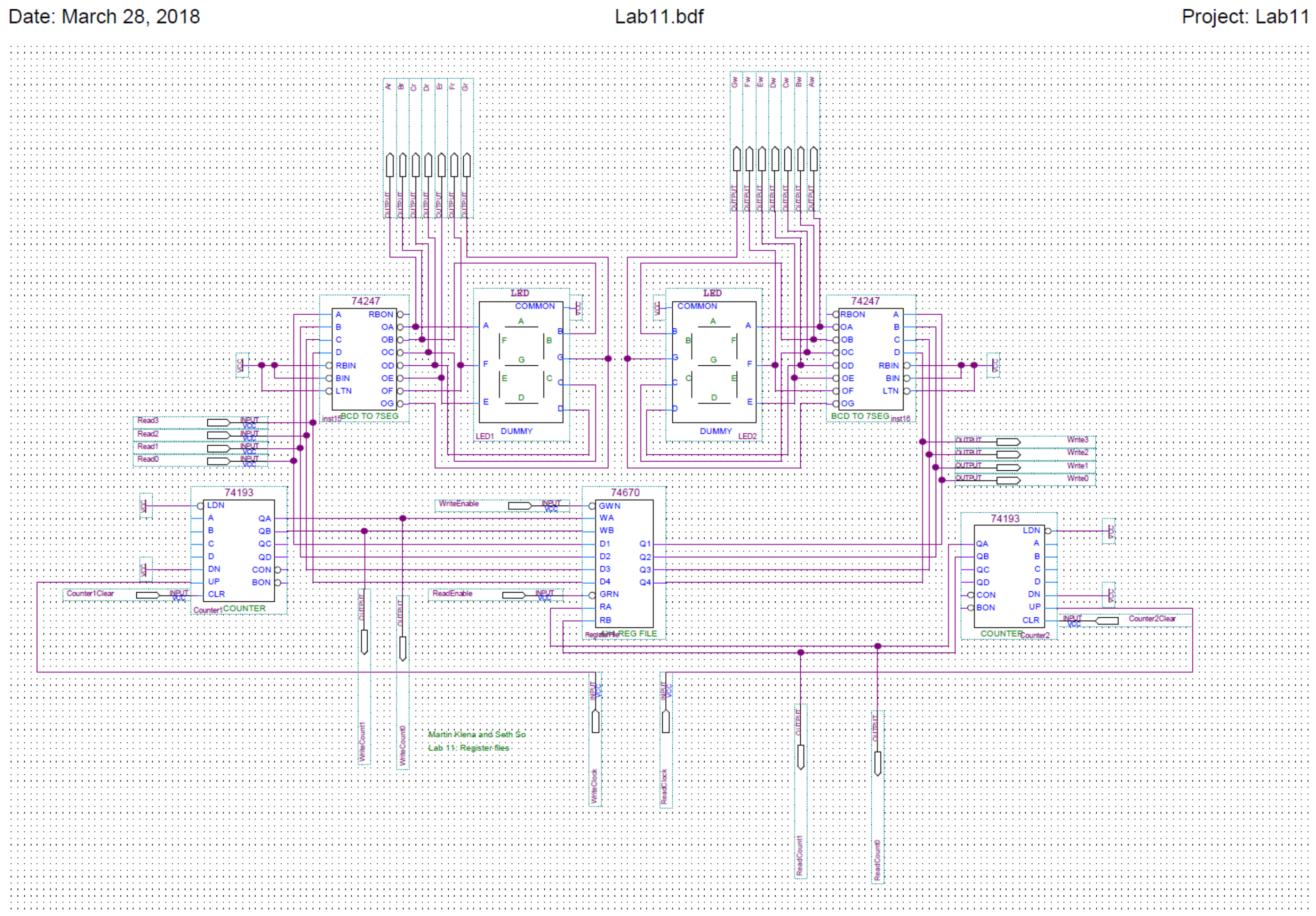


Figure 3 above displays the final design schematic for the 4x4 register file. Following the conceptual design that was planned before, first the 74LS193 chips were added to serve as counters that made the read and write address select lines. The pins specified in the parts list were set HIGH, the pins that would be used as outputs (QA and QB) were identified, and the two were connected to synchronised clock inputs. Next the 74LS670 register file chip was added. Read and Write Enable were controllled manually with inputs, and the address select lines from the counters were connected to the proper pins. Next, the 74LS247 LED display driver chips were added so that the Read ($Q_3Q_2Q_1Q_0$) and Write ($D_3D_2D_1D_0$) data outputs on the 74LS670 could be decoded as LED outputs. Again, as per the parts sheet, RBIN, BIN, and LTN were all driven HIGH. Finally, the 74LS247 outputs were connected to the respective pins on the 7-segment LED displays.

Outputs lines were connected to each data line so that they could be viewed in the Waveform Editor for easier debugging and analysis.

Figure 4: Waveform Verification of 7-Segment LED Display Outputs

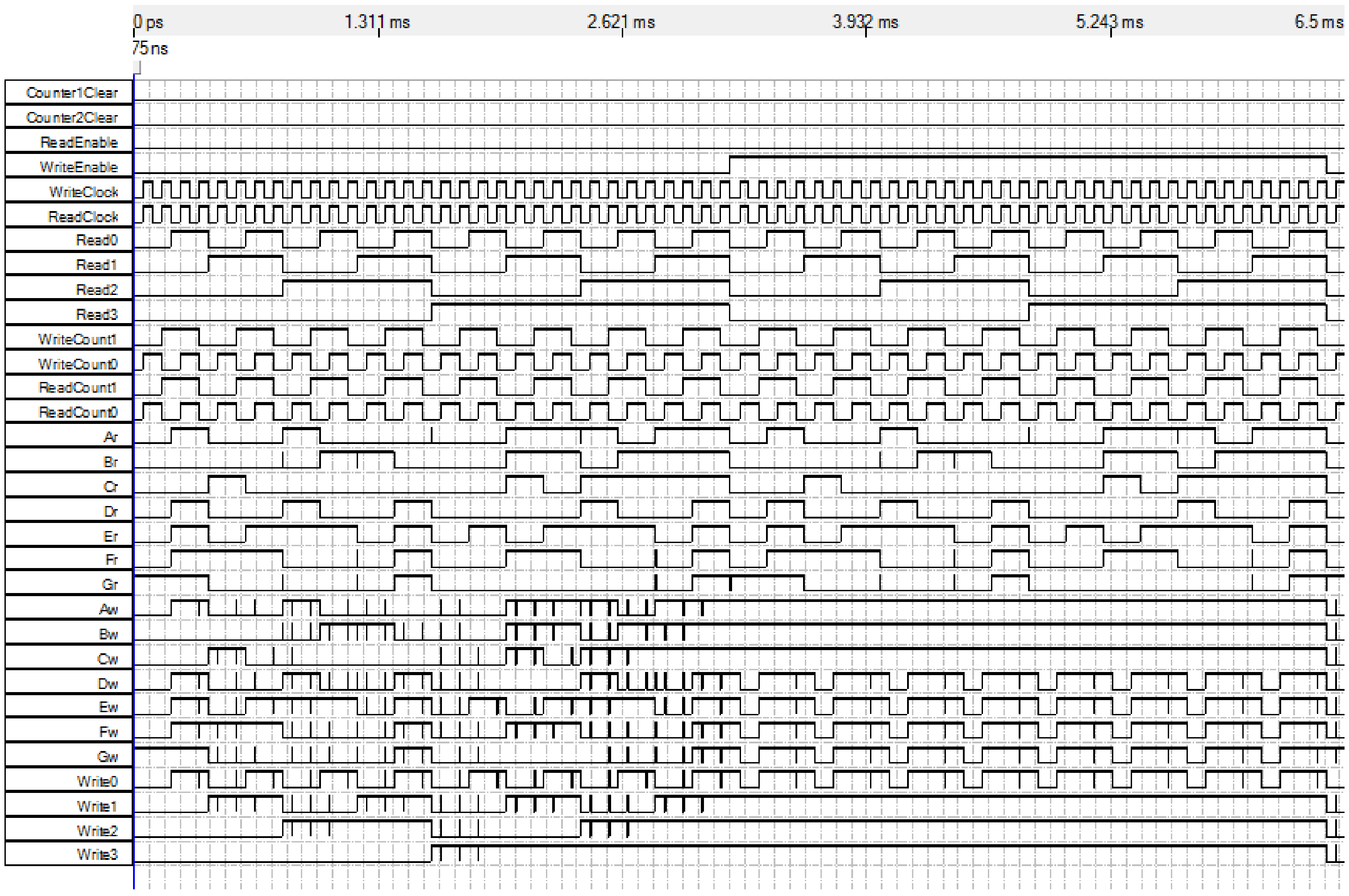


Figure 4 above displays the output waveform generated from simulating the schematic in the Waveform Editor. Describing the signals from top to bottom: the clear signals were always set LOW so that they would not interfere with the counter signals. Read Enable and Write Enable were true-low signals, so Read Enable was always enabled, while Write Enable was disabled for the second half of the full period tested. Read and Write clocks were synchronised clock signals on a 100µs period connected to the counters to the 74L193 counters to simulate the button being pressed. Note that all changes were rising edge triggered. $Read_{3-0}$ and $Write_{3-0}$ were the LED instructions passed from the 74LS670 to the 72LS247 chips. Write $Count_{1-0}$ and Read $Count_{1-0}$ were the the select lines into the 74LS670. $[A\text{-}G]_r$ and $[A\text{-}G]_w$ were the LEDs that would be lit up on the display.

$[A\text{-}G]_r$ and $[A\text{-}G]_w$ each combine with the other LED segments in their display to show 0-F with each $Read_0$ or $Write_0$ cycle, with the associated representations of A-F shown in the *References* section. The display is able to succesfully display all combintions 0x0 to 0xF for the correct Read ($Q_3Q_2Q_1Q_0$) and Write ($D_3D_2D_1D_0$) inputs. The Write display also stops updating when Write Enable is disable, as expected. Note that each LED segment is true-low.

Figure 6: Waveform Results of Functionality Procedure

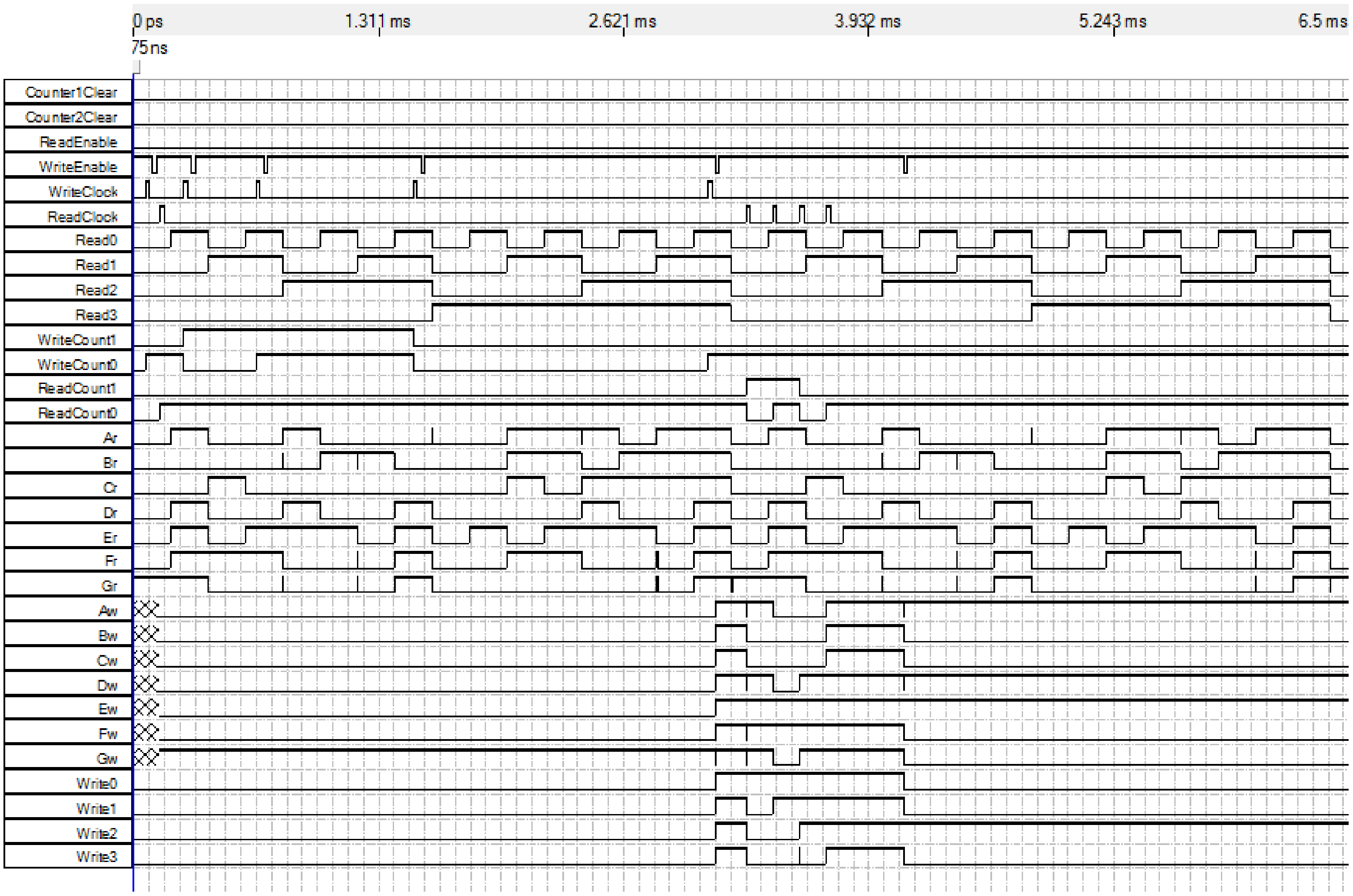


Figure 6 above displays the results when the schematic was simulated with the Waveform Editor, with the clock signals customized to enact the procedure stated in *Figure 5*. Note, each time division in the figure is 75µs, and a HIGH signal for an LED means that it is off, LOW meaning it is on. At **step 0**, t = 0s, 0 is shown on the Read display, and nothing is shown on the Write display. At **step 1**, t = 75µs, 0 is written to address 1, so Read Display shows 0 but Write display still shows nothing. At **step 2**, t = 150µs, address 1 is read, so both Read and Write displays show 0. At **step 3**, t = 300µs, 1 is written to address 2, so Read display now shows 1 but Write display still shows 0. At **step 4**, t = 600µs, 3 is written to address 3, so Read display now shows 3 but Write display still shows 0. At **step 5**, t = 1.45ms, 7 is written to address 0, so Read display now shows 7 but Write display still shows 0. At **step 6**, t = 2.9ms, 15 is written to address 1, so Read display now shows nothing (F is displayed as no lights, see *References*) but

Write display also displays nothing since address 1 is still being read. Note now that address 1 has been written over from 0 to 15. At **step 7**, t = 3.0ms, address 2 is read, so Read display shows 0 and Write displays shows 1. At **step 8**, t = 3.15ms, address 3 is read, so Read display shows 1 and Write displays shows 3. At **step 9**, t = 3.3ms, address 0 is read, so Read display shows 1 and Write displays shows 7. At **step 10**, t = 3.45ms, address 1 is read, so Read display shows 2 and Write displays shows nothing (15). At **step 11**, t = 4ms, 4 is written to address 1 and then address 1 is read, so Read display shows 4 and Write displays shows 4. The schematic simulation acted exactly as intended.

Conclusion:

The digital design and simulation of the 4x4 register file was a success. All schematic connections are in the right place, the 7-segment display displays the correct hex number per input, and all results were as expected for the functionality test. At this point, the circuit was ready to be built.

Part II: Proto-Board Construction

Purpose:

The purpose of part 2 was to realize the digital schematic made in Quartus II on the proto-boards. The design would again be tested twice for functionality – both manually and with the Logic Analyzer. This circuit will be saved for future use for the next laboratory in making the ALU.

Procedure:

1. The schematic in *Figure 3* was constructed on the protoboards with the addition of several extra components. Rather than have clock signals, inverted debounced buttons from laboratory 7 (see *References*) were constructed to control the counters and Read Enable. Additionally, four DIP switches with 1kΩ pull-up resistors were used to control the Read inputs $Q_3Q_2Q_1Q_0$. Finally, two LEDs with 150Ω current limiting resistors were added onto each of the Counter outputs (address select lines) so that the Read and Write addresses could be visibly kept track of.

2. The now built circuit board was run through the functionality procedure described in *Figure 5*. The test was extremely successful, and the results have bee compiled into a chart shown in *Figure 7* in the results section below.

3. Once the manual test of the circuit was complete, all relevant circuit outputs were connected to the logic analyzer. Using the logic analyzer for a more accurate and reliable measurements, the circuit was once again run through the functionality procedure, but this time images were taken at each step of the logic values in the circuit.

   Because there were 28 channels being used, the Logic analyzer filled up with data too quickly from live feed rates. To circumvent this problem, the logic analyzer was set to trigger on click and then sample only once. This was repeated for each step in the functionality procedure.

Another quick fix to make the logic analyzer output the correct values was adjusting the logic threshold to 1.25V down from 2. For this design, this allowed the logic analyzer to distinguish between HIGHs and LOWs accurately. See *Figure 8$_{1\text{-}12}$* below for full results.

Results:

Figure 7: Proto-Board Circuit Functionality Test Results

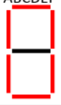
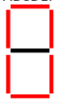
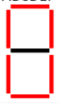
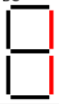
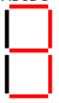
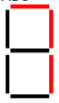
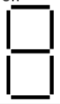
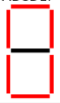
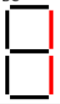
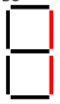
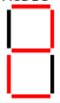
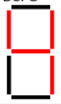
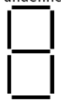
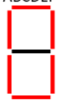
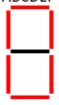
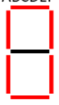
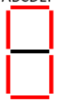
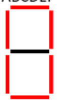
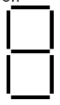
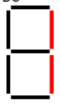
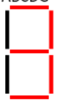
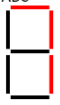
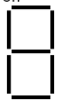
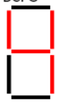
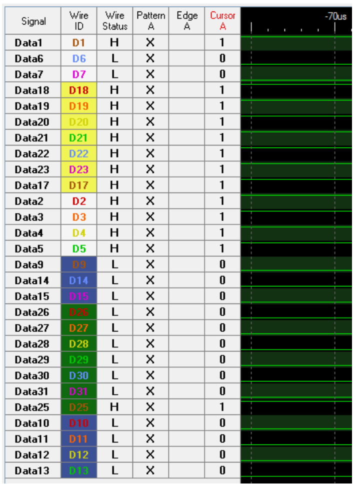
Step:	0	1	2	3	4	5	6	7	8	9	10	11
Write Enabled?	No	Yes	No	Yes	Yes	Yes	Yes	No	No	No	No	Yes
Read 3:0	0000	0000	000o	0001	0011	0111	1111	0000	0001	0001	0010	0100
Read 3:0 (Hex)	0	0	1	1	3	7	F	0	1	1	2	4
A_r: G_r (visual)	ABCDEF 	ABCDEF 	ABCDEF 	BC 	ABCDG 	ABC 	off 	ABCDEF 	BC 	BC 	ACDEG 	BCFG 
Write Address W_BW_A	00	01	01	10	11	00	01	01	01	01	01	01
Read Address R_BR_A	00	00	01	01	01	01	01	10	11	00	01	01
A_w: G_w (visual)	undefined 	ABCDEF 	ABCDEF 	ABCDEF 	ABCDEF 	ABCDEF 	off 	BC 	ABCDG 	ABC 	off 	BCFG 
Write 3:0	0000	0000	0000	0000	0000	0000	1111	0001	0011	0111	1111	0100
Write 3:0 (Hex)	0	0	0	0	0	0	15	1	3	7	15	4

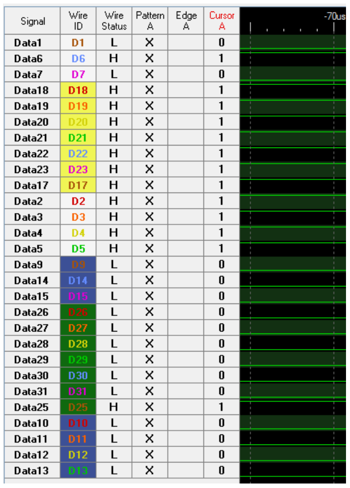
Figure 7 above compiles the results of the how the physical circuit did with the functionality test (see *Figure 5)*. The table displays the visual result of the LED's as well as all the corresponding address and outputs. The results, except for the clock timings, are identical to those of the Waveform seen in *Figure 6*. Note that Read3:0 corresponds to $Q_3Q_2Q_1Q_0$ and Write3:0 corresponds to $D_3D_2D_1D_0$. Recall that for $Q_3Q_2Q_1Q_0$ = 1111 = 0xF, the LED configuration is all LEDs off. The physical circuit was able to replicate the results from the waveform diagram test.

Figure 8: Logic Analyzer Functionality Test Results



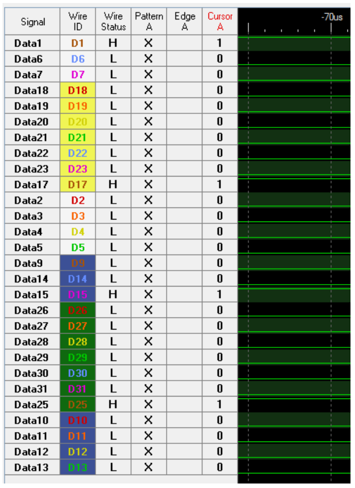
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	H	X		1
Data6	D6	L	X		0
Data7	D7	L	X		0
Data18	D18	H	X		1
Data19	D19	H	X		1
Data20	D20	H	X		1
Data21	D21	H	X		1
Data22	D22	H	X		1
Data23	D23	H	X		1
Data17	D17	H	X		1
Data2	D2	H	X		1
Data3	D3	H	X		1
Data4	D4	H	X		1
Data5	D5	H	X		1
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	L	X		0
Data26	D26	L	X		0
Data27	D27	L	X		0
Data28	D28	L	X		0
Data29	D29	L	X		0
Data30	D30	L	X		0
Data31	D31	L	X		0
Data25	D25	H	X		1
Data10	D10	L	X		0
Data11	D11	L	X		0
Data12	D12	L	X		0
Data13	D13	L	X		0

Step 0: Reset



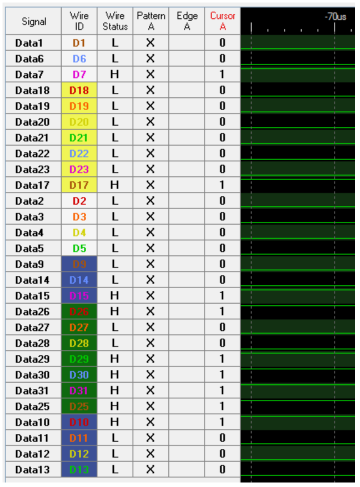
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	L	X		0
Data6	D6	H	X		1
Data7	D7	L	X		0
Data18	D18	H	X		1
Data19	D19	H	X		1
Data20	D20	H	X		1
Data21	D21	H	X		1
Data22	D22	H	X		1
Data23	D23	H	X		1
Data17	D17	H	X		1
Data2	D2	H	X		1
Data3	D3	H	X		1
Data4	D4	H	X		1
Data5	D5	H	X		1
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	L	X		0
Data26	D26	L	X		0
Data27	D27	L	X		0
Data28	D28	L	X		0
Data29	D29	L	X		0
Data30	D30	L	X		0
Data31	D31	L	X		0
Data25	D25	H	X		1
Data10	D10	L	X		0
Data11	D11	L	X		0
Data12	D12	L	X		0
Data13	D13	L	X		0

Step 1: Write 0 to address 1



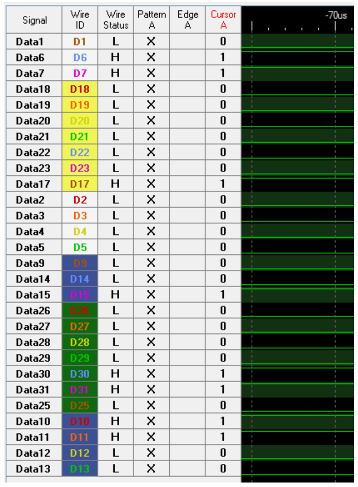
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	H	X		1
Data6	D6	L	X		0
Data7	D7	L	X		0
Data18	D18	L	X		0
Data19	D19	L	X		0
Data20	D20	L	X		0
Data21	D21	L	X		0
Data22	D22	L	X		0
Data23	D23	L	X		0
Data17	D17	H	X		1
Data2	D2	L	X		0
Data3	D3	L	X		0
Data4	D4	L	X		0
Data5	D5	L	X		0
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	H	X		1
Data26	D26	L	X		0
Data27	D27	L	X		0
Data28	D28	L	X		0
Data29	D29	L	X		0
Data30	D30	L	X		0
Data31	D31	L	X		0
Data25	D25	H	X		1
Data10	D10	L	X		0
Data11	D11	L	X		0
Data12	D12	L	X		0
Data13	D13	L	X		0

Step 2: Read address 1



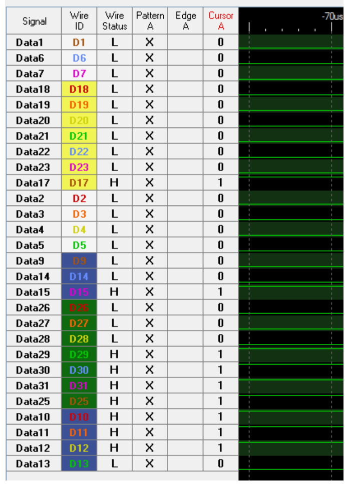
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	L	X		0
Data6	D6	L	X		0
Data7	D7	H	X		1
Data18	D18	L	X		0
Data19	D19	L	X		0
Data20	D20	L	X		0
Data21	D21	L	X		0
Data22	D22	L	X		0
Data23	D23	L	X		0
Data17	D17	H	X		1
Data2	D2	L	X		0
Data3	D3	L	X		0
Data4	D4	L	X		0
Data5	D5	L	X		0
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	H	X		1
Data26	D26	H	X		1
Data27	D27	L	X		0
Data28	D28	L	X		0
Data29	D29	H	X		1
Data30	D30	H	X		1
Data31	D31	H	X		1
Data25	D25	H	X		1
Data10	D10	H	X		1
Data11	D11	L	X		0
Data12	D12	L	X		0
Data13	D13	L	X		0

Step 3: Write 1 to address 2



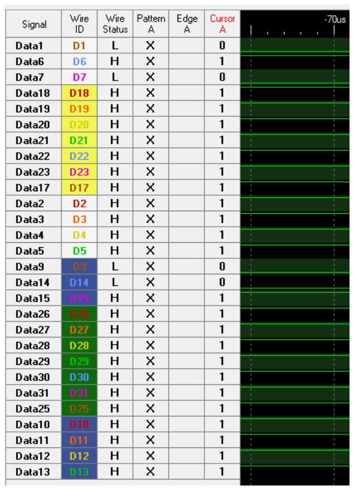
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	L	X		0
Data6	D6	H	X		1
Data7	D7	H	X		1
Data18	D18	L	X		0
Data19	D19	L	X		0
Data20	D20	L	X		0
Data21	D21	L	X		0
Data22	D22	L	X		0
Data23	D23	L	X		0
Data17	D17	H	X		1
Data2	D2	L	X		0
Data3	D3	L	X		0
Data4	D4	L	X		0
Data5	D5	L	X		0
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	H	X		1
Data26	D26	L	X		0
Data27	D27	L	X		0
Data28	D28	L	X		0
Data29	D29	L	X		0
Data30	D30	H	X		1
Data31	D31	H	X		1
Data25	D25	L	X		0
Data10	D10	H	X		1
Data11	D11	H	X		1
Data12	D12	L	X		0
Data13	D13	L	X		0

Step 4: Write 3 to address 3



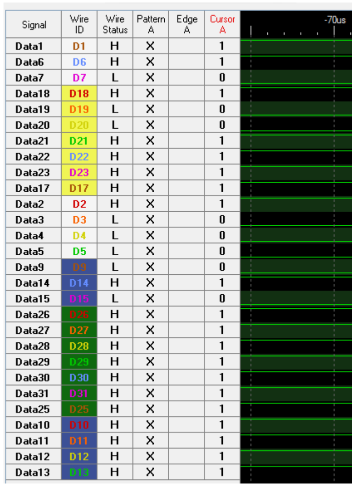
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	L	X		0
Data6	D6	L	X		0
Data7	D7	L	X		0
Data18	D18	L	X		0
Data19	D19	L	X		0
Data20	D20	L	X		0
Data21	D21	L	X		0
Data22	D22	L	X		0
Data23	D23	L	X		0
Data17	D17	H	X		1
Data2	D2	L	X		0
Data3	D3	L	X		0
Data4	D4	L	X		0
Data5	D5	L	X		0
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	H	X		1
Data26	D26	L	X		0
Data27	D27	L	X		0
Data28	D28	L	X		0
Data29	D29	H	X		1
Data30	D30	H	X		1
Data31	D31	H	X		1
Data25	D25	H	X		1
Data10	D10	H	X		1
Data11	D11	H	X		1
Data12	D12	H	X		1
Data13	D13	L	X		0

Step 5: Write 7 to address 0



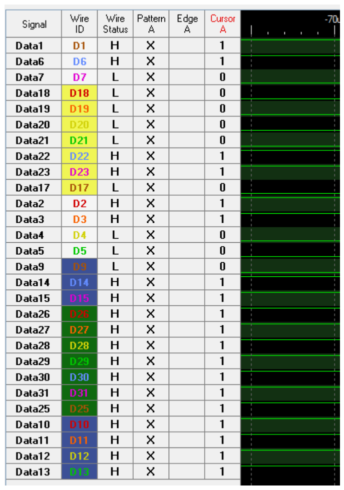
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	L	X		0
Data6	D6	H	X		1
Data7	D7	L	X		0
Data18	D18	H	X		1
Data19	D19	H	X		1
Data20	D20	H	X		1
Data21	D21	H	X		1
Data22	D22	H	X		1
Data23	D23	H	X		1
Data17	D17	H	X		1
Data2	D2	H	X		1
Data3	D3	H	X		1
Data4	D4	H	X		1
Data5	D5	H	X		1
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	H	X		1
Data26	D26	H	X		1
Data27	D27	H	X		1
Data28	D28	H	X		1
Data29	D29	H	X		1
Data30	D30	H	X		1
Data31	D31	H	X		1
Data25	D25	H	X		1
Data10	D10	H	X		1
Data11	D11	H	X		1
Data12	D12	H	X		1
Data13	D13	H	X		1

Step 6: Write 15 to address 1



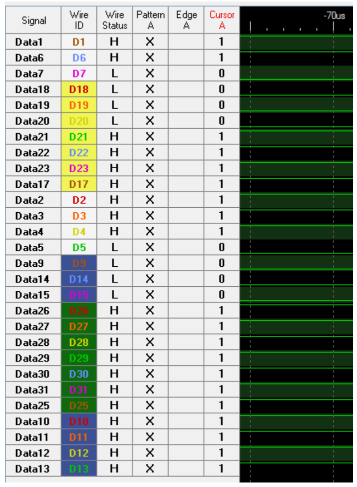
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	H	X		1
Data6	D6	H	X		1
Data7	D7	L	X		0
Data18	D18	H	X		1
Data19	D19	L	X		0
Data20	D20	L	X		0
Data21	D21	H	X		1
Data22	D22	H	X		1
Data23	D23	H	X		1
Data17	D17	H	X		1
Data2	D2	H	X		1
Data3	D3	L	X		0
Data4	D4	L	X		0
Data5	D5	L	X		0
Data9	D9	L	X		0
Data14	D14	H	X		1
Data15	D15	L	X		0
Data26	D26	H	X		1
Data27	D27	H	X		1
Data28	D28	H	X		1
Data29	D29	H	X		1
Data30	D30	H	X		1
Data31	D31	H	X		1
Data25	D25	H	X		1
Data10	D10	H	X		1
Data11	D11	H	X		1
Data12	D12	H	X		1
Data13	D13	H	X		1

Step 7: Read address 2



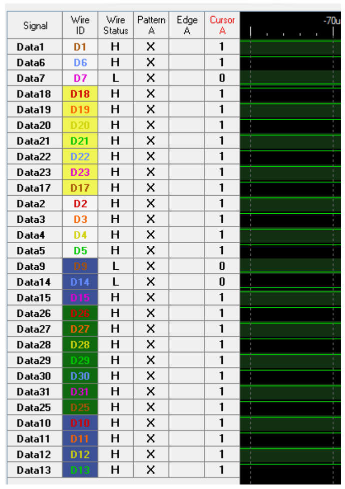
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	H	X		1
Data6	D6	H	X		1
Data7	D7	L	X		0
Data18	D18	L	X		0
Data19	D19	L	X		0
Data20	D20	L	X		0
Data21	D21	L	X		0
Data22	D22	H	X		1
Data23	D23	H	X		1
Data17	D17	L	X		0
Data2	D2	H	X		1
Data3	D3	H	X		1
Data4	D4	L	X		0
Data5	D5	L	X		0
Data9	D9	L	X		0
Data14	D14	H	X		1
Data15	D15	H	X		1
Data26	D26	H	X		1
Data27	D27	H	X		1
Data28	D28	H	X		1
Data29	D29	H	X		1
Data30	D30	H	X		1
Data31	D31	H	X		1
Data25	D25	H	X		1
Data10	D10	H	X		1
Data11	D11	H	X		1
Data12	D12	H	X		1
Data13	D13	H	X		1

Step 8: Read address 0



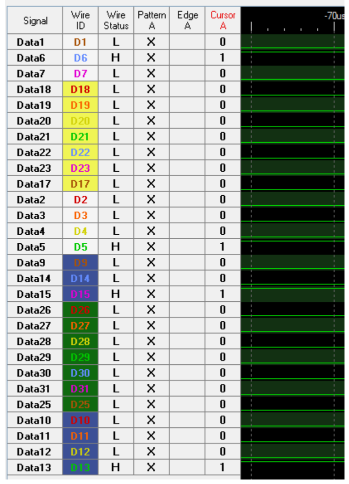
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	H	X		1
Data6	D6	H	X		1
Data7	D7	L	X		0
Data18	D18	L	X		0
Data19	D19	L	X		0
Data20	D20	L	X		0
Data21	D21	H	X		1
Data22	D22	H	X		1
Data23	D23	H	X		1
Data17	D17	H	X		1
Data2	D2	H	X		1
Data3	D3	H	X		1
Data4	D4	H	X		1
Data5	D5	L	X		0
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	L	X		0
Data26	D26	H	X		1
Data27	D27	H	X		1
Data28	D28	H	X		1
Data29	D29	H	X		1
Data30	D30	H	X		1
Data31	D31	H	X		1
Data25	D25	H	X		1
Data10	D10	H	X		1
Data11	D11	H	X		1
Data12	D12	H	X		1
Data13	D13	H	X		1

Step 9: Read address 0



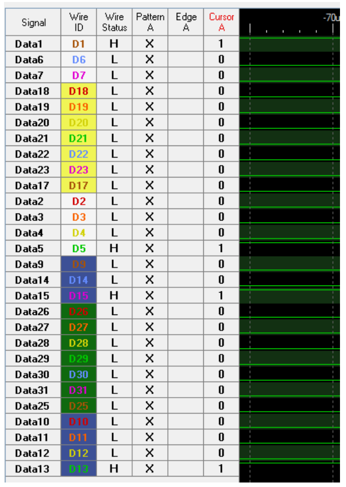
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	H	X		1
Data6	D6	H	X		1
Data7	D7	L	X		0
Data18	D18	H	X		1
Data19	D19	H	X		1
Data20	D20	H	X		1
Data21	D21	H	X		1
Data22	D22	H	X		1
Data23	D23	H	X		1
Data17	D17	H	X		1
Data2	D2	H	X		1
Data3	D3	H	X		1
Data4	D4	H	X		1
Data5	D5	H	X		1
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	H	X		1
Data26	D26	H	X		1
Data27	D27	H	X		1
Data28	D28	H	X		1
Data29	D29	H	X		1
Data30	D30	H	X		1
Data31	D31	H	X		1
Data25	D25	H	X		1
Data10	D10	H	X		1
Data11	D11	H	X		1
Data12	D12	H	X		1
Data13	D13	H	X		1

Step 10: Read address 1



Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	L	X		0
Data6	D6	H	X		1
Data7	D7	L	X		0
Data18	D18	L	X		0
Data19	D19	L	X		0
Data20	D20	L	X		0
Data21	D21	L	X		0
Data22	D22	L	X		0
Data23	D23	L	X		0
Data17	D17	L	X		0
Data2	D2	L	X		0
Data3	D3	L	X		0
Data4	D4	L	X		0
Data5	D5	H	X		1
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	H	X		1
Data26	D26	L	X		0
Data27	D27	L	X		0
Data28	D28	L	X		0
Data29	D29	L	X		0
Data30	D30	L	X		0
Data31	D31	L	X		0
Data25	D25	L	X		0
Data10	D10	L	X		0
Data11	D11	L	X		0
Data12	D12	L	X		0
Data13	D13	H	X		1

Step 11: Write 8 to address 1



Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data1	D1	H	X		1
Data6	D6	L	X		0
Data7	D7	L	X		0
Data18	D18	L	X		0
Data19	D19	L	X		0
Data20	D20	L	X		0
Data21	D21	L	X		0
Data22	D22	L	X		0
Data23	D23	L	X		0
Data17	D17	L	X		0
Data2	D2	L	X		0
Data3	D3	L	X		0
Data4	D4	L	X		0
Data5	D5	H	X		1
Data9	D9	L	X		0
Data14	D14	L	X		0
Data15	D15	H	X		1
Data26	D26	L	X		0
Data27	D27	L	X		0
Data28	D28	L	X		0
Data29	D29	L	X		0
Data30	D30	L	X		0
Data31	D31	L	X		0
Data25	D25	L	X		0
Data10	D10	L	X		0
Data11	D11	L	X		0
Data12	D12	L	X		0
Data13	D13	H	X		1

Step 12: Read address 1

Logic Analyzer Data Key:
(Top to Bottom)

Signal		Function
1	-	Write Enable
6,7	-	W_1W_0
17-23	-	$[A\text{-}G]_W$
2-5	-	$D_3D_2D_1D_0$
- - - - - - - - - -	- - - - -	- - - - - - - - - -
9	-	Read Enable
14,15	-	R_1R_0
25-31	-	$[A\text{-}G]_R$
10-13	-	$Q_3Q_2Q_1Q_0$

Figure 8 above displays the Logic Analyzer images taken after connecting the LogicPort Logic Analyzer to the circuit and carrying out the functionality test procedure. It is very important to note that for the logic analyzer, the test procedure was slightly modified in the end – rather than write and read a 4, an 8 was written and read to show use of all LEDs.

The signal was able to very clearly display the changes in values of each value measured. There were clear changes rom logic values of 1 and 0 between the different registers addresses, LED displays, enables, and counters. After much analysis and double checking, the results of the logic analyzer are the same as both the Waveform Editor digital functionality test and the circuit board manual functionality test (except for the last 8 written, which was deliberately done). Testing at all three points of the design procedure agree and confirm that the circuit works as designed.

Conclusion:

All results were as expected. The built circuit was able to perform as intended, producing the same outputs as those generated in the Quartus II simulation. The one major advantage of the Logic Analyzer was that it displayed all of the logic values of any relevant signal. Even though the results were all accurate, this would have made debugging much simpler, as the particular signal that was incorrect could be singled out and troubleshot.

Summary:

The goal of Laboratory 11 was to design and build a 4x4 register file to write and read 4-bit data and then display it to a 7-segment LED display. The design this time was once again first thought out, then built digitally, and then physically. Because the circuit had many functions to test, a test procedure was made to rigorously test the functionality of the circuit at every stage of the process to ensure that design and connections were entirely correct before moving onto the next stage. This helped prevent build up of error and find problems at their base cause.

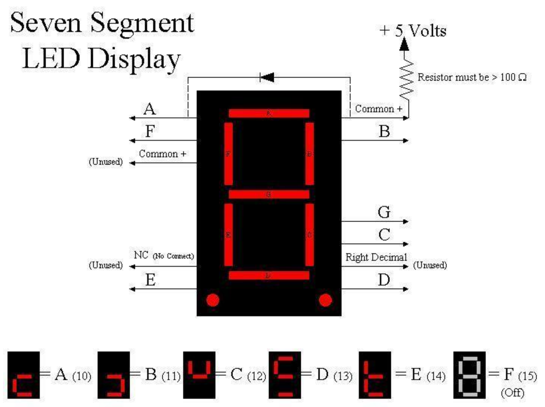
References:

74193 datasheet: https://courseweb.pitt.edu/bbcswebdav/pid-24291878-dt-content-rid-23464854_2/courses/2184_UPITT_ECE_0501_SEC1010/74193.pdf

74247 datasheet: https://courseweb.pitt.edu/bbcswebdav/pid-24291878-dt-content-rid-23464855_2/courses/2184_UPITT_ECE_0501_SEC1010/74247.pdf

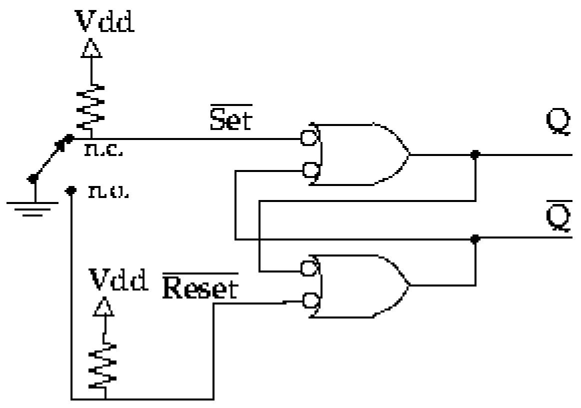
74670 datasheet: https://courseweb.pitt.edu/bbcswebdav/pid-24291878-dt-content-rid-23464856_2/courses/2184_UPITT_ECE_0501_SEC1010/74670.pdf

LED outputs to Display Conversion:



7 Segment Display symbols for Hexadecimal values A-F in old set of LEDs.

Debounce Switch Circuit:



Vdd
Set
n.c.
n.o.
Q
$\overline{Q}$
Vdd
Reset