Introduction to Logic Design

ECE 265

ECE UIC Department of Electrical and Computer Engineering

12 Laboratory Experiments

Written by Vladimir Goncharoff and Robert Becker

Laboratory Report Format:

1. Objective

   Concisely state the goal of this experiment

2. Theoretical Work

   Show all work required to derive a Boolean algebraic expression for the desired logical function, the corresponding truth table, and anything else that is asked for. (For later experiments this will include state transition diagram, state table, etc.)

3. Constructed Circuit Diagram

   Neatly draw the circuit that you have built using logical gate symbols. Clearly label input and output variables. IC pin numbers and power supply connections need not be shown.

   Label each node in your circuit, then derive logical expressions for these nodes in terms of input variables (and later, also in terms of state variables). This information is extremely useful for debugging your circuit. Write these logical expressions underneath the circuit diagram.

4. Parts List and Pinout Diagrams

   Write a list of logic gate IC's (e.g. 74LS00 - Quad NAND) used in your circuit, and how many of each were needed. Include, for reference purposes, a pinout diagram for each type of IC. (Reference your source).

5. Circuit Testing Procedure and Results

   How was the circuit tested, what results were obtained? A truth table format is sufficient.

6. Discussion

   Did your results agree with the theoretical predictions? Explain any discrepancies between theoretical and measured results. Optional: include your suggestions for an improved experimental procedure, alternative solution methods, etc.

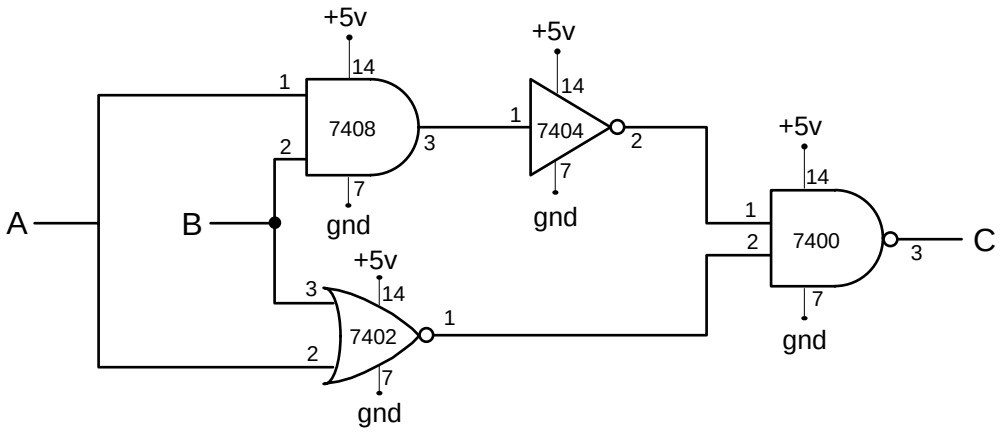
7. Answers to Questions

University of Illinois at Chicago
Department of Electrical and Computer Engineering
ECE 265 - Introduction to Logic Design

Lab Experiment #1

Digital Circuits and Truth Table

Circuit 1:



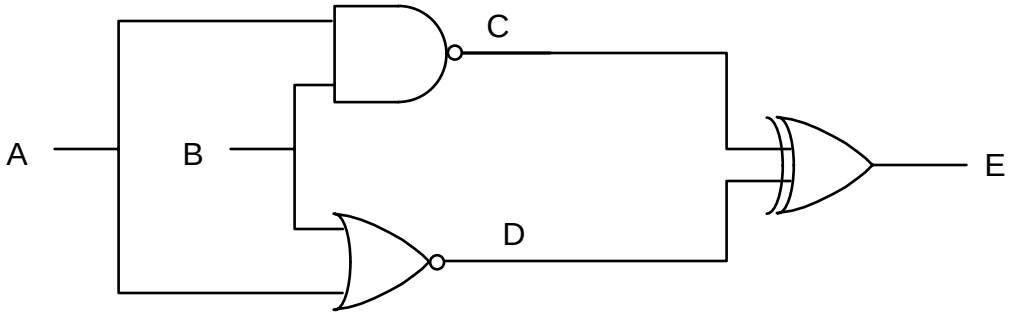
1. Draw a truth table for Circuit #1. Include colums for intermediate binary variables (e.g. pin 3 of 7408, etc., themselves functions of A, B).

2. Write a Boolean algebraic expression for output variable C in terms of variables A and B. It need not be in simplified form.

Before proceeding, make sure that you have read pp. 1-30 of the Becker lab manual.

3. Construct Circuit #1. Set the values of input variables A,B using a switch and resistor combination as shown in Figures 1.5 and 3.9 of the lab manual. Measure the value of output variable C by connecting an LED/resistor series combination, as shown in Figures 1.6 and 3.8 of the lab manual, to the NAND gate output. Verify that the circuit operation is correctly predicted by your truth table in part 1.

Circuit 2: (consult the lab manual data sheets, beginning at p. 31, for IC pinouts)



4. Draw a truth table for Circuit #2. Include two columns for input binary variables A,B; include 2 columns for intermediate binary variables C, D; and include a column for output binary variable E.

5. Write Boolean algebraic expressions for variables C, D and E in terms of variables A and B.

6. Construct Circuit #2 (alongside Circuit #1). Set the values of input variables A,B and measure the value of output variable as with Circuit #1. Verify that the circuit operation is correctly predicted by your truth table in part 4.

7. Demonstrate both circuits to your T.A.

Questions for Experiment #1:

- Visit the web site of the mail-order electronic parts vender www.jameco.com, and find their single-quantity prices for the integrated circuits that you have used in this experiment. For example, an inverter may be found under part number 74LS04 (DIP-14 version). Include the first page of the technical document that they provide for each IC as part of your lab report.

- Draw a simpler circuit (i.e. having fewer gates) that has the same logic function as does Circuit #2. How much does it cost to build the circuit (according to the prices at www.jameco.com) before and after the simplification? Assume single quantity pricing and ignore shipping costs.

University of Illinois at Chicago
Department of Electrical and Computer Engineering
ECE 265 - Introduction to Logic Design

Lab Experiment #2

Introduction to Digital Circuit Design

The goal of this lab experiment is to introduce the topic of digital logic design based on a set of performance specifications.

1. First, wire up three of the switches on the DIP-switch-package in a circuit so that 3 green LEDs can be individually turned on and off. Be sure to put a 220Ω resistor in series with each LED to limit the current flowing through it, as described in the lab manual.

2. Let binary variables G_1, G_2 and G_3 represent the green LED states: $G_1 = 1$ when the first green LED is on, $G_1 = 0$ when the first green LED is off, etc. Draw a truth table for function $Y(G_1,G_2, G_3)$, such that $Y = 1$ whenever two or fewer green LEDs are on, and $Y = 0$ when this is not the case. Write a Boolean algebraic expression for Y in terms of G_1, G_2 and G_3.

3. Similarly, draw a truth table for another function $R(G_1, G_2, G_3)$, such that $R = 1$ whenever two or more green LEDs are on, and $R = 0$ when this is not the case. Write a Boolean algebraic expression for R in terms of G_1, G_2 and G_3.

4. Design and construct a digital logic circuit to produce outputs Y and R from inputs G_1, G_2 and G_3. (Since you probably have not yet studied logical function simplification, your circuit need not be in its most compact form.) Use a common-sense approach in selecting and interconnecting digital logical gates from your parts kit. Display the value of output variable Y using a yellow LED, and the value of output variable R using a red LED.

5. Demonstrate the circuit to your T.A.

6. Answer the following questions:

(a) Write the truth tables and Boolean algebraic expressions found for $Y(G_1, G_2, G_3)$ and $R(G_1, G_2, G_3)$ in parts 2 and 3.

(b) Using Boolean algebraic manipulations, derive expressions for $Y(G_1, G_2, G_3)$ and $R(G_1, G_2, G_3)$ in Sum of Products (SOP) form.

(c) Using Boolean algebraic manipulations, derive expressions for Y(G1, G2, G3) and R(G1, G2, G3) in Product of Sums (POS) form.

University of Illinois at Chicago
Department of Electrical and Computer Engineering
ECE 265 - Introduction to Logic Design

Lab Experiment #3

Designing Digital Circuits from Problem Specifications

Problem Specification

There are three locks on a river, situated one next to the other. Each lock has a gate that can be opened or closed in response to a control signal. Because the three locks are in series blocking the normal river flow to prevent flooding downstream, one of these lock gates must be closed at any time. To provide an additional measure of safety, operations rules recommend that at any time two of the three gates be closed.

The river lock operator has three switches on a console: these generate logic variables X_1, X_2 and X_3 (logical 1 to open a gate, logical 0 to close a gate). Logic variables X_1, X_2 and X_3 aren't used to control the lock gates directly, however, because the operator is known to have a few drinks before coming to work!

Your task is to design a digital logic circuit that accepts operator-generated signals X_1, X_2 and X_3 as input variables and produces output signals $\{Y_1, Y_2, Y_3\}$, W and A. Output signals Y_1, Y_2, and Y_3 are then be used as the actual controlling signals for the lock gates, output W is a warning signal and output A is the alarm signal.

Definition of Y_1, Y_2, Y_3 , W, A in terms of X_1, X_2 and X_3:

- $Y_n = X_n$ and W = A = 0 when one or fewer input signals are high (normal condition, at most one lock is open);
- $Y_n = X_n$, W = 1 and A = 0 when exactly two input signals are high (warning condition, 2 locks are open);
- $Y_n = 0$, W = 0 and A = 1 when the river lock operator had moved switches intending for all three locks to open (causing all locks to close and sounding an alarm).

Design and build a circuit that realizes the above-specified behavior (displaying the states of all input and output variables using LEDs) and demonstrate its operation to your T.A.

Design Method

M1. Draw a truth table that relates all possible input states of X_1, X_2 and X_3 to the desired outputs Y_1, Y_2, Y_3, W and A.

M2. Find expressions for each output variable in canonical SOP and canonical POS forms.

M3. Using Boolean algebraic simplification, find Boolean algebraic expressions for each output variable in both simplest SOP and simplest POS forms.

M4. Draw a schematic diagram of the circuit that you have designed to implement the functional relations described in M3. Use only gates of the type found in your lab kit (e.g. no three-input gates allowed).

Questions:

Q1. Assuming that an output A already exists (but all other outputs do not), write expressions for {Y_1, Y_2, Y_3, W} in terms of {X_1, X_2, X_3 and A}. Do this using your common sense, a truth table solution is not required.

Q2. Draw a simplified circuit that obtains output signals {Y_1, Y_2, Y_3, W, A} from input signals {X_1, X_2, X_3} based on your results in Q1. You may choose to build this circuit instead of the one derived in M3 for demonstration purposes.

. . .

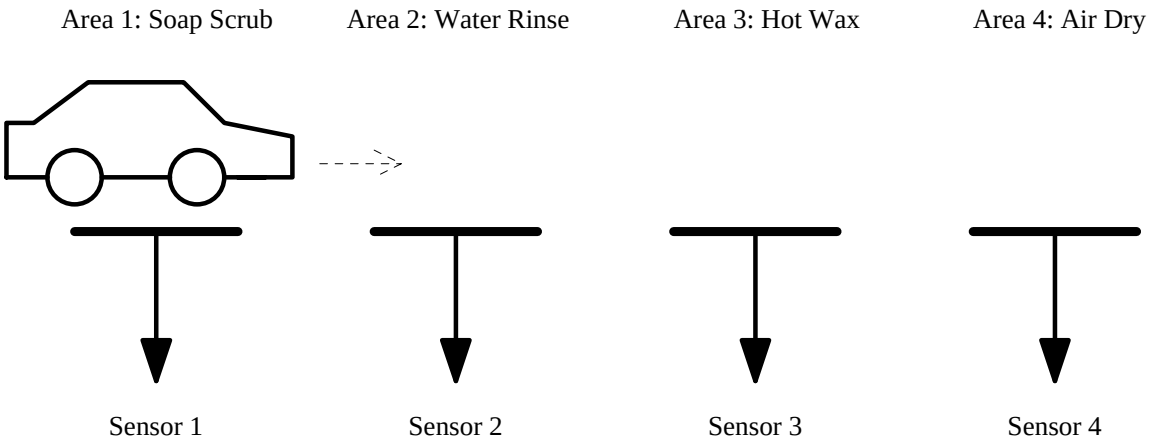
University of Illinois at Chicago
Department of Electrical and Computer Engineering
ECE 265 - Introduction to Logic Design

Lab Experiment #4

Simplification of Boolean Functions Using "Don't Care" Conditions

Introduction

The diagram below depicts an automobile in an automatic car wash, beginning a wash sequence as it is slowly being pulled through. When the auto first enters Area 1, Sensor 1 detects its presence causing soapy water to be applied by rotating scrubbers. When the auto leaves Area 1, Sensor 1 is no longer activated causing that function to be turned off. Similarly, Sensors 2 through 4 control the other three car washing functions as the auto is pulled through the corresponding areas.



Input/Output Control Signals

Binary variables $\{S_1, S_2, S_3, S_4\}$ are the outputs of Sensors 1 through 4 (= 1 when the auto is above the sensor, and otherwise = 0). You will simulate these signals using switches, and they will serve as input variables to the circuit that you will design.

Binary variables $\{C_1, C_2, C_3, C_4\}$ control the car wash functions. These signals are the outputs of a digital circuit that you will design. For each of these control signals logical "1" activates the corresponding car wash function to be ON, and logical "0" turns the function OFF.

Your task will be to design digital logic circuits that accept input signals $\{S_1, S_2, S_3, S_4\}$ and produce output signals $\{C_1, C_2, C_3, C_4\}$ according to certain design specifications.

Method 1

This circuit should produce $C_k = S_k$ (k=1 through 4), with the following exception:

> As an automobile crosses the boundary between adjacent car wash areas, it is possible that two sensors are simultaneously activated. To prevent any interference between different washing operations, turn off all car wash functions {make $C_1 = C_2 = C_3 = C_4 = 0$} whenever any two *adjacent* sensor outputs are high.

Method 2

Same specifications as in Method 1, except:

> Make C1 = C2 = C3 = C4 = X (don't care) for any row where any two *non-adjacent* sensor outputs are high. This corresponds to sensor states that should never occur when a single car is being washed.

Choose the method resulting in fewest gates, build the circuit and demonstrate it to your T.A. Show what happens when (a) only one sensor is high; (b) exactly two adjacent sensors are high, and (c) exactly two non-adjacent sensors are high.

Design Method

For each of the two methods:

- State and explain the design specifications (why they make sense)
- Write a truth table for each output variable $\{C_1, C_2, C_3, C_4\}$ as a function of input variables $\{S_1, S_2, S_3, S_4\}$
- Find expressions for each output variable $\{C_1, C_2, C_3, C_4\}$ in canonical SOP and canonical POS forms.
- Draw K-maps and arrive at simplest expressions for $\{C_1, C_2, C_3, C_4\}$ in SOP form.
- Draw K-maps and arrive at simplest expressions for $\{C_1, C_2, C_3, C_4\}$ in POS form.
- Design a digital circuit to compute output variables from input variables having the minimum number of gates. Neatly draw and label a schematic diagram of the circuit. Include circuitry to create input signal levels using switches and pull-up resistors, and LEDs to display both input and output variable states.

Questions:

Q1. For Method #2, what is the actual truth table that your circuit has implemented? Note: replace any X (don't care) with a 1 or 0 that the circuit actually produces.

Q2. What are the advantages and disadvantages of the circuit from Method 2 as compared the the circuit from Method 1?

Q3. How would you change the circuit specifications to allow for more than one car at a time to be in the car wash? Write a truth table.

University of Illinois at Chicago
Department of Electrical and Computer Engineering
ECE 265 - Introduction to Logic Design

Lab Experiment #5

Advanced Combinational Logic - 2's Complement Operator

Introduction

This experiment introduces multi-level combinational circuit design, 2's complement binary code, and the 2's complement operation.

Consider the following three-bit 2's complement codes for integers in the range [–4 ,+3]:

integer value:	2's complement code:
–4	1 0 0
–3	1 0 1
–2	1 1 0
–1	1 1 1
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1

2's complement binary notation has only one code for zero[1], and subtraction uses the same algorithm as does addition. For example, 3–4 is calculated as 3+(–4):

$$\begin{array}{rr} (0\,1\,1)_{\text{2's compl.}} & (+3)_{10} \\ +\,(1\,0\,0)_{\text{2's compl.}} & +(-4)_{10} \\ \hline (1\,1\,1)_{\text{2's compl.}} & =(-1)_{10} \end{array}$$

Note that in the above example the carry-out bit had been ignored, as is always the case when performing 2's complement subtraction.

[1] as compared to sign-magnitude notation, which has two codes for zero (+0 and –0)

The process of negating a number that is stored in 2's complement code is called the 2's complement operation. This is shown in the table below:

input integer value:	input 2's complement code:	output 2's complement code:	output (negated[2]) integer value:
–4	1 0 0	1 0 0	–4
–3	1 0 1	0 1 1	3
–2	1 1 0	0 1 0	2
–1	1 1 1	0 0 1	1
0	0 0 0	0 0 0	0
1	0 0 1	1 1 1	–1
2	0 1 0	1 1 0	–2
3	0 1 1	1 0 1	–3

Simple rule for performing the 2's complement operation:

Scan the bits from the right, keep all bits unchanged up to and including the first "1", and complement each of the bits to the left of there.

In this experiment you will design a circuit that performs the 2's complement operation on a 4-bit code.

Let $X_1X_2X_3X_4$ be the input variables (the input 4-bit 2's complement binary code, where X_1 is the most significant bit), and $Y_1Y_2Y_3Y_4$ be the corresponding output variables (the output 4-bit 2's complement binary code, where Y_1 is the most significant bit).

Create the input variables $X_1X_2X_3X_4$ using switches. Design a circuit that calculates $Y_1Y_2Y_3Y_4$ from $X_1X_2X_3X_4$, where the relation between the two is the 2's complement operation. Display input and output variables with LED's.

[2] Result is not correct for input value –4, because +4 cannot be represented by a 3-bit 2's complement code.

Design Method

M1. Draw a truth table that expresses $Y_1Y_2Y_3Y_4$ in terms of $X_1X_2X_3X_4$, as shown below:

X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
⋮	⋮	⋮	⋮				

M2. Draw K-maps to obtain simplified expressions for each of the output variables as a function of the input variables $\{X_1X_2X_3X_4\}$. Express your results in minimized SOP form.

M3. Based on the simplest SOP expressions found above, design four independent circuits, one for each of the output Y variables, using only gates that are available in your lab kits (i.e. 2-input logic gates). You need not build these circuits, only provide neat schematic diagrams of your designs.

M4. Often simpler circuits may be obtained by sharing terms that are common to all of the output variables. Such is the case here; create another paper design of a circuit applying this principle to generate $Y_1Y_2Y_3Y_4$ from $X_1X_2X_3X_4$ using fewer gates than sum total used by the four circuits found in M3.

> For example:
>
> If A = CD'(E'+ F) and B = EF'(C' + D), it is possible to share common terms.
> Let W = EF' = (E' + F)', and Z = CD' = (C' + D)'. Then A = ZW' and B = WZ',
> reducing the number of {AND, OR, NOT} operations from 10 to 8 overall.

M5. There is a simple method of generating $Y_1Y_2Y_3Y_4$ from $X_1X_2X_3X_4$ using three Exclusive-OR gates (plus some other logic). It is based on the "Simple rule for performing the 2's complement operation" that was stated previously. (Remember, an Exclusive-OR gate may be thought of as a programmable inverter.) Design and draw a circuit that generates $Y_1Y_2Y_3Y_4$ in this manner.

> Hint:
>
> Generate a signal that specifies whether or not all bits to the left of the current bit are to be inverted.

M6. Build a circuit based on what you have done in either M4 or in M5, and demonstrate to your T.A.

Questions:

Q1. Why isn't the simplest SOP (or POS) expression for an output variable always best for minimizing the total number of gates?

Q2. What would be the input/output characteristic of a circuit obtained by putting two of your 2's-complementers in cascade (outputs of first serving as inputs for the second)?

Q3. Assume that $X_1X_2X_3$ represents a signed integer in sign-magnitude code; that is, the integer value N is:

$$N = (-1)^{X_1} \times (X_2X_3)_{\text{unsigned binary}}$$

Observe that, in this code, X_1 is the sign bit (X_1=1 indicates a minus sign, X_1=0 indicates a plus sign). The range of integers represented is [–3 ,+3]. Design and draw a circuit that converts input variables $X_1X_2X_3$ (sign-magnitude code for value N) to output variables $Y_1Y_2Y_3$ (2's complement code for value N). Show the truth table describing your circuit's operation.

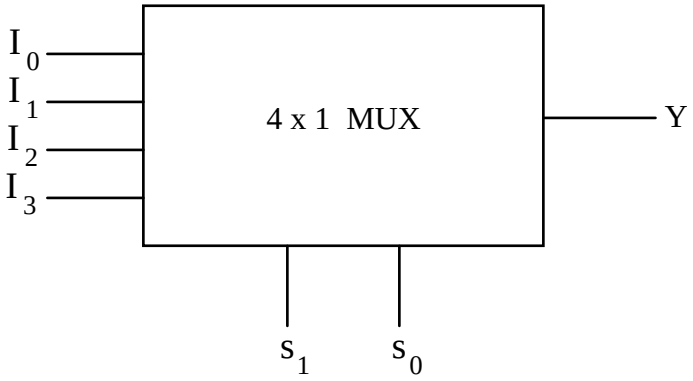
University of Illinois at Chicago
Department of Electrical and Computer Engineering
ECE 265 - Introduction to Logic Design

Lab Experiment #6

Combinational Logic Circuit Design Using Digital Multiplexers

Introduction

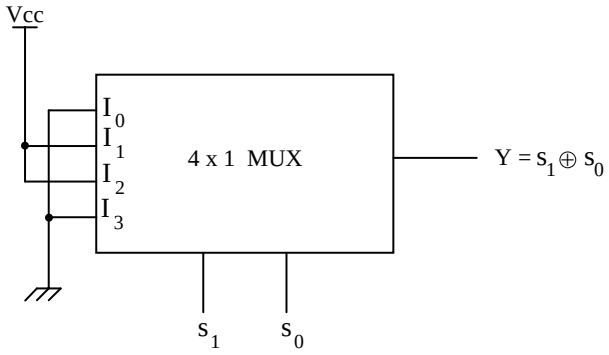
In this experiment you will design and build an advanced combinational digital circuit using the two 74153 multiplexer IC's that come with your lab kit. The basic concept of applying digital multiplexers to implementing logic functions is shown below:



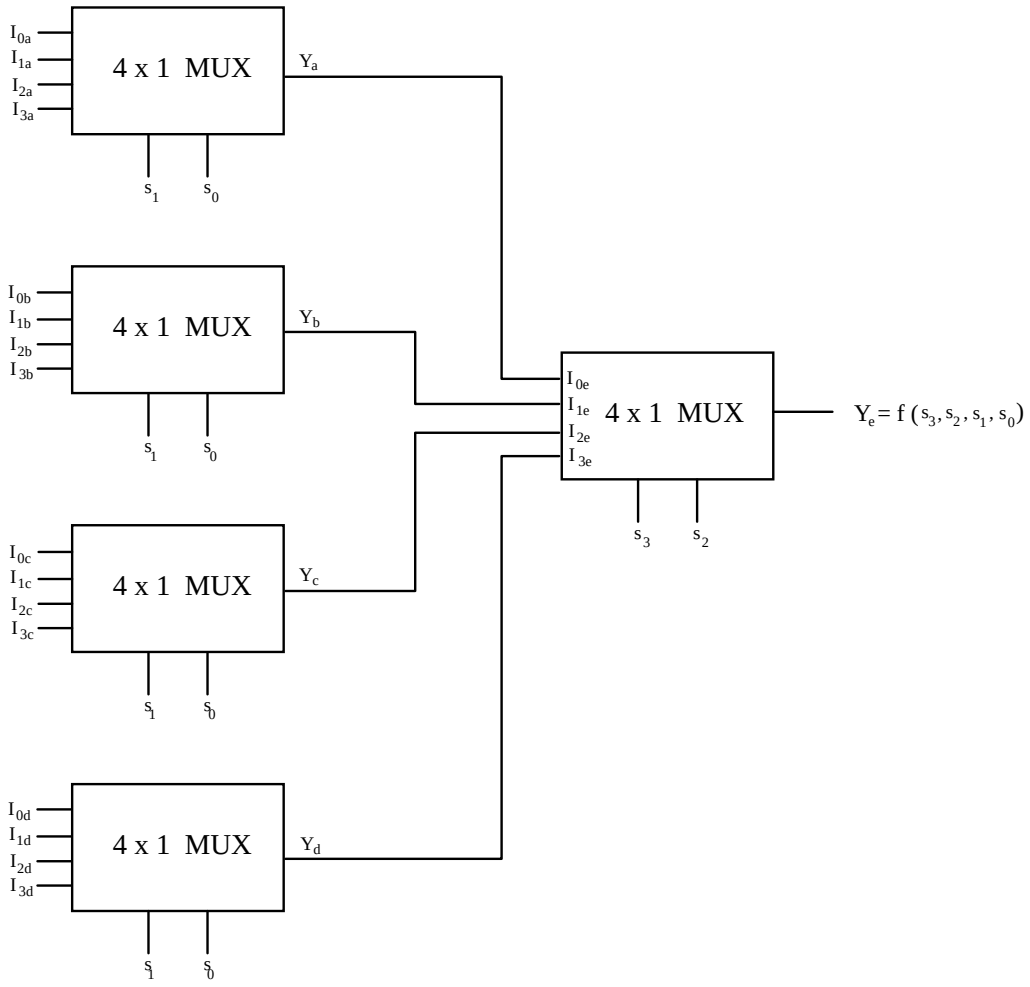
s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

We see from the input/output description of the 4×1 multiplexer above, line select variables s_1 and s_0 determine which one of the four input lines is sent to the output Y. The main application of digital multiplexers is for routing signals from many sources to a single destination. But because the table above is in exactly the same form as a combinational circuit truth table, it is also possible to use digital multiplexers for implementing Boolean logic functions.

For example, to realize the function $Y(s_1,s_0) = s_1 \oplus s_0$, one connects inputs $\{I_0, I_3\}$ to 0, and inputs $\{I_1, I_2\}$ to 1:



In order to realize a logical function of N variables in this manner, a single $2^N \times 1$ multiplexer is required. By using more than one $2^N \times 1$ multiplexer, however, logic functions having more than N inputs may be implemented. For example, the following circuit realizes a logic function having 4 inputs and 1 output using five 4×1 multiplexers:



Design Method

M1. Draw a truth table for functions $\{f_0, f_1, f_2, f_3\}$ in terms of input variables {A, B, C} when the relation between them is as follows:

$f_0 = 1$ when none of the input variables are equal to 1; otherwise, $f_0 = 0$.

$f_1 = 1$ when exactly one of the input variables is equal to 1; otherwise, $f_1 = 0$.

$f_2 = 1$ when exactly two of the input variables are equal to 1; otherwise, $f_2 = 0$.

$f_3 = 1$ when all three of the input variables are equal to 1; otherwise, $f_3 = 0$.

M2. Use K-maps to obtain simplified expressions for $\{f_0, f_1, f_2, f_3\}$ in terms of input variables {A, B, C}. Express your results in minimized SOP form.

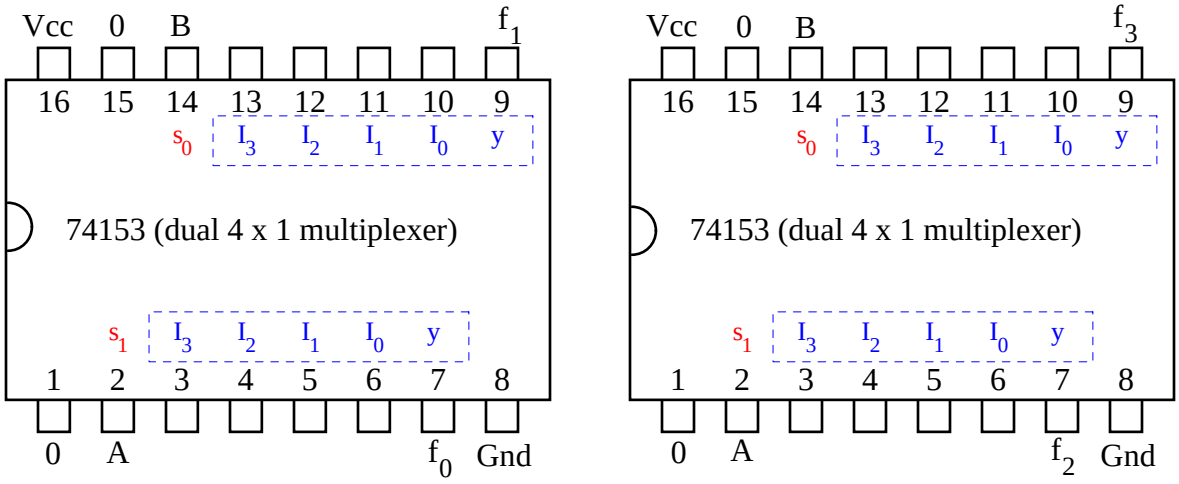
M3. Draw a schematic diagram of a digital logic circuit to implement functions $\{f_0, f_1, f_2, f_3\}$ according to the simplified SOP forms found above, using standard logic gates (no multiplexers). If necessary, make circuit-level gate transformations so that your circuit has gates only of the types and quantities that are included in your lab kit.

M4. You will next design and construct a circuit using four 4×1 digital multiplexers and one inverter to implement the same functions that were defined previously: $\{f_0, f_1, f_2, f_3\}$. Input variables A and B will serve as the multiplexer select line inputs, and one of the following will drive the data input lines I_0, I_1, I_2, I_3 for each multiplexer: {0, 1, C, C'}.

Based on the information in the truth table from M1, fill in the table below with one of the following in each square: {0, 1, C, C'}

	f_0	f_1	f_2	f_3
I_0				
I_1				
I_2				
I_3				

M5. Write one of {0, 1, C, C'} next to each unlabelled IC pin below to realize functions f_0 through f_3. Build this circuit and demonstrate it to your T.A. Use switches to select the states of the input variables, and LED's to display both input and output variable states.



(Refer to page 35 in the Becker lab manual. Connect both "Strobe" inputs 1G and 2G to ground, which enables the multiplexers for normal operation. The two multiplexers on this IC are both controlled by the same select lines. Be aware that the Function Table on p. 35 labels the most significant select bit s_1 as select input "B", and least significant select bit s_0 as select input "A" -- contrary to our variable labels in this experiment!)

Questions:

Q1. Using five 4×1 digital multiplexers in the same configuration as shown at the bottom of page 2, design a digital circuit that calculates the least significant bit of the sum of four binary numbers: X_1, X_2, X_3 and X_4 (i.e., calculate S_0 if $S_2S_1S_0 = X_1 + X_2 + X_3 + X_4$).

Q2. Repeat the same task, this time using one 16×1 digital multiplexer.

Q3. Repeat the same task, this time using only one 4×1 digital multiplexer. This time use some of the input variables to drive the multiplexer inputs {I_0 ... I_3}.

Q4. How many fewer IC's does the multiplexer-based circuit in M5 require than the direct implementation designed in M3?

Q5. How will incorrectly connecting the two input variables (e.g. connecting variable B to pin 2 and variable A to pin 14) affect the circuit function? Draw the resulting truth table that expresses {f_0, f_1, f_2, f_3} in terms of input variables {A, B, C} for this incorrectly-wired circuit.

University of Illinois at Chicago
Department of Electrical and Computer Engineering
ECE 265 - Introduction to Logic Design

Lab Experiment #7

Experimenting with the 555 Timer and Binary Counter (Intersection Traffic Signal)

Background

Oscillator using the 555 Timer

In its astable mode, the 555 Timer is a digital oscillator (produces a periodic clock signal). To see the oscillator in action, build the following circuit:

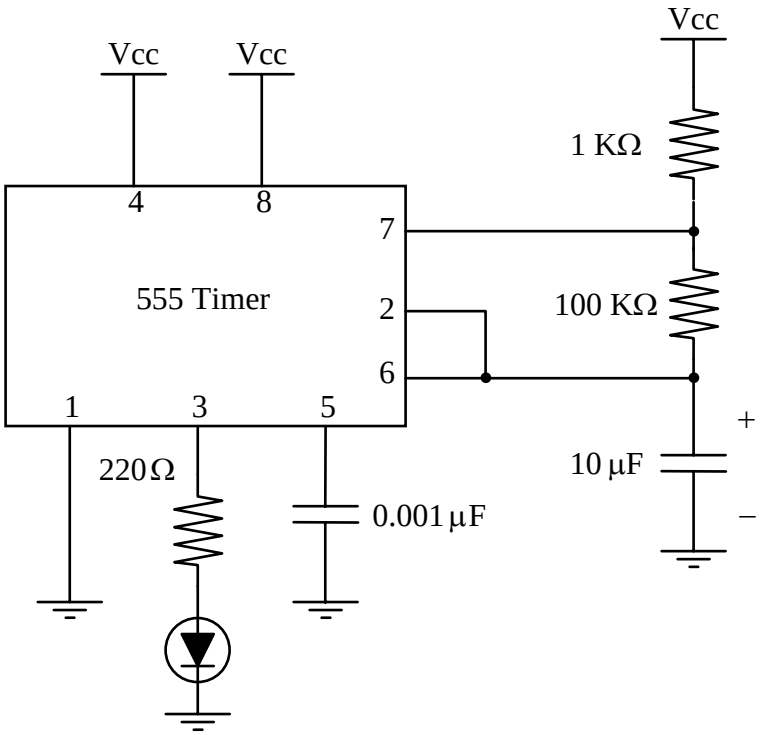


Figure 1.

(The 0.001 μF capacitor is the flat brown disc with label "**102**" = code for **10** × 10^**2** pF, whereas the 10 μF capacitor has a large cylindrical shape with voltage polarity markings)

You will notice that the LED driven by the output signal at pin 3 flashes on and off with a cycle time of approximately 1.5 seconds.

The 555 oscillator circuit functions this way: internally to the 555 timer, a flip-flop controls whether pin 7 is floating or at 0 volts. Assume that initially pin 7 is floating. The $10\mu F$ capacitor charges through the series combination of the $100\,K\Omega$ and $1\,K\Omega$ resistors until pin 6 senses that the capacitor voltage just exceeds $2/3\ Vcc$, and sets the internal flip-flop which causes pin 7 to go to 0 volts. Now the capacitor discharges through the $100\,K\Omega$ resistor until its voltage is just below $1/3\ Vcc$; this is sensed by pin 2 and the flip-flop is reset causing the cycle to repeat. Because the capacitor charges through $101\,K\Omega$ but discharges through $100\,K\Omega$, the flip-flop will be set slightly longer than it is set. The general formula for cycle time here is:

$$t_{period} = 0.695\,(1K\Omega + 2\cdot 100K\Omega)(10\mu F) = 1.4\ \text{sec}$$

What do you expect will happen to the clock period when the $10\mu F$ capacitor is replaced by a $1\mu F$ capacitor? When the $1\,K\Omega$ resistor is replaced by a $180K\Omega$ resistor? Make these changes to confirm your predictions.

Binary Counter

The pin 3 output of the 555 timer circuit shown in Figure 1 will serve as the input signal to a binary counter, as shown in Figure 2 below. Build this circuit using the lab manual as a reference guide for the 74193 binary counter IC. What count sequence do you see for output variables Q_D, Q_C, Q_B and Q_A?

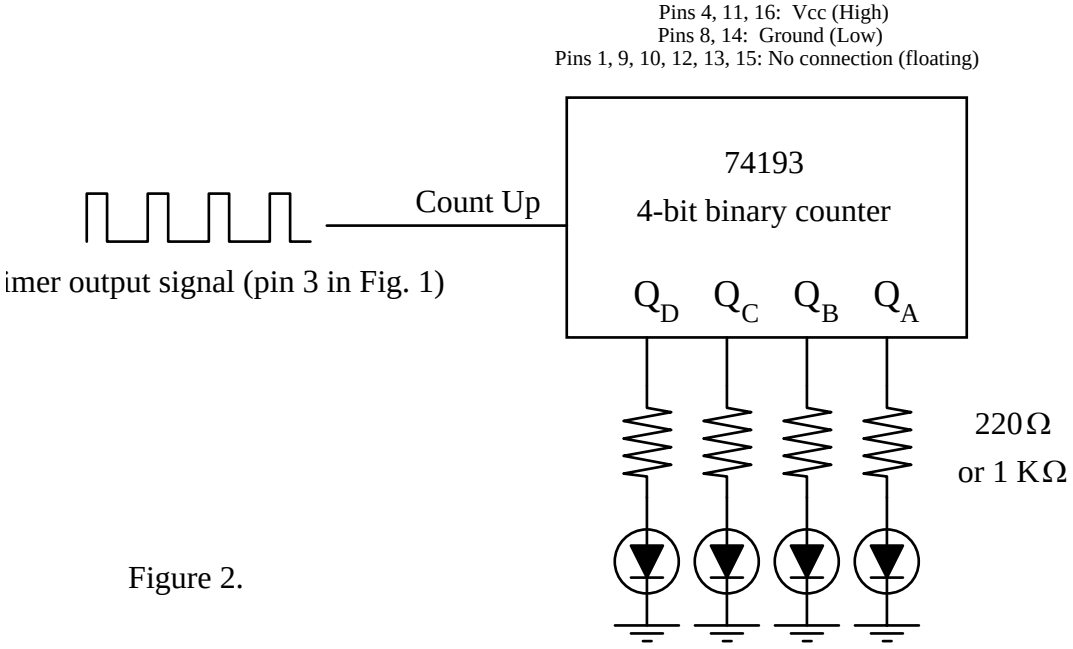


Figure 2.

Traffic Light Circuit Design

The goal of this experiment is to build a simulation of an intersection traffic signal, one that cycles between green-yellow states in the North/South directions while East/West displays red, and vice versa. Figure 3 below shows the physical arrangement of red, yellow and green LEDs that you are to use. These LEDs are driven by binary variables {R1,Y1,G1} for the North/South direction and {R2,Y2,G2} for the East/West direction, as shown in Figure 4.

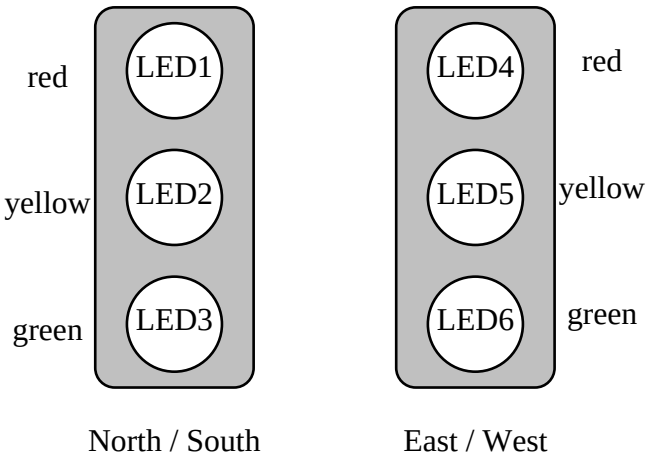


Figure 3.

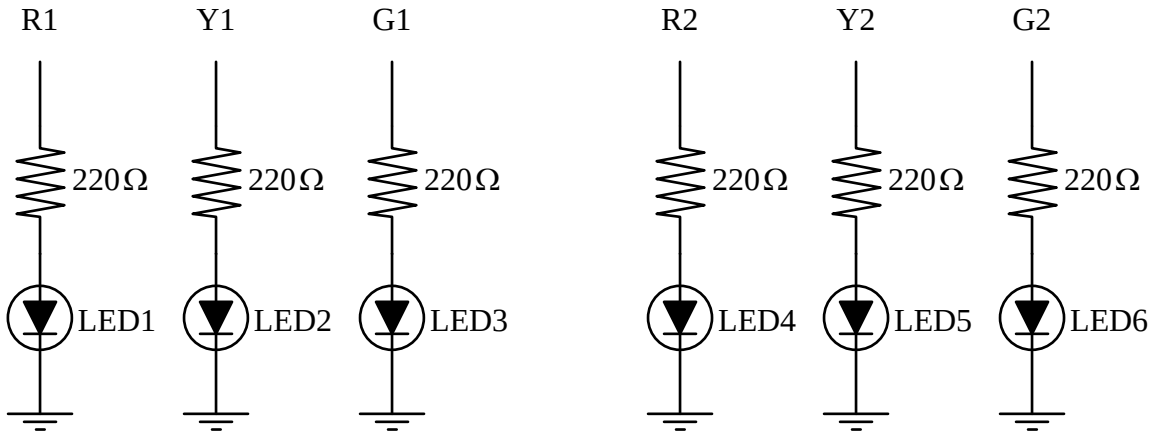


Figure 4.

The goal of this experiment is for you to convert the binary counter outputs {Q_D, Q_C, Q_B, Q_A} shown in Figure 2 to output variables {R1,Y1,G1,R2,Y2,G2}in order to simulate the operation of actual intersection traffic signal lights.

Design Method

M1. The table below shows the desired output variable {R1, Y1, G1, R2, Y2, G2} states as a function of binary count value. Write simplest SOP expressions for each of these output variables in terms of the binary counter output state variables $\{Q_D, Q_C, Q_B, Q_A\}$. Note that the value "Count" below is actually the decimal value of the counter state.

Count	R1	Y1	G1	R2	Y2	G2
0	0	0	**1**	**1**	0	0
1	0	0	**1**	**1**	0	0
2	0	0	**1**	**1**	0	0
3	0	0	**1**	**1**	0	0
4	0	0	**1**	**1**	0	0
5	0	**1**	0	**1**	0	0
6	0	**1**	0	**1**	0	0
7	0	**1**	0	**1**	0	0
8	**1**	0	0	0	0	**1**
9	**1**	0	0	0	0	**1**
10	**1**	0	0	0	0	**1**
11	**1**	0	0	0	0	**1**
12	**1**	0	0	0	0	**1**
13	**1**	0	0	0	**1**	0
14	**1**	0	0	0	**1**	0
15	**1**	0	0	0	**1**	0

M2. Design, draw and construct the complete circuit (using 4×1 digital multiplexers if necessary to simplify the required logic) that demonstrates the intersection traffic light, choosing resistor/capacitor combinations for the 555 timer circuit so that one traffic cycle lasts approximately15 seconds. Simulate the appearance of traffic signals by physically arranging your LED's on the breadboard as shown in Figure 3. Demonstrate this circuit to your T.A.

Questions:

Q1. Does your traffic light circuit have memory? Explain.

Q2. The circuit you have built has green on for 5 clock periods, followed by yellow on for 3. Write new simplified SOP expressions for {Y1, G1} in terms of the binary counter state variables $\{Q_D,Q_C,Q_B,Q_A\}$ had we instead made green on for 6 clock periods and yellow on for 2.

Q3. What does the state of binary counter output signal Q_D indicate regarding traffic flow?

Q4. To prevent serious traffic accidents from occurring in practice, traffic lights should never have one of {Y1, G1} signals lit at the same time that one of {Y2, G2} is lit. This event may occur, however, when there is an error condition in the digital logic circuit (such as a broken or shorted wire, e.g.). Can you think of any circuit design that will lessen the likelyhood of this error condition? Answer this question in words - no circuit design is required.

University of Illinois at Chicago
Department of Electrical and Computer Engineering
ECE 265 - Introduction to Logic Design

Lab Experiment #8

Edge-Triggered Flip-Flop Design

555 Timer IC - Monostable Mode

The second application of the 555 Timer IC that we will use the "monostable", or "one-shot" mode. Build the circuit below, that is also shown in the Becker Lab Manual:

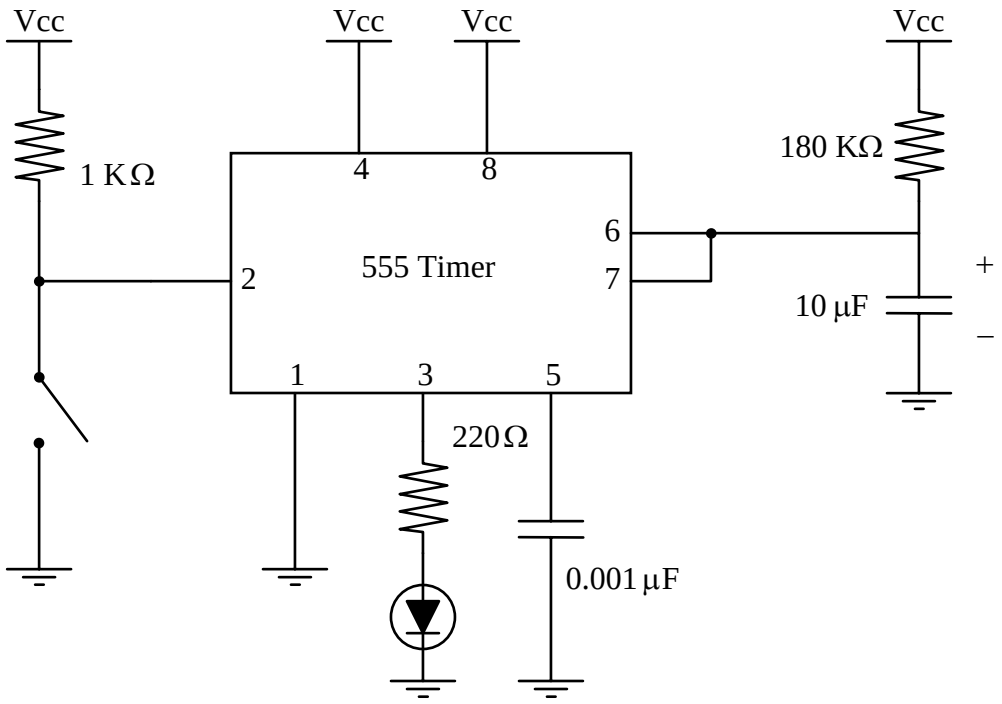


Figure 1.

Apply a short negative-going pulse to pin 2 by quickly closing and opening the switch. In response pin 3 should go high for approximately 2 seconds ($t = 1.1 \times 10\mu\text{F} \times 180\text{K}\Omega$). This behavior is called "pulse-stretching," and is useful for eliminating the effects of switch bounce on the output pulse waveform. We will use this circuit to generate a clean clock signal for testing flip-flops.

Edge-Triggered D Flip-Flop

Build this circuit:

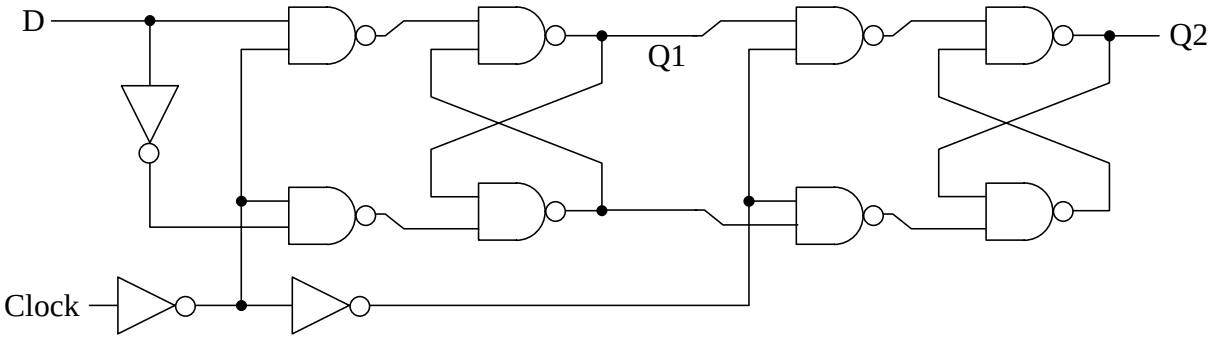
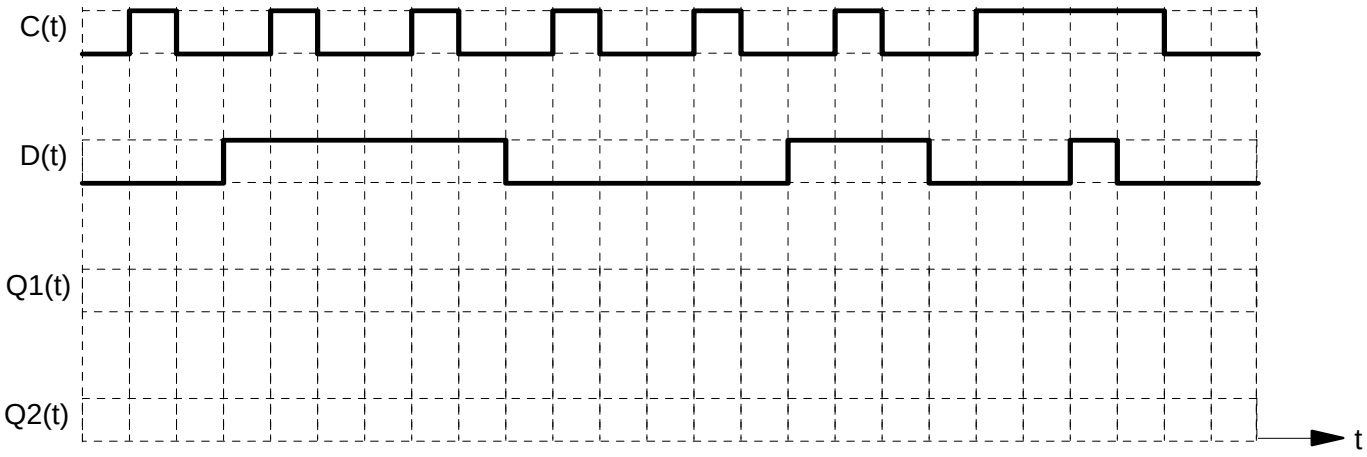


Figure 2

Control the voltage at D using a switch as in previous experiments, and display the logic levels at Q1 and Q2 using LEDs (with current-limiting resistors in series). Take the de-bounced Clock signal from pin 3 in the One-Shot circuit of Figure 1.

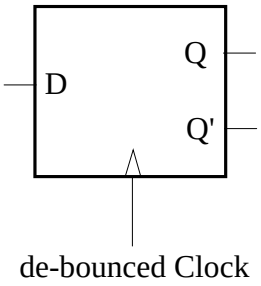
Complete the graph below by drawing the waveforms for Q1 and Q2 in response to the signals D and Clock that you should generate according to the waveforms that are given. Ask your T.A. to verify that your measured waveforms are correct. Is this a positive-edge or negative-edge triggered flip-flop?



Design Method

M1. Write the characteristic equation of a T flip-flop.

M2. Based on this characteristic equation, add logic gates to the D flip-flop below to create an edge-triggered T flip-flop:



M3. Using the D flip-flop circuit in Figure 2 as its core, build the edge-triggered T flip-flop that you have designed above and demonstrate its operation to your T.A. Your lab report should include a schematic diagram of the complete circuit on the gate level.

Questions:

Q1. Perform circuit-level manipulations on the edge-triggered D flip-flop shown in Figure 2 so that only 2-input NOR gates are used (inverters are implemented as NOR gates with one input grounded); it should be functionally equivalent to the original circuit.

Q2. In a similar manner, design and neatly draw the gate-level diagram for a negative-edge-triggered J-K flip-flop using only 2-input NAND gates (inverters are implemented as NAND gates with one input at Vcc). Show every step of the design process.

Q3. It is critically important to have a de-bounced clock signal when demonstrating the T flip-flop, but this is not as important when demonstrating the D flip-flop. Why is this so? (*Hint: consider what happens when T=1*)

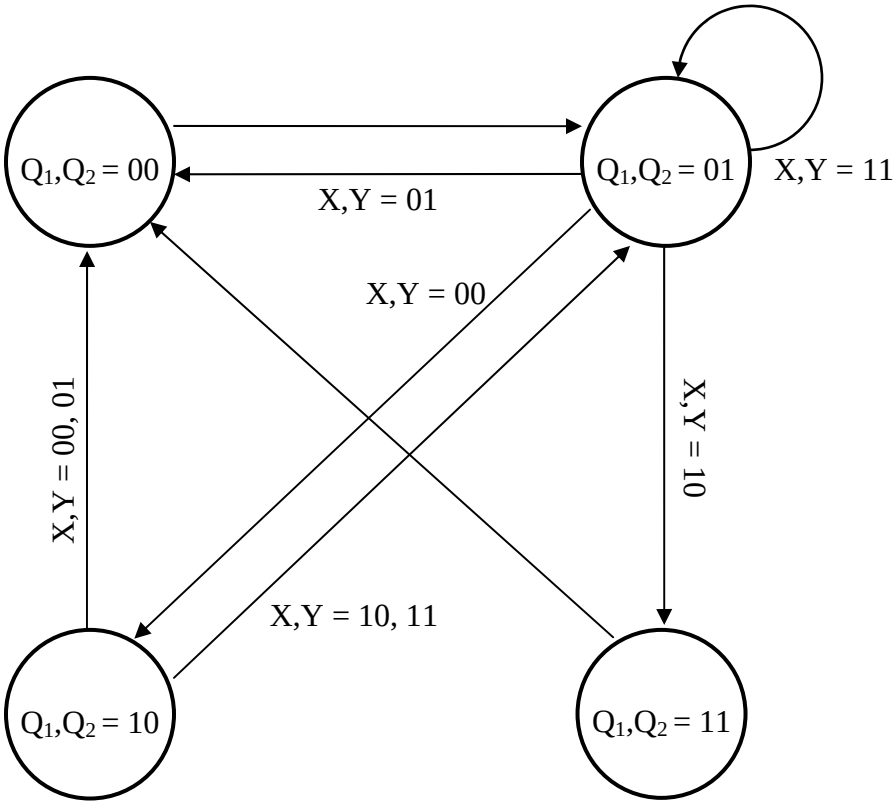
University of Illinois at Chicago
Department of Electrical and Computer Engineering
ECE 265 - Introduction to Logic Design

Experiment # 9

Electronic Light Dimmer

Introduction

In this experiment you will design and build a clocked sequential circuit based on a specification of a Moore Finite State Machine state variable transition diagram. Your circuit will have input variables {X,Y} and output variables $\{Q_1,Q_2\}$. Q_1 and Q_2 are also the state variables, for they are taken as the outputs of two JK flip-flops in the circuit. The desired state transition diagram is shown below:



Design Method

M1. Transfer the information from the state transition diagram to the state transition table below. Also fill in the values of J_1, K_1, J_2, and K_2 that are required to produce the desired changes in state from $\{Q_1,Q_2\}$ to $\{Q_1^+,Q_2^+\}$:

X	Y	Q_1	Q_2	Q_1^+	Q_2^+	J_1	K_1	J_2	K_2
0	0	0	0						
0	0	0	1						
0	0	1	0						
0	0	1	1						
0	1	0	0						
0	1	0	1						
0	1	1	0						
0	1	1	1						
1	0	0	0						
1	0	0	1						
1	0	1	0						
1	0	1	1						
1	1	0	0						
1	1	0	1						
1	1	1	0						
1	1	1	1						

M2. To convert flip-flop outputs {Q_1,Q_2} and user switch-generated inputs {X,Y} into flip-flop excitation signals {J_1, K_1, J_2, and K_2}, it is best to use the multiplexer approach as in previous experiments. Using inputs {X,Y} as the multiplexer select lines, determine the inputs for each of the four MUX's and write them in the table below:

X	Y	Q_1	Q_2	MUX #1 (J_1)	MUX #2 (K_1)	MUX #3 (J_2)	MUX #4 (K_2)
0	0	0	0				
0	0	0	1	I_0 =	I_0 =	I_0 =	I_0 =
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1	I_1 =	I_1 =	I_1 =	I_1 =
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1	I_2 =	I_2 =	I_2 =	I_2 =
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1	I_3 =	I_3 =	I_3 =	I_3 =
1	1	1	0				
1	1	1	1				

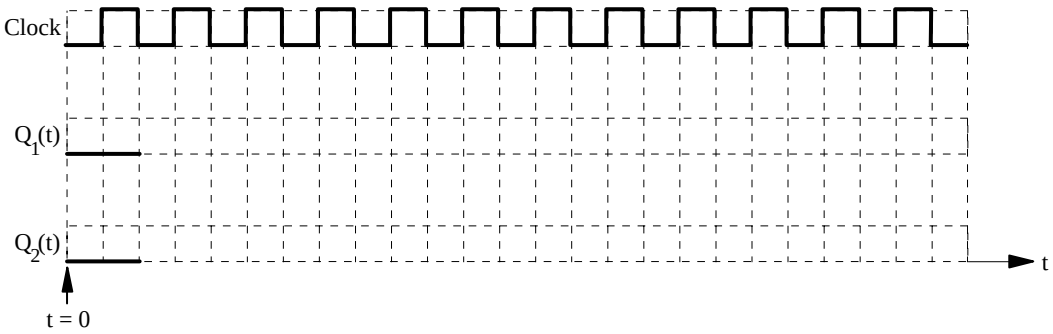
M3. When testing your circuit, configure a 555 Timer IC clock circuit having a clock period of approximately 1 sec. This will serve as your clock signal input to both JK flip-flops. Use switches and pull-up resistors to generate input variables X and Y; include 2 LED's to display the output variables Q_1 and Q_2.

M4. For the demonstration use two 1K resistors and one 1 uF capacitor in the 555 timer so that LED flashing is not detected by the eye. You may need to turn off the room lights to see subtle changes in brightness level of the LED. Show your T.A. that the intensity of Q_2 varies with the state of X,Y. (This programmable duty-cycle method is commonly used in light dimmer and d.c. motor speed control circuits.)

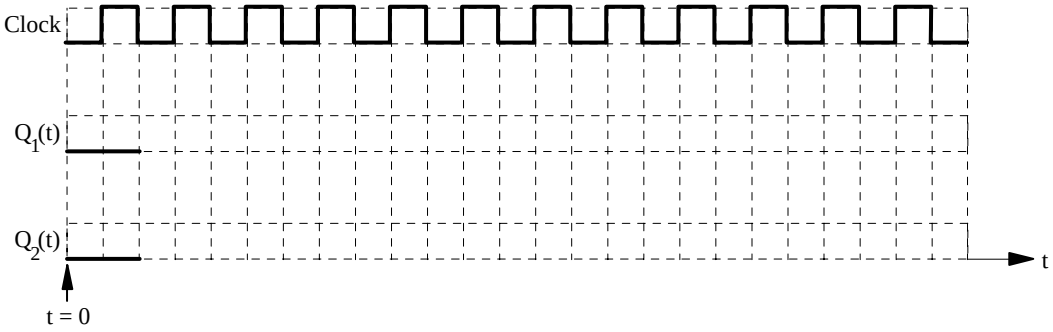
Questions

Q1. Based on the state diagram, draw waveforms for $Q_1(t)$ and $Q_2(t)$ in response to the clock signal. Assume that both Q_1 and Q_2 are initially equal to 0, and that *negative* edge-triggered JK flip-flops are used:

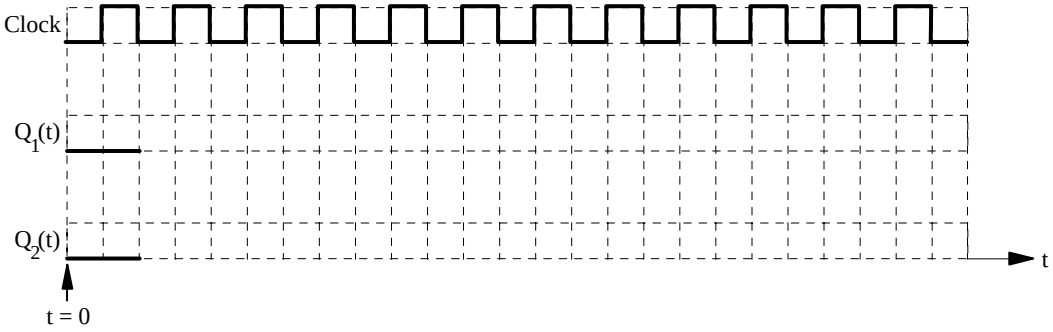
XY = 00:



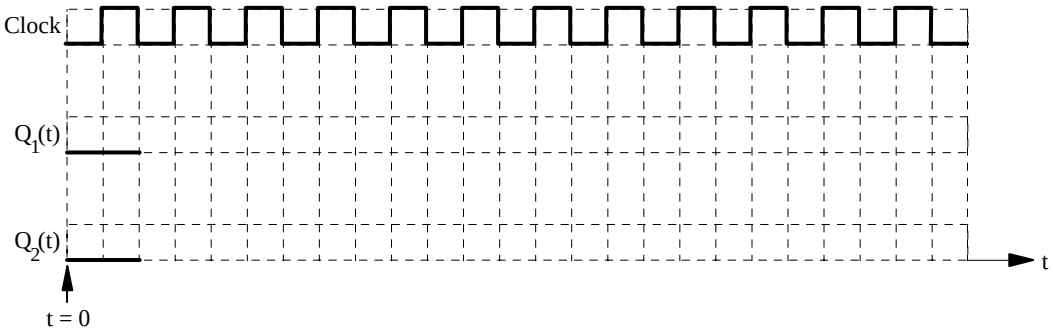
XY = 01:



XY = 10:



XY = 11:



Q2. What frequency of clock did you use for testing, and for demonstration? (Frequency = 1/Period)

Q3. What fraction of the time is Q_2 is high (its duty cycle) for each combination of X and Y, assuming that the count is in its endless loop? Express duty cycle in percent.

Q4. Design and neatly draw the state transition diagram for a light dimmer circuit similar to the one shown on the first page, except for the fact that duty cycle vs. input variables is specified as follows:

X	Y	Duty cycle of Q_2
0	0	0%
0	1	25%
1	0	50%
1	1	75%

Q5. Assuming there are only two state variables in our Finite State Machine, how many different duty cycles (of output signal) may it produce? Express duty cycle in percent.

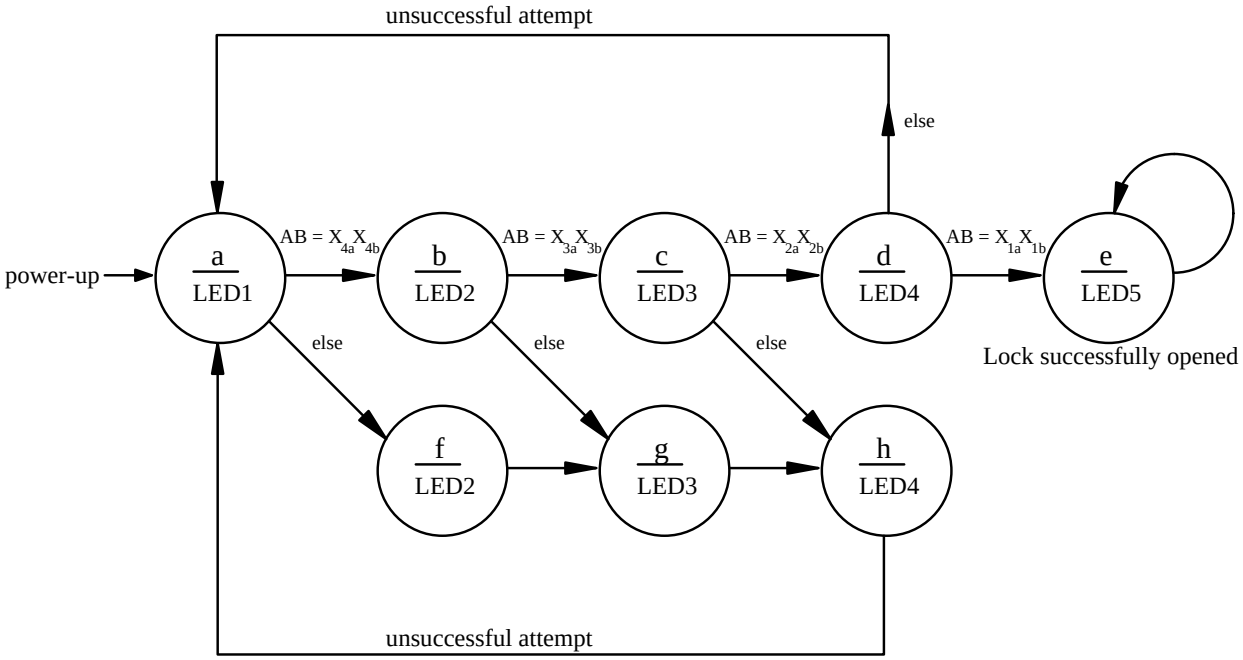
University of Illinois at Chicago
Department of Electrical and Computer Engineering
ECE 265 - Introduction to Logic Design

Experiment # 10

Sequential Circuit Design -- Digital Combination Lock

Introduction

In this experiment you will design and build a clocked sequential circuit that performs the function of a digital combination lock. Because the 8-bit code to open the lock is derived from the last 4 digits of your Social Security number (student ID#), each student will design and demonstrate one of the 256 different versions of this lock. The state diagram of the digital combination lock is shown below:



Input binary variables A and B are generated using switches as usual; these switches are flipped to establish the correct values of A and B in preparation for the next state transition.

The circuit will use the JK flip-flops that are provided in your lab kit. All flip-flop clock inputs are to be taken from the 555 Timer operating as a pulse generator ("one-shot") that is shown on page 40 of the Becker lab manual. The clock circuit is a debounced switch

that you control – creating a pulse on the master clock line only after you've had time to change code bits A,B in preparation for the next state transition. If you want to be fancy, try the touch-switch circuit shown in section 7.32 (10MΩ or higher resistors are available in the Senior Design Lab).

The combination to open your lock will be a sequence of four 2-bit binary codewords: AB = $X_{4a}X_{4b}$; $X_{3a}X_{3b}$; $X_{2a}X_{2b}$; X_{1a} X_{1b}. Let $(Y_4Y_3Y_2Y_1)_{10}$ be the last four digits of your student ID number. Each codeword $X_{Na}X_{Nb}$ is the least-significant two bits of decimal digit Y_N, when it is represented in binary. Example: when the last four SS# digits are 1583, the binary code to open the lock is: 01; 01; 00; 11.

Five LEDs are to provide an indication that the lock is in the following states:
LED1: state a; LED2: states b,f ; LED3: states c,g; LED4: states d,h; LED5: state e.

<u>Include this in your Lab Report:</u>

1. What are the last four digits of your student ID number?

(base 10)

2. Based on this, what is your digital lock combination?

(base 2)

3. Based on the state diagram shown previously and your particular digital lock combination, fill in the state table below (please use these state variable assignments!):

	Q_1	Q_2	Q_3	A	B	Q_1^+	Q_2^+	Q_3^+	J_1	K_1	J_2	K_2	J_3	K_3
state a	0	0	0	0	0									
	0	0	0	0	1									
	0	0	0	1	0									
	0	0	0	1	1									
state b	0	0	1	0	0									
	0	0	1	0	1									
	0	0	1	1	0									
	0	0	1	1	1									
state c	0	1	0	0	0									
	0	1	0	0	1									
	0	1	0	1	0									
	0	1	0	1	1									
state d	0	1	1	0	0									
	0	1	1	0	1									
	0	1	1	1	0									
	0	1	1	1	1									
state e	1	0	0	0	0									
	1	0	0	0	1									
	1	0	0	1	0									
	1	0	0	1	1									
state f	1	0	1	0	0									
	1	0	1	0	1									
	1	0	1	1	0									
	1	0	1	1	1									
state g	1	1	0	0	0									
	1	1	0	0	1									
	1	1	0	1	0									
	1	1	0	1	1									
state h	1	1	1	0	0									
	1	1	1	0	1									
	1	1	1	1	0									
	1	1	1	1	1									

4. Find simplified expressions for flip-flop excitation signals J_1, K_1, J_2, K_2, J_3, K_3:

J_1 =

K_1 =

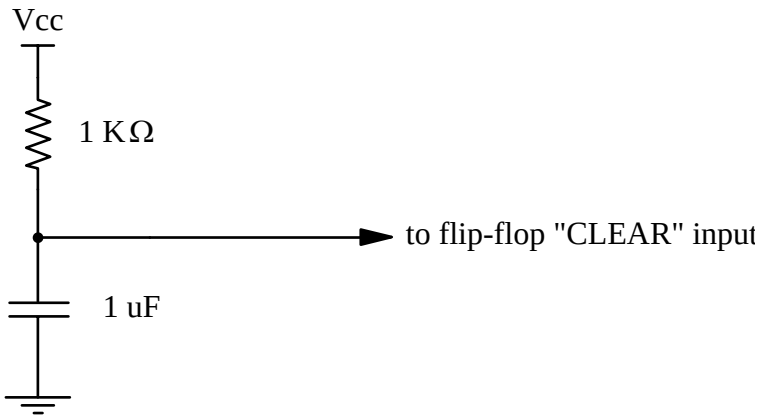
J_2 =

K_2 =

J_3 =

K_3 =

5. Because of the complexity of this circuit, it is recommended that you build and test it in stages. First, hook up three additional switches to simulate flip-flop outputs $\{Q_1,Q_2,Q_3\}$ in order to build and troubleshoot the circuitry that produces excitation signals J_n,K_n. Next, build and test the 555 Timer one-shot circuit. At this point replace the switches simulating $\{Q_1,Q_2,Q_3\}$ with the actual JK flip-flop outputs, and drive the flip-flops with the clock and excitation signals. Instead of tying the JK flip-flop "CLEAR" pins directly to Vcc, use the circuit below (this will momentarily clear the flip-flops upon first hooking up the power supply, for the purpose of making sure that the circuit begins in state "a"):



Finally, when this sub-circuit has been fully tested using both correct and incorrect lock codes, add the circuitry necessary to obtain the five LED output signals from $\{Q_1,Q_2,Q_3\}$.

Neatly draw a complete schematic diagram of the resulting circuit and include it in your lab report. Demonstrate the operation of the lock circuit to your T.A. You will be asked to demonstrate the effect of both correct and incorrect lock codes.

This experiment may require two weeks to complete. Make sure that your batteries are fully charged. Be deliberate in constructing and then testing your circuit in stages, as described previously. Some social security numbers will lead to more complicated designs, but that's life!

Also to be included in the Lab report:

6a. Show the state transition diagram for your circuit, labeled with your own combination code bits.

6b. Show the state transition table, and all K-maps used to find the J,K input signals to the three flip-flops.

6c. Neatly draw a schematic diagram of the entire circuit that you have built and plan to demonstrate to the T.A. (showing every component: 555 timer circuit, flip-flops, MUXs, logic gates, switches, LEDs, and resistors).

6d. In this application why is it necessary to use the 555 Timer one-shot circuit instead of directly creating a clock signal using a switch?

6e. Explain the advantage of having all incorrect codes result in passing through state "h" of the state diagram, rather than directly leading back to state "a".

6f. In Part 3 you were asked to use specific state variable assignments for each of the eight states of the digital combination lock. Looking back, why do you think this particular state assignment was a good idea? (For example, why is it useful to have state b represented by $Q_1Q_2Q_3 = 001$ and state f represented by $Q_1Q_2Q_3 = 101$, etc.?)

6g. Your digital combination lock has 4 codewords of 2 bits each. A similar digital combination lock is being designed using N codewords of 8 bits each. What is the minimum value of N needed to ensure that a code of N randomly-selected codewords is likely to succeed at most once in 1 trillion attempts, on the average?