ECE 501 Digital Systems Laboratory
Experiment 10 – Design of Finite State Machines

Seth So
Lab Partner: Martin Klena
Station 27

Date Performed: 3-21-18 to 3-24-18
Date Written: 3-24-18

…….

Introduction:

This laboratory was a set of four procedures to design, simulate, implement, and test a finite state machine for a synchronous, sequential circuit. The circuit must have two single-bit inputs A and B, and a single-bit output Z. Z will output a repeating sequence of digits based on the inputs A and B according to *Figure 1* below:

Figure 1: Required inputs and outputs of the finite state machine

AB (Input)	**Z** (Output Sequence)
00	0000000100000001…
01	0000011100000111…
10	0001111100011111…
11	0111111101111111…

Z should sequentially progress to the next bit in each sequence (shown in *Figure 1*) with each rising clock edge and loop back at the beginning. This clock edge must be set by a 555-timer circuit, which acts as the main clock according to which the rest of the circuit must act. This is what makes the timing synchronous and independent of different outputs throughout the circuit. The circuit must also use 7474 D flip-flops to store the states, and the number of logic gates and ICs needed should be minimized if possible.

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

Part I: Finite State Machine

Purpose:

The purpose of part 1 was to review synchronous design techniques to use in designing a finite state machine that satisfied the criteria listed in the Introduction. Methods from the 0132 Digital Logic course (i.e. K-map, state transition diagram, state assignment, and state transition table) were followed to design and optimize the logic of the circuit.

Procedure:

1. The first step in designing the finite state machine was choosing to make a Moore or Mealy machine. Moore machines are state-defined (output is tied to the state). They are simpler in concept, they but can become more complex than Mealy machines when many states are required. The output of Mealy machines is determined by current state as well as input, which helps to cut down on the total number of states needed. Therefore, a Mealy machine design was chosen since it would only require 8 states compared to 32 states needed by a Moore machine.

2. Using the Mealy design, a state transition diagram was created that would output the proper sequence for each combination of inputs (see *Figure 1*). Because there are 8 bits in each sequence, the design will have 8 states. Each clock cycle will move the current state

to the next state and output a bit of the sequence based off the inputs. See *Figure 2* in the results section for the diagram.

3. Next a state transition table was made from the state transition diagram. It shows current state, possible inputs (A and B), and the output (Z) and next state associated with each input. Additionally, the states were assigned 3-bit binary numbers (Current state = D_2 D_1 D_0 and Next state = D_2^+ D_1^+ D_0^+). This would allow for the derivation of the logic expressions using K-maps. The complete table is shown in *Figure 3* in the results section.

4. K-maps were then used on the transition table to find the minimized next state logic for next state bits D_2^+, D_1^+, and D_0^+ in terms of current state bits D_2, D_1, and D_0. The input bits were not required, because the machine was designed so that the current state would move onto the next state every clock edge regardless of the input. The results for each of these are documented in *Figure 4 a-c* in the results section.

5. K-maps were used to find the output bit Z. Here, Z was dependent on both current state and input, so a slightly more complicated five-variable K-map had to be computed to obtain the minimized output logic. The result of this K-map is displayed in *Figure 5* below in the results section.

6. From the final expressions for next state logic and output logic, the total number of gates and physical components needed for implementation was determined. The list is compiled in the table in *Figure 6*.

Results:

Figure 2: Finite state transition diagram (Mealy design)

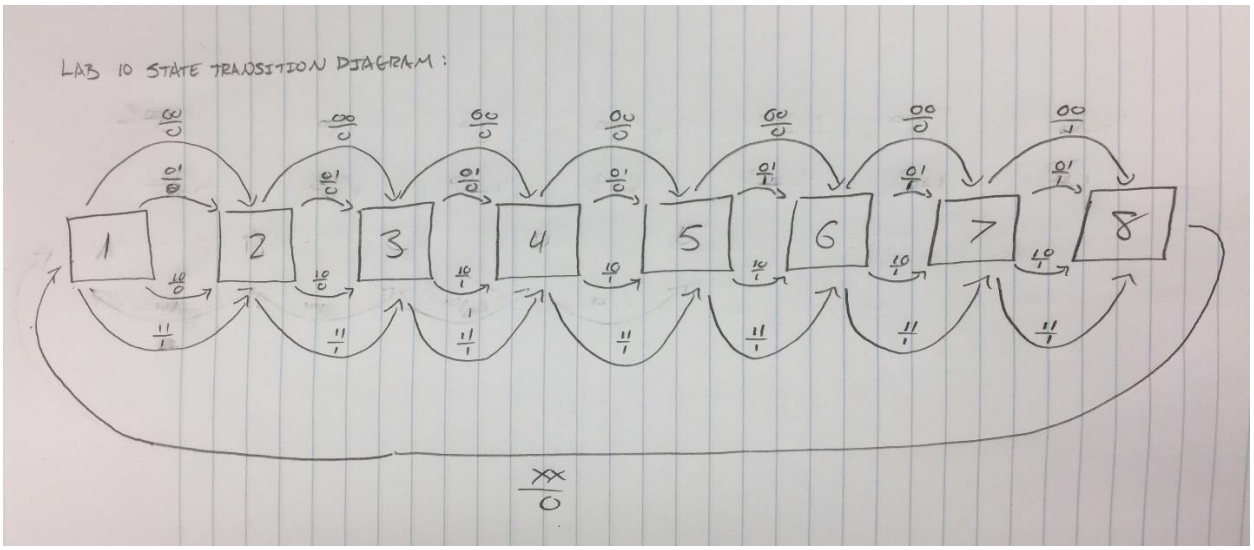


Figure 2 above displays the state transition diagram for the proposed finite machine design. There are 8 states, 1-8, for each of the 8 bits in the output sequence. This also corresponds to 8 clock cycles, as each state moves to the next on the rising edge of every clock cycle. The diagram follows normal Mealy machine convention – the arrows indicate the which state comes

next, and above each arrow is the output for given input ($\frac{Inputs}{Output} = \frac{AB}{Z}$). The output is dependent on current state and inputs, but next state is only dependent on current state. After reaching state 8 ($D_2D_1D_0 = 111$), the system will always output a 0 and loop back to state 1 on the next clock cycle. Because the 8 states are counted in binary, there are 3 state bits D_2, D_1, and D_0. Each of these state bits will be realized with a d flip-flop, calling for 3 d flip-flops.

With this Mealy design, only 8 states are needed as opposed to 32 needed for an equivalent Moore machine. Additionally, the sequence does not start over if the input is changed mid-cycle, but rather it picks up at the same index of the new sequence. The behavior of the circuit is a 3-bit synchronous counter (counting from state to state) that outputs different values depending on the current state and input. All changes occur on rising clock edges.

Figure 3: State transition diagram and state assignments

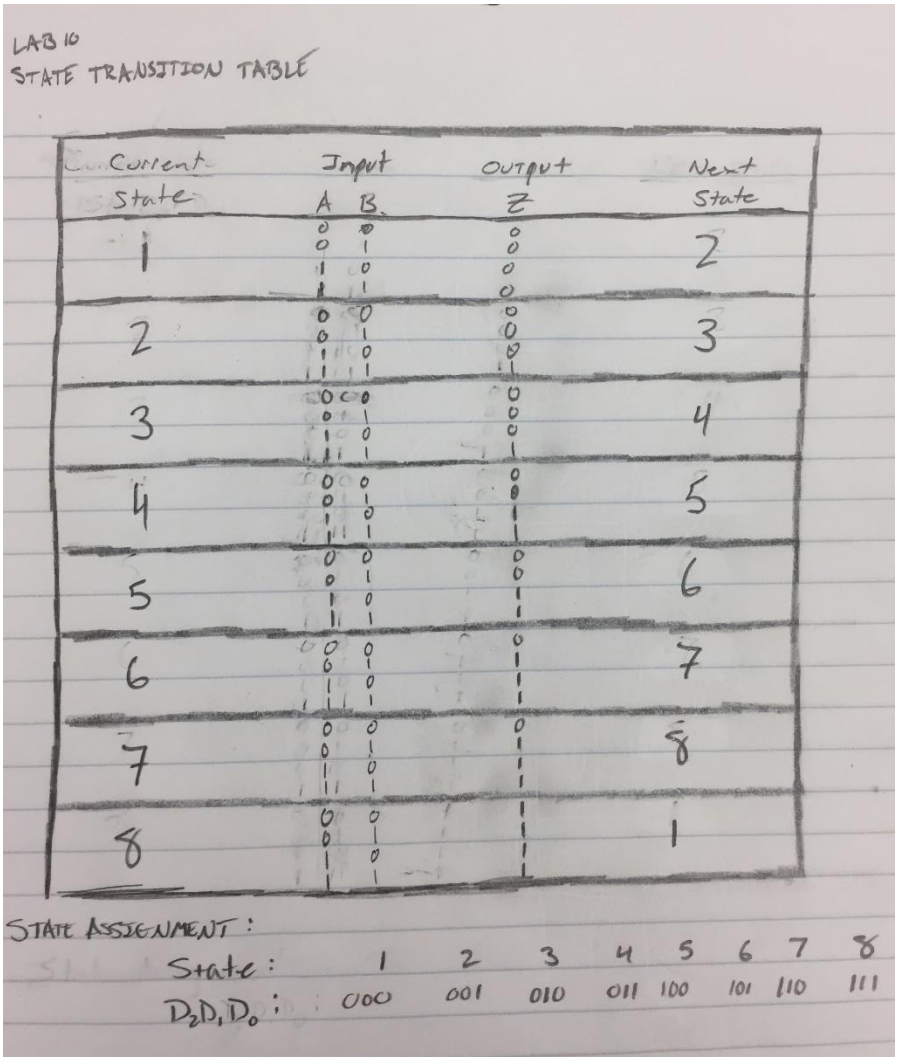


Figure 3 above displays the state transition table created from the state transition diagram. Each

row tells what the current state is, and the 4 possible inputs it can receive, the outputs that would results from those inputs, and the next state (independent of input). At the bottom is the annotated state assignment: state 1 becomes $D_2D_1D_0 = 000$, state 2 becomes $D_2D_1D_0 = 001$ etc…. This same convention holds for the next state logic: next state 1 becomes $D_2^+D_1^+D_0^+ = 000$, next state 2 becomes $D_2^+D_1^+D_0^+ = 001$, etc…. Replacing the state values with these, the state transition table becomes a full truth table with inputs D_2, D_1, D_0, A, and B and outputs Z, D_2^+, D_1^+, and D_0^+. The logic required for the D_2^+, D_1^+, and D_0^+ outputs will determine the inputs into the flip-flops. The upcoming *Figures 4a-c* and *Figure 5* will minimize this logic.

Figure 4a: K-map logic derivation and minimization for D_0^+

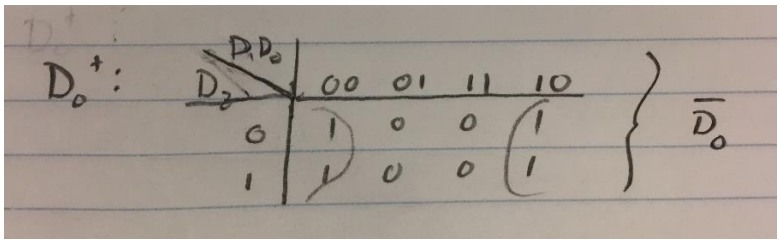


Figure 4a above shows that the logic expression for D_0^+ requires one NOT gate.

Figure 4b: K-map logic derivation and minimization for D_1^+

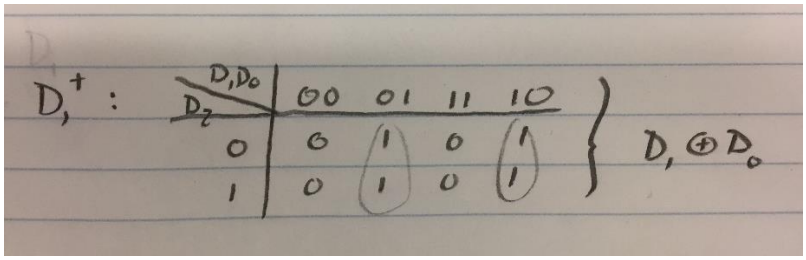


Figure 4b above shows that the logic expression for D_1^+ requires one XOR gate.

Figure 4c: K-map logic derivation and minimization for D_2^+

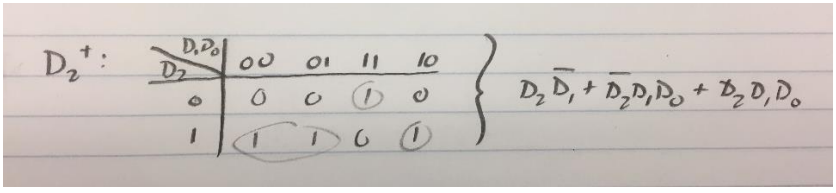


Figure 4c above shows that the logic expression for D_2^+ requires two NOT gates, five 2-input AND gates, and one OR gate.

Figure 5: K-map logic derivation and minimization for Z

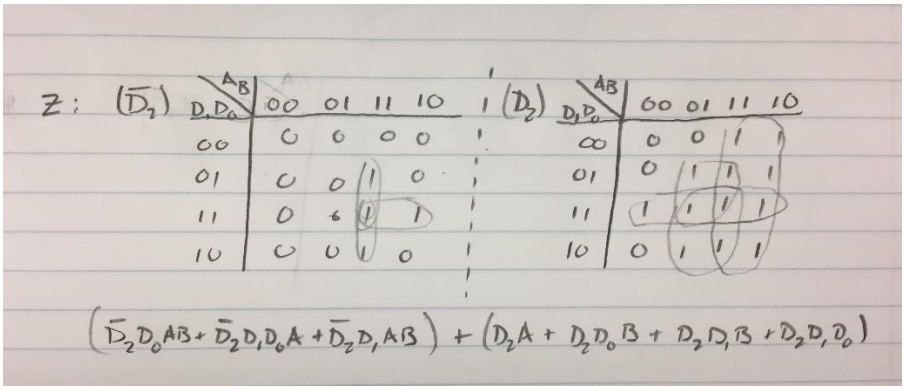


Figure 5 above shows that the logic expression for Z would need one NOT gate, seventeen 2-input AND gates, and six OR gates. Because this is so many gates, instead of just logic gates, an 8:1 MUX will be used with $D_2D_1D_0$ as the select lines and combinations of A and B (A, AB, A+B, 0, and 1) for the input lines. This will drastically reduce the number of gates needed, reducing final count to one 8:1 MUX, one AND gate, and one OR gate.

Figure 6: Gates and components list

Component	Quantity	Chips needed	Part Code
8:1 MUX	1	1 (1/chip)	74151
2input AND	6	2 (3/chip)	7408
2input OR	3	1 (3/chip)	7432
2input XOR	1	1 (3/chip)	7486
D flip flop	3	2 (2/chip)	7474
Timer	1	1 (1/chip)	555 timer

This is the final parts list for our proposed circuit. Note that the inverters are not included in the list, because the 7474 d flip-flop package contains an inverted output with each flip-flop.

Conclusion:

The initial design and planning of the finite state machine and circuit are complete. All results were as expected this was an exercise in methods learned in 0132 Digital Logic course. The Mealy design, state transition state assignments and table, and K-mapping allowed for the minimization of this solution to the circuit. The following parts will simulate, build, and test this solution.

Part II: Quartus II Simulation

Purpose:

The purpose of part 2 was to digitally make the finite state machine designed in part 1 and simulate it using the waveform generator using Quartus II software to see if the design worked as intended.

Procedure:

1. The logic expressions found in *Figures 4a-c* and *Figure 5* were converted into a schematic, utilizing two 7474 d flip-flop chips and one 74151 8:1 MUX chip. Chip packages for the gates were not used to ensure the correct logic was being used.

2. External Preset and Clear on the flip-flop were added and set to HIGH, because they are true-low inputs. This enable the system to initialize from a natural state.

3. Inputs A, B, and Clock, outputs D_2, D_1, D_0, and Z were added to the schematic in addition to the chips and gates to finish the schematic. See *Figure 7* for the final schematic.

4. The Waveform editor function of Quartus II was used to generate a timing diagram. All inputs were tested and each successfully output the correct sequence. See *Figure 8* in the results section for the waveform

Results:

Figure 7: Finite state machine circuit final schematic

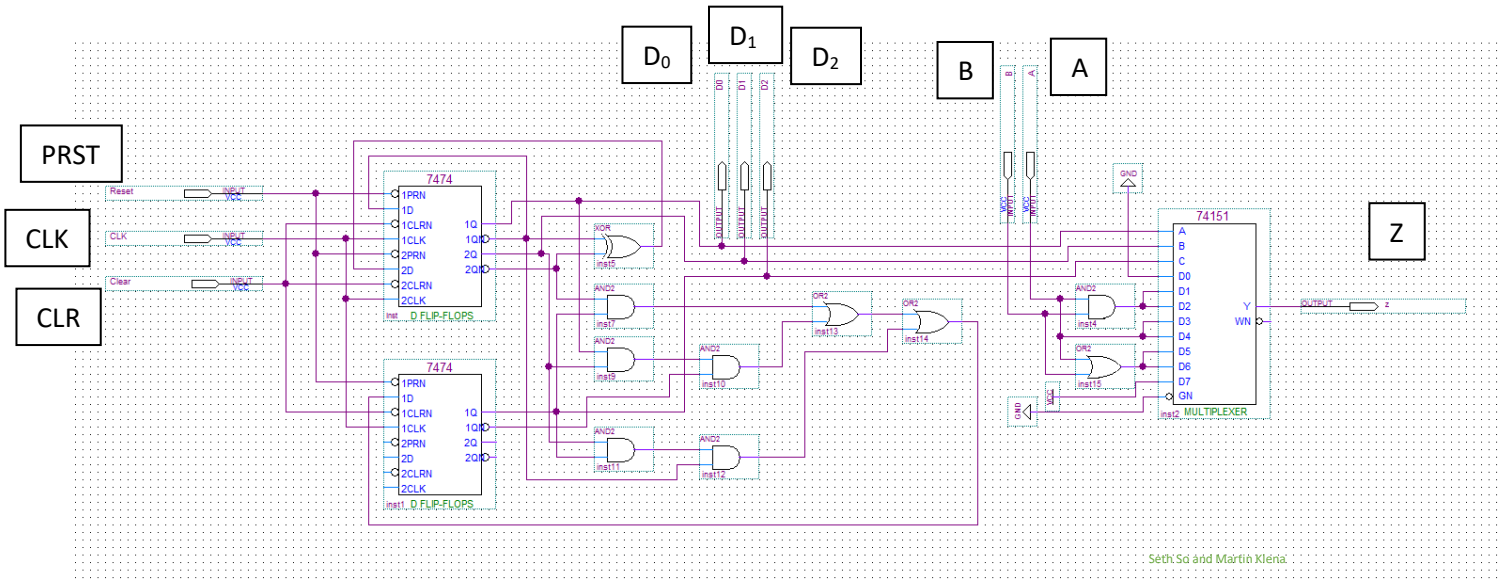


Figure 7 above shows the schematic for the finite state machine designed in part 1, designed using Quartus II software. For the purpose of viewing in the waveform editor, preset (PRST), clock (CLK), clear (CLR), A, and B are set to inputs, while flip-flop outputs D_2 D_1 D_0 and output Z were set to outputs.

Going from left to right: the flip-flops are initialized and set up to run synchronously, as they all use the same clock signal. Preset and clear are set to HIGH as they are true-low inputs. The outputs of the flip-flops go through several levels of gates to create the next state logic expressions found in *Figures 4a-c* and *Figure 5*. Lines from $D_2\,D_1\,D_0$ become the select lines of the multiplexer, which also takes in combinations of inputs A and B, finally outputting Z.

Figure 8: Complete waveform and timing diagram of inputs and outputs of schematic

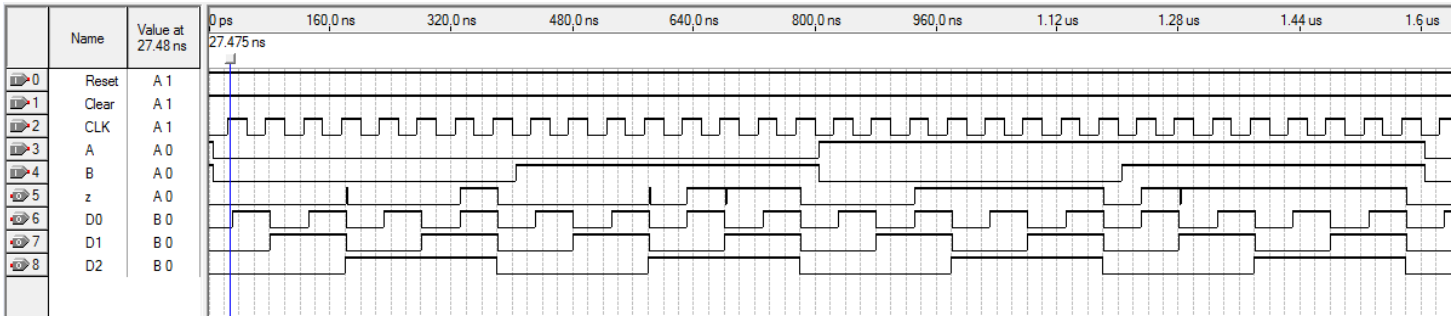


Figure 8 above displays the timing diagram generated from simulating the digital schematic with the inputs set as mentioned above. Working down the list of waveforms: preset (named reset on the dagram) and clear are set to HIGH for the duration of the simulation as explained before. Clock is set to a 50ns period, each combination of A and B (00, 01, 10, and 11) last a full eight clock cycles so that the entire output sequence may be seen in output Z. $D_2\,D_1\,D_0$ at the bottom act as the state counter, and work as described in Laboratory 9.

Important relations to note: the period of A lasts twice as long as B, which lasts twice as long as D_2, which lasts twice as long as D_1, which lasts twice as long as D_0, which lasts twice a long as CLK. This way, all combinations are tested to ensure that the circuit outputs the correct sequence, which it does. All changes occur on the rising edge of the clock. *Figure 1*, reproduced here, matches the behavior, and confirms that Z's behavior is outputting the correct sequence:

Figure 1: Required inputs and outputs of the finite state machine

AB (Input)	**Z** (Output Sequence)
00	0000000100000001…
01	0000011100000111…
10	0001111100011111…
11	0111111101111111…

Conclusion:

Building and simulation of the design through Quartus II software confirmed that the circuit will work when implemented. All results were as expected, though this procedure took over an hour to complete. The individual gates were meant to ensure that the correct next state logic expressions were being connected, but it may have been easier to instead implement the chips directly into the schematic. This would also greatly simplify building the actual circuit

Part III: Proto-Board Realization

Purpose:

The purpose of part 3 was to convert the digital schematic into a physical circuit utilizing the different integrated circuit chips in the lab kits. After completion, simulated results for the correct waveforms obtained in part 2 would be tested by checking the outputs of the realized circuit on the oscilloscope to confirm that the circuit was built correctly.

Procedure:

1. The main circuit was constructed according to the schematic, replacing all single logic gates with their respective IC chips. Preset and set were tied to HIGH voltage first, followed by the rest of the connections progressing through the circuit as a signal would. No LEDs were used, as outputs would be observed with the oscilloscope.

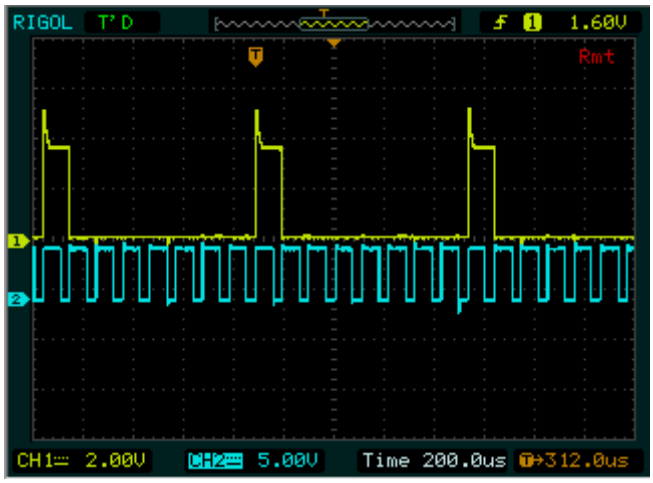
2. The 555 timer was then added to provide the clock signal. The 50ns period set in the simulation proved infeasible due to inability to make it with standard parts, so it was set to 100µs (C = 2nF, R_1 = 5.1kΩ, and R_2 = 4.7kΩ). See reference section for 555 timer circuit.

3. Inverted logic sources for A and B were built using SPDT switches along with 1kΩ resistors to prevent direct shorts to ground. These would enable for fast toggling and testing of A and B combination inputs.

4. After the entire circuit was completed, the different outputs and inputs labelled in *Figure 7* were observed on the oscilloscope. Despite the different clock period, the pattern of the waveforms exactly matched those in found in the simulation, confirming that the circuit was built correctly. See *Figure 9a-d* for the Z outputs compared with CLK.

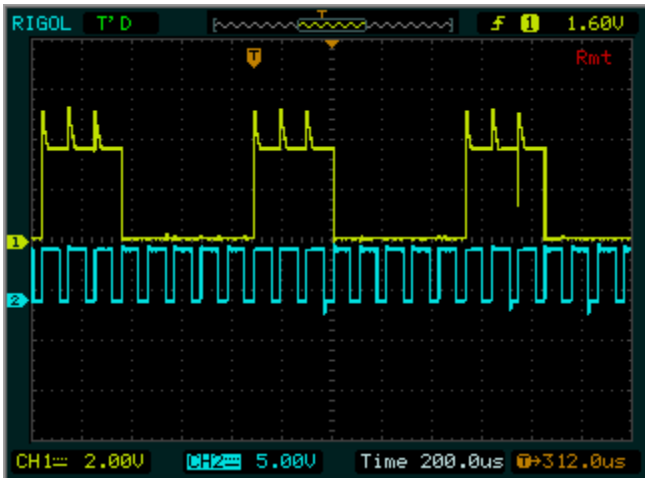
Results:

Figure 9a: oscilloscope observation of output Z (yellow) for AB = 00 and CLK (cyan)



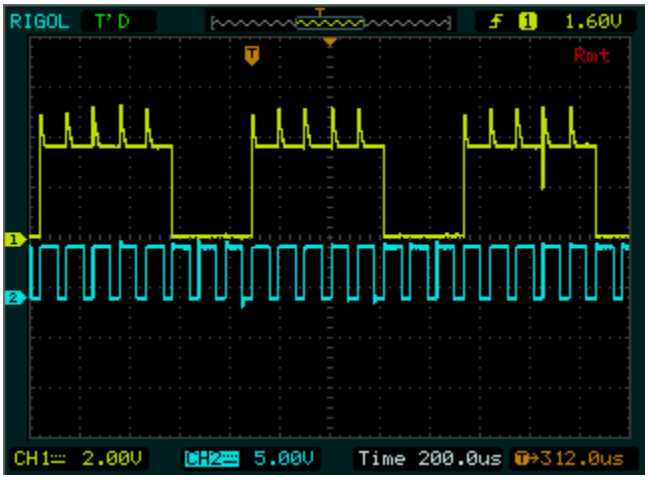
Seen in *Figure 9a* (previous page), the output for Z is as expected. There is a ratio of 1 output HIGH for every 8 CLK cycles, indicating that Z is outputting 0000000100000001… for input AB = 00, which is correct according to *Figure 1*.

Figure 9b: oscilloscope observation of output Z (yellow) for AB = 01 and CLK (cyan)



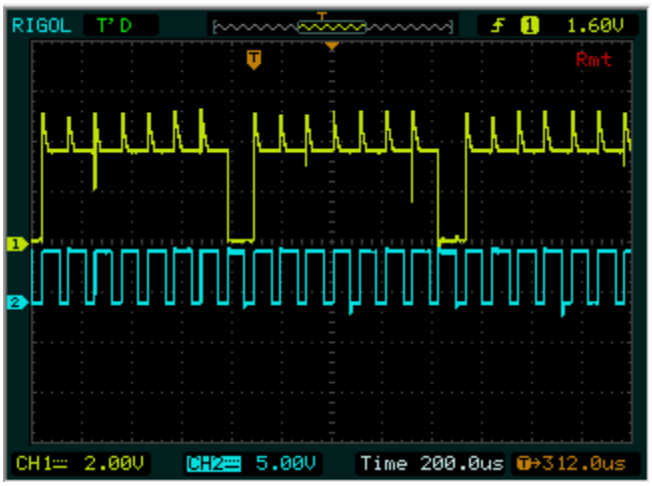
Seen above in *Figure 9b*, the output for Z is as expected. There is a ratio of 3 output HIGHs for every 8 CLK cycles, indicating that Z is outputting 0000011100000111… for input AB = 01, which is correct according to *Figure 1*.

Figure 9c: oscilloscope observation of output Z (yellow) for AB = 10 and CLK (cyan)



Seen above in *Figure 9c*, the output for Z is as expected. There is a ratio of 5 output HIGHs for every 8 CLK cycles, indicating that Z is outputting 0001111100011111… for input AB = 10, which is correct according to *Figure 1*.

Figure 9d: oscilloscope observation of output Z (yellow) for AB = 11 and CLK (cyan)



Seen above in *Figure 9d*, the output for Z is as expected. There is a ratio of 7 output HIGHs for every 8 CLK cycles, indicating that Z is outputting 01111111101111111… for input AB = 11, which is correct according to *Figure 1*.

Conclusion:

The oscilloscope observations confirmed that the circuit was functioning correctly. The circuit took over 3 hours to complete as the schematic did not contain the logic gate IC chips, making a direct transfer surprisingly difficult. Additionally, the timer period had to be changed to a value that was able to be made with standard value resistors and capacitors, but also was large enough that it could be observed accurately on the oscilloscope. These design considerations will be considered on the next Laboratory. Another final note is that though the 555 timer was emitting a stable signal, there appears to be some bouncing affect at the output. The cause is unknown, but it will be investigated at a later time.

Part IV: Logic Analyzer

Purpose:

The purpose of part 4 was to compare manual oscilloscope measurements to digital measurements (taken with the Intronic Logic Port Logic Analyzer). Whereas the digital oscilloscope has a course resolution, limited display, and simple triggering, the Logic Port Logic Analyzer has much finer resolution (nanoseconds compared to 10's of nanoseconds), can display many more channels (40 compared to 3), and has more complex triggering methods (much less susceptible to noise).

Procedure:

1. No LEDs were added to the circuit, so the probes were connected directly to the circuit according to the table in *Figure 10*:

Figure 10: Logic Analyzer to Circuit board mapping

Signal Color	Logic Analyzer	Circuit Board
Purple	CLK1	Timer output
Not displayed	GRN	Ground
Red	D0	A
Yellow	D1	B
Cyan	D3	Z
Green	D2	D_0
Green	D4	D_1
Green	D5	D_2

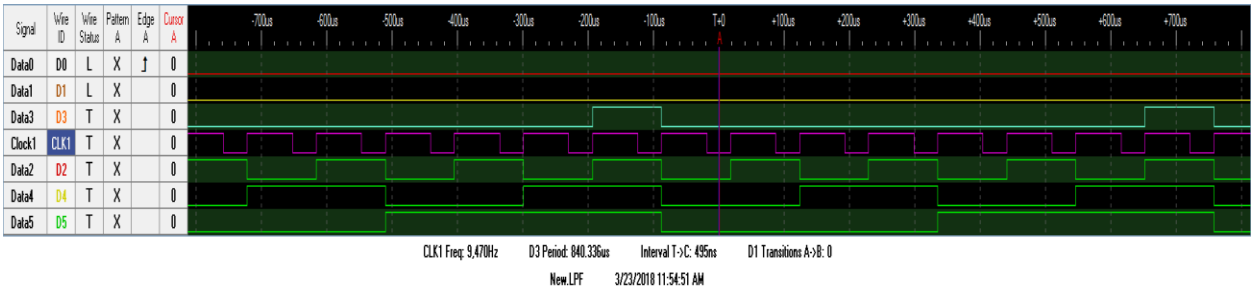
Note that D2 and D3 are out of order. This is deliberate and reflected in *Figures 11a-d* that display the waveform timing diagrams of these signals.

2. The trigger pattern was set to trigger once on the rising edge of the A. However, since A had no rising edge (always set to HIGH or LOW), the trigger was set off immediately. The trigger pattern should have been set to trigger once on the rising edge of the clock signal. That way, it would consistently line up with the a cycle, since the circuit is synchronous.

3. Sample rate was set to 200MHz to enable nanosecond precision (sample every 5 ns) and logic threshold to 1.4V to filter out any circuit noise. The trigger was then run so that images could be generated. *Figures 11a-d* show these results. See *Figure 10* for the coding.

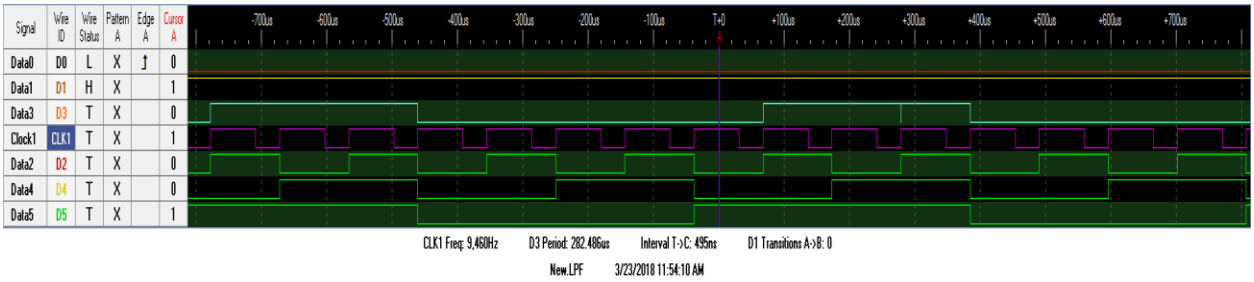
Results:

Figure 11a: Logic Analyzer analysis of circuit for AB = 00



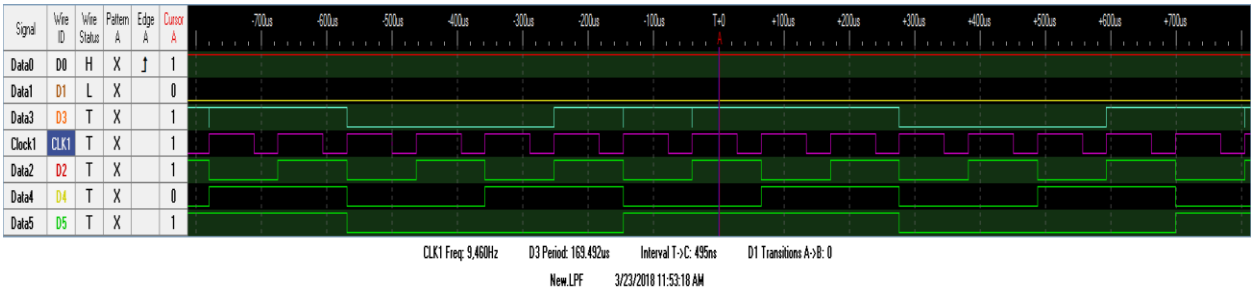
Seen in *Figure 11a* above, $D_2D_1D_0$ counts from 0-7, and the corresponding Z output is 00000001 as expected. All changes occur very neatly on the rising clock edge. Z is only HIGH for the final clock cycle in groups of 8, occurring when $D_2D_1D_0 = 111$.

Figure 11b: Logic Analyzer analysis of circuit for AB = 01



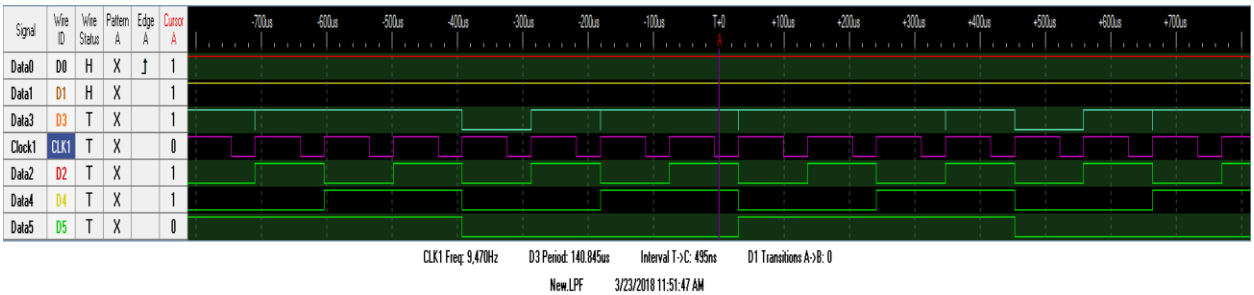
Seen in *Figure 11b* above, $D_2D_1D_0$ counts from 0-7, and the corresponding Z output is 00000111 as expected. There is an odd instant when Z is HIGH that it drops LOW, but only for an instant. It is unknown if this is a bug or the HIGH signal dips below the 1.4V logic threshold because of a large spike in noise for an incredibly brief timeframe. Z is HIGH for the final 3 of 8 clock cycles.

Figure 11c: Logic Analyzer analysis of circuit for AB = 10



Seen in *Figure 11c* above, $D_2D_1D_0$ counts from 0-7, and the corresponding Z output is 00011111 as expected. The strange LOWs in Z occur again, but all else is as it should be. Z is HIGH for the final 5 of 8 clock cycles

Figure 11d: Logic Analyzer analysis of circuit for AB = 11



Seen in *Figure 11d* above, $D_2D_1D_0$ counts from 0-7, and the corresponding Z output is 01111111 as expected. This is the longest Z where Z is HIGH for 7 of 8 clock cycles.

Conclusion:

The Logic Analyzer is a fantastic way to observe the input and output signals of logic circuits. Its improved triggering helps filter out much of the noise observed on the digital oscilloscope, and it can observe 13x as many channels as the oscilloscope can simultaneously. It would be very useful in debugging a circuit, as it can provide clean displays rapidly through very easy connections.

With regard to the laboratory, the Logic Analyzer matched both the simulated as well as the observed results, confirming that our circuit functioned as designed.

Summary:

The goal of Laboratory 10 was to design a synchronous, sequential circuit by planning out the finite state machine, simulating it in Altera Quartus II, building it on the proto-boards, observing it with the oscilloscope, and finally analyzing it through the specialized peripheral LogicPort.

Important design considerations arose including whether it was possible to build around parameters set in simulations (i.e. the timer period), whether to use logic gates in the schematic for clarity or the final components for practicality (lesson learned to use the final components), an, in summary, how to go about constructing a finite state machine from start to finish.

The application of the circuit is that it simulates pulse width modulation, which can be used to alter duty cycle (vary the fraction for which a signal is active). Pulse width modulation also has a myriad of other uses, primarily in using digital signals to represent analog signals.

References:

555 timer datasheet: https://courseweb.pitt.edu/bbcswebdav/pid-24291878-dt-content-rid-23464842_2/courses/2184_UPITT_ECE_0501_SEC1010/555_timer.jpg

7474 datasheet: https://courseweb.pitt.edu/bbcswebdav/pid-24291878-dt-content-rid-23464850_2/courses/2184_UPITT_ECE_0501_SEC1010/7474.pdf

7408 datasheet: https://courseweb.pitt.edu/bbcswebdav/pid-24291878-dt-content-rid-23464848_2/courses/2184_UPITT_ECE_0501_SEC1010/7408.pdf

7432 datasheet: https://courseweb.pitt.edu/bbcswebdav/pid-24291878-dt-content-rid-23464848_2/courses/2184_UPITT_ECE_0501_SEC1010/7408.pdf

7486 datasheet: https://courseweb.pitt.edu/bbcswebdav/pid-24291878-dt-content-rid-23464851_2/courses/2184_UPITT_ECE_0501_SEC1010/7486.pdf

74151 datasheet: https://courseweb.pitt.edu/bbcswebdav/pid-24291878-dt-content-rid-23464845_2/courses/2184_UPITT_ECE_0501_SEC1010/74HC_HCT151_CNV_2.pdf