Tiffany Q. Liu
March 21, 2011
CSC 270
Lab #7

Lab #7: GYR Sequencer

Introduction

The focus of this lab was to use a 74LS74 to build a 4-state sequencer that generates signals representing the Green, Yellow, and Red lights of a traffic light.

Materials

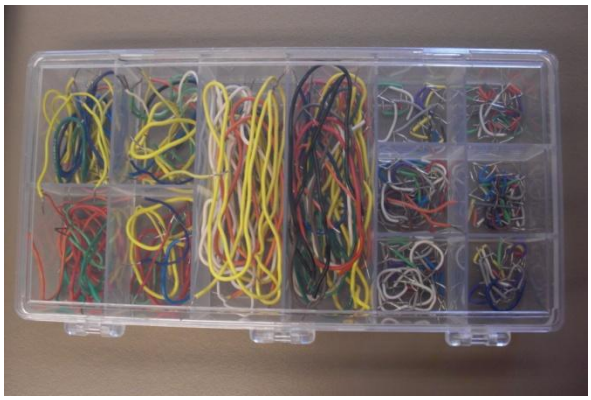

Figure 1. Wiring Kit.

Figure 2. Digital Training Kit.

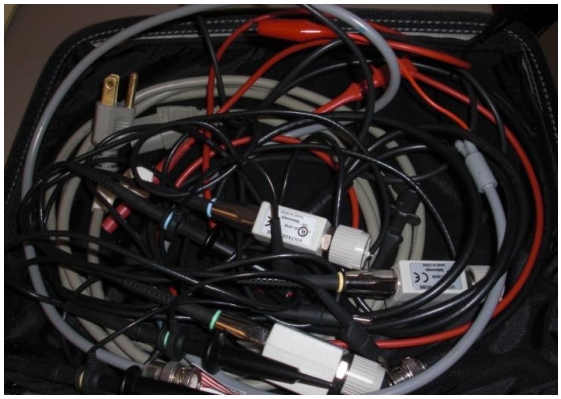

Figure 3. Oscilloscope cables.

Figure 4. Tektronix MSO3000/DPO3000 Oscilloscope

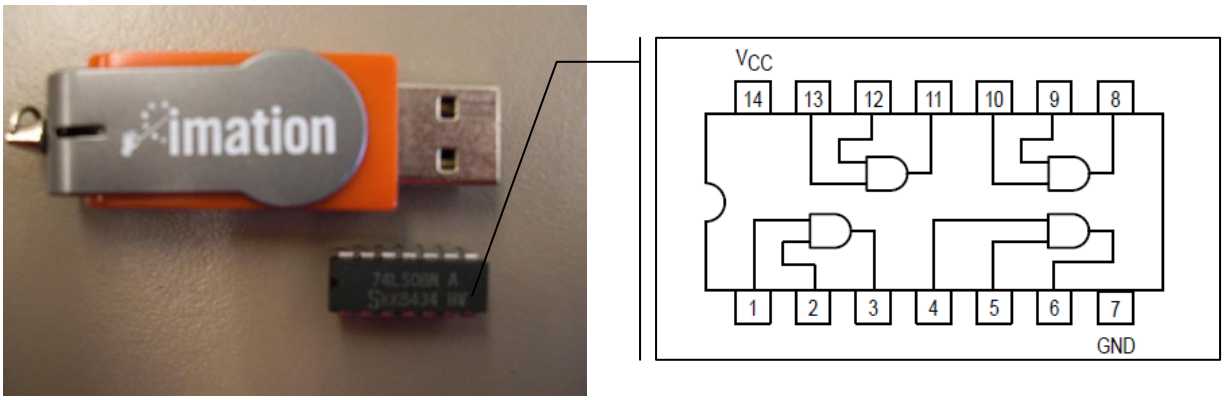

Figure 5. Quad 2-Input AND Gate 74LS08 Compared to a USB Flash Drive.

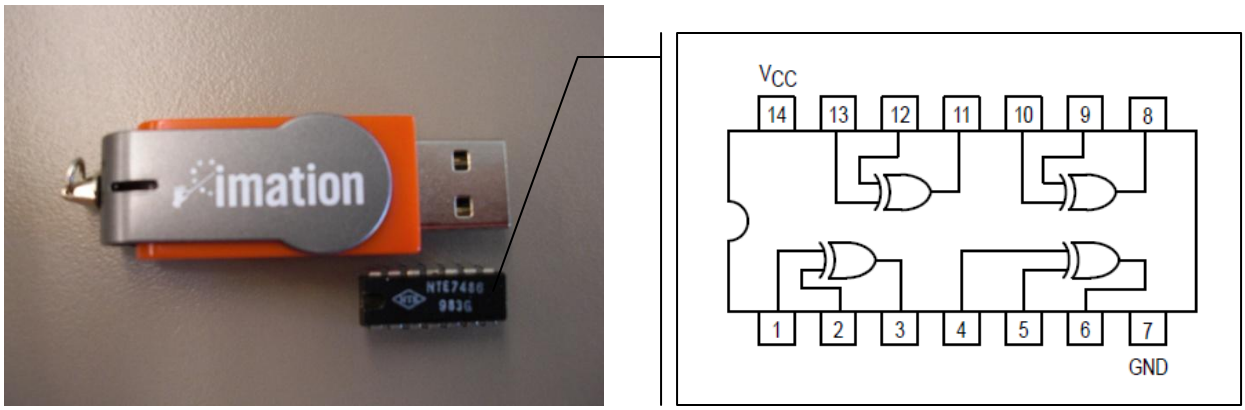

Figure 6. Quad 2-Input XOR Gate 74LS86 Compared to a USB Flash Drive.

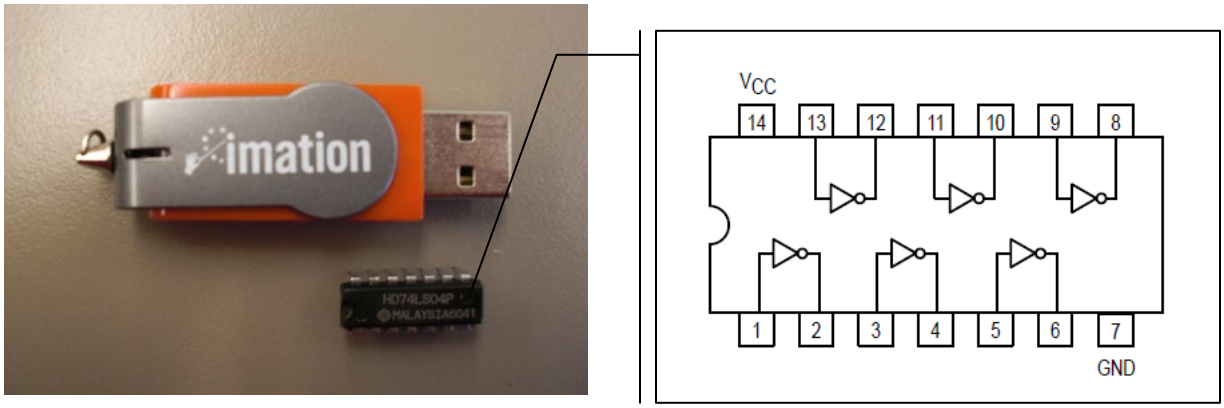

Figure 7. Hex Inverter 74LS04 Compared to a USB Flash Drive.

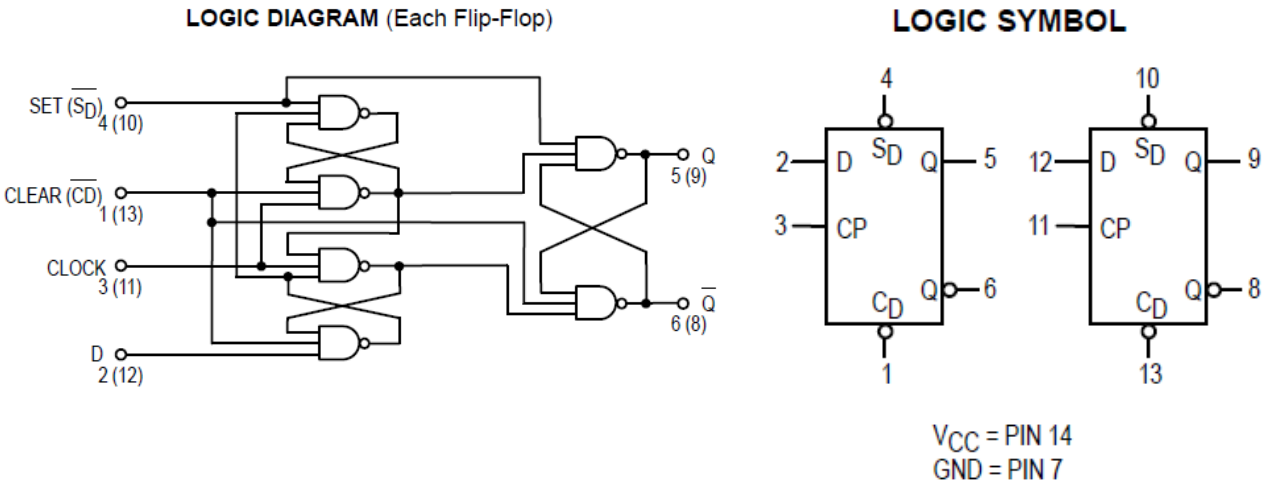

Figure 8. Dual D-Type Positive Edge-Triggered Flip-Flop 74LS74.

The GYR Sequencer

To implement a traffic-light sequencer according to the given state diagram (Fig. 9), we first drew a timing diagram (Fig. 10).

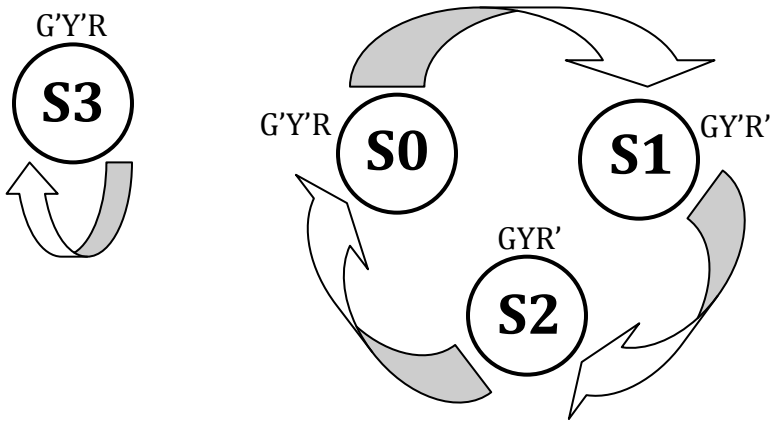

Figure 9. State Diagram of GYR Sequencer.

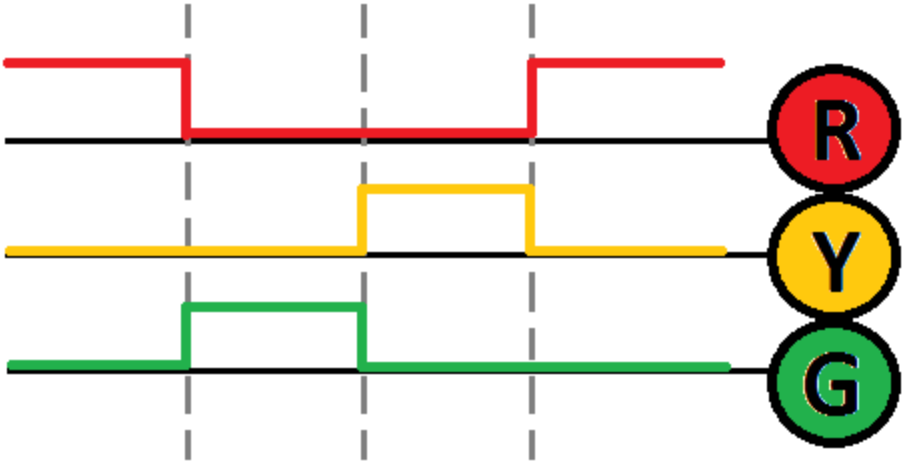

Figure 10. Corresponding Timing Diagram for GYR Sequencer.

Next, we wrote up the following state transition table based on the state diagram:

Present State S_n	Future State S_{n+1}
S_0	S_1
S_1	S_2
S_2	S_0
S_3	S_3

Table 1. State Transition Table for GYR Sequencer.

Knowing that there are 4 states to generate, we calculated the number of D flip-flops we needed to build our sequencer based on the fact that n flip-flops can generate 2^n states. In our case, $2^n = 4$, so n would equal 2, which means that we needed 2 D flip-flops.

Then we assigned each state node in the state diagram with the binary representation of flip-flop output Q_1Q_0:

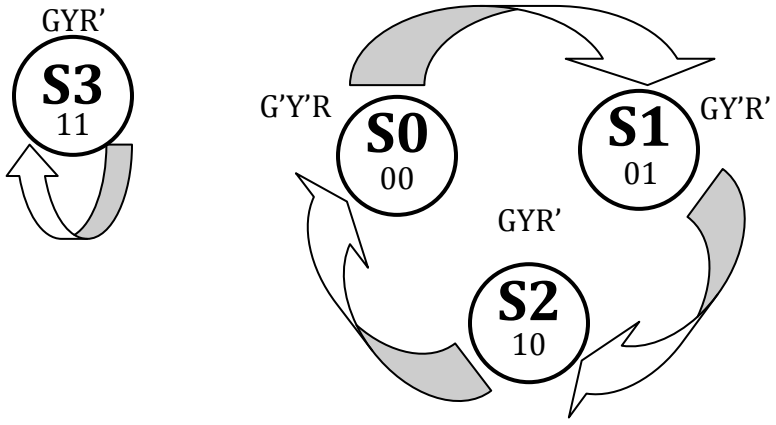

Figure 11. State Diagram of GYR Sequencer with Assigned Flip-Flop Output.

We then updated the state transition table to include the Q outputs at time n and n+1:

Present			Future		
S_n	$Q^n{}_1$	$Q^n{}_0$	$Q^{n+1}{}_1$	$Q^{n+1}{}_0$	S_{n+1}
S_0	0	0	0	1	S_1
S_1	0	1	1	0	S_2
S_2	1	0	0	0	S_0
S_3	1	1	1	1	S_3

Table 2. State Transition Table with Q Outputs.

Then we added the outputs (G, Y, R) to the above table:

Present			Future			Output		
S_n	$Q^n{}_1$	$Q^n{}_0$	$Q^{n+1}{}_1$	$Q^{n+1}{}_0$	S_{n+1}	G	Y	R
S_0	0	0	0	1	S_1	0	0	1
S_1	0	1	1	0	S_2	1	0	0
S_2	1	0	0	0	S_0	0	1	0
S_3	1	1	1	1	S_3	0	0	1

Table 3. State Transition Table with G, Y, R Outputs.

Finally, we determined the Boolean equations for the D inputs and for the G, Y, and R outputs. Using the above table we were able to determine that $D_1 = Q^{n+1}{}_1 = Q^n{}_0$, $D_0 = Q^{n+1}{}_0 = Q^n{}_1' \cdot Q^n{}_0' + Q^n{}_1 \cdot Q^n{}_0 = (Q^n{}_1 \oplus Q^n{}_0)'$, $G = Q^n{}_1' \cdot Q^n{}_0$, $Y = Q^n{}_1 \cdot Q^n{}_0'$, and $R = D_0 = (Q^n{}_1 \oplus Q^n{}_0)'$. We verified our Boolean expressions using a modified version of D. Thiebaut's GYRSequencer python program:

```
# GYRSequencer.py
# Tiffany Liu and Millie Walsh
# Modified version of D. Thiebaut's very quick and dirty way to check a
# sequencer...
# This sequencer activates a G, Y, and R light system
# s.t. G is on for 2 cycles, followed by Y for 1 cycle
# followed by R for 1 cycle.  Then the whole cycle repeats
# ----------------------------------------------------------------------
# Output when Q1 and Q0 start at 00:
# Q1Q0 = 0 0  | GYR = 0 0 1
# >
# Q1Q0 = 0 1  | GYR = 1 0 0
# >
# Q1Q0 = 1 0  | GYR = 0 1 0
# >
# Q1Q0 = 0 0  | GYR = 0 0 1
# >
# Q1Q0 = 0 1  | GYR = 1 0 0
# >
# Q1Q0 = 1 0  | GYR = 0 1 0
# >
# Q1Q0 = 0 0  | GYR = 0 0 1
# >
# Q1Q0 = 0 1  | GYR = 1 0 0
# >
# Q1Q0 = 1 0  | GYR = 0 1 0
# >
# Q1Q0 = 0 0  | GYR = 0 0 1
# >
#
# Output when Q1 and Q0 start at 11:
# Q1Q0 = 1 1  | GYR = 0 0 1
# >
# Q1Q0 = 1 1  | GYR = 0 0 1
# >
# Q1Q0 = 1 1  | GYR = 0 0 1
# >
# Q1Q0 = 1 1  | GYR = 0 0 1
# >
# Q1Q0 = 1 1  | GYR = 0 0 1
# >
# Q1Q0 = 1 1  | GYR = 0 0 1
# >
# Q1Q0 = 1 1  | GYR = 0 0 1
# >
# Q1Q0 = 1 1  | GYR = 0 0 1
# >
# ----------------------------------------------------------------------

Q1 = 0
Q0 = 0

for step in range( 20 ):

    # the Q1 and Q0 outputs go through combinational logic to generate
    # the new values of D1, D0, and the outputs G, Y, R...
    D1 = Q0
    D0 = not (Q0 ^ Q1)
    G  = (not Q1) & Q0
    Y  = Q1 & (not Q0)
    R = D0

    # show the stable circuit signals
    print "Q1Q0 = %d %d  | GYR = %d %d %d" % ( Q1, Q0, G, Y, R )

    # wait for the next clock tick (the user presses Enter)
    raw_input( "> " )

    # as soon as the clock has ticked, D1 and D0 get latched in the
    # flipflops and Q1 and Q0 reflect the values captured.
    Q1 = D1
    Q0 = D0
```

Figure 12. Python Program Used to Verify Boolean Expressions with Output.

Once we verified that our Boolean expressions were valid, we wired up the following circuit with the clock CLK set to 1Hz to demonstrate that our circuit worked with the G, Y, and R outputs connected to LEDs:

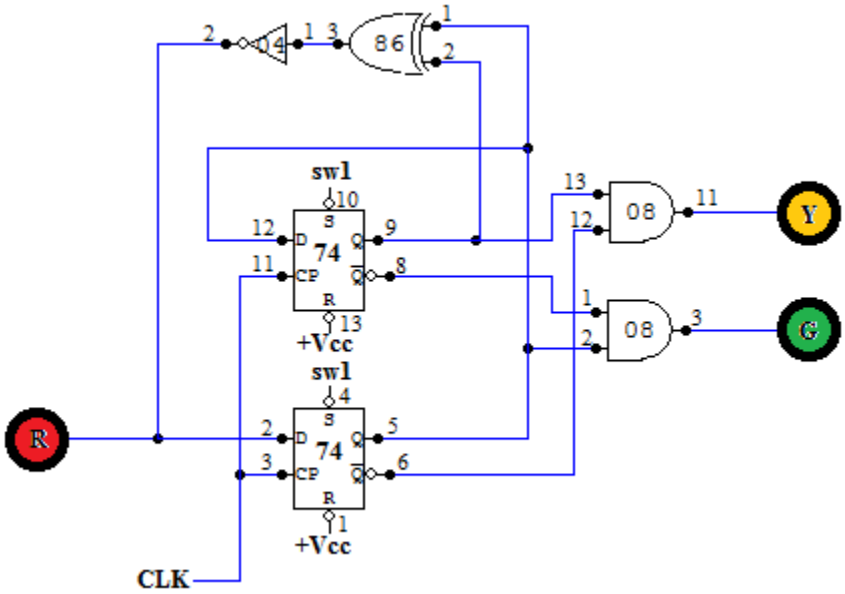

Figure 13. Circuit Diagram for GYR Sequencer.

When sw1 was set to 1, our GYR traffic-light sequencer went through cycles of $S_0 \rightarrow S_1 \rightarrow S_2$. When sw1 was set to 0, our GYR traffic-light sequencer stayed in S_3. In addition, when we set sw1 back to 1, our GYR traffic-light sequencer kept staying in S_3. This followed our timing and state diagrams for our sequencer.

Finally, we connected our circuit to the oscilloscope to generate the 3 output signals G, Y, and R with the clock CLK set to 100kHz. At first, we disconnected the output from the LEDs and connected them to the oscilloscope with the voltage probes. However, the signals generated by the scope were not what we expected, so we had to do some troubleshooting. First we checked to see if there were any loose connections. When we determined that there were none, we connected the outputs to both the LEDs and to the scope. Here, the signals for Y and R were being generated as expected, however G was not. When the probe was connected to the G output via the LED, the signal for G came out as expected. When this was done, we obtained the following snapshot:

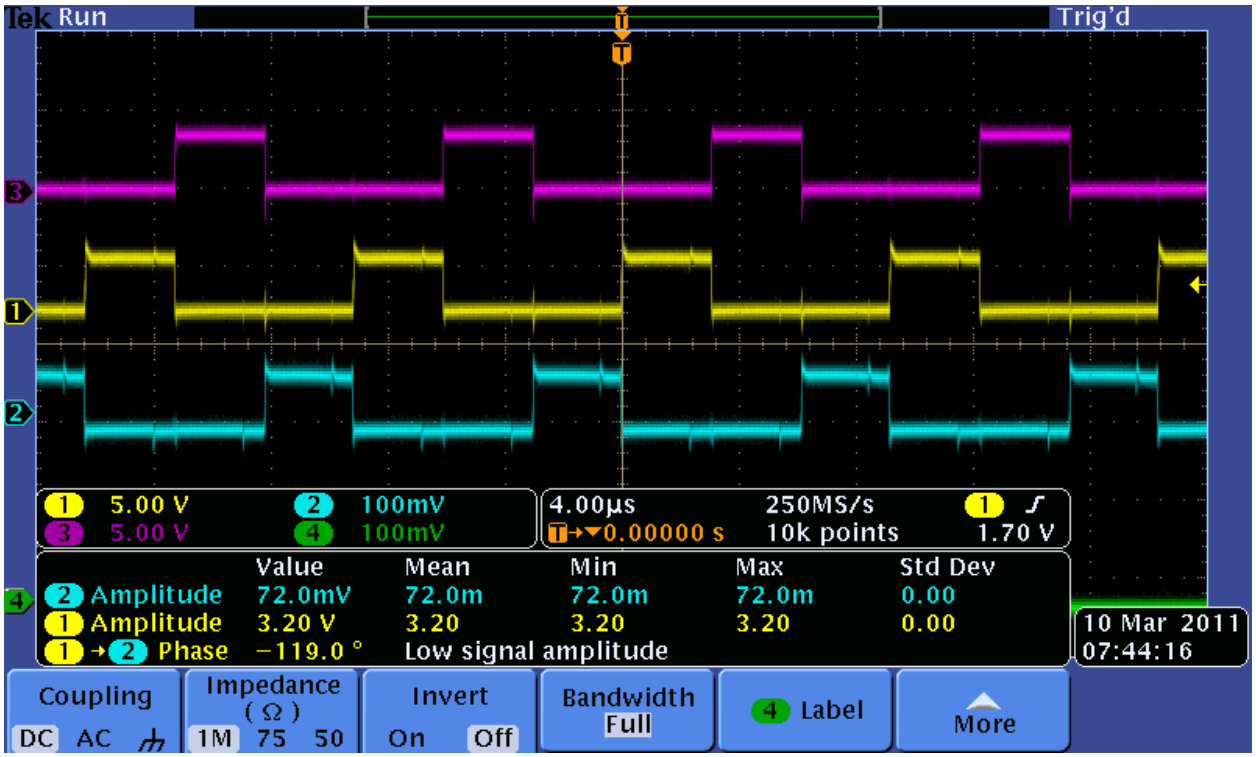

Figure 14. Screen Shot from Oscilloscope of Circuit from Figure Showing the Output Signals R (Purple), Y (Yellow), and G (Blue).

Figure 14 matched the timing diagram for our GYR sequencer (Fig.10), verifying that our circuit worked.