

Lab Exercise #1

Assertion Levels, Voltage Tables, and Truth Tables

Objectives

After completing this experiment, you will

- 1) be able to explain the difference between high-asserting and low-asserting inputs and outputs;
- 2) be able to determine experimentally a voltage table for an unknown digital circuit; and
- 3) be able to determine experimentally a truth table for an unknown digital circuit when given the assertion levels of the inputs and outputs.

Pre-lab

There is no prelab for this experiment.

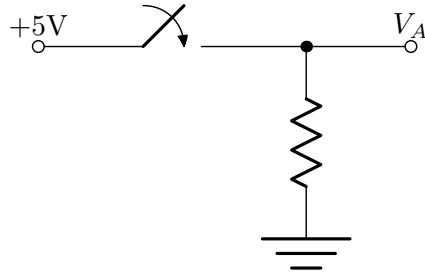
Introduction

Every digital system has input signals and output signals that may assume one of two values. The logical values of the signal are usually defined as “1 and 0”. Some people refer to “1” as “asserted” and refer to “0” as “deasserted”. If the digital system is electrical, the input signals and output signals are represented by voltages or currents. For our purposes, let’s stick with voltages. A “1” or “asserted” variable will be represented by one particular voltage while a “0” or “deasserted” variable will be represented by a different voltage. One of the two possible voltages, which we will call “H” is higher than the other, which we will call “L”. We will be working with a family of integrated circuits known as TTL, for which we may assume that “H” is 5 V and that “L” is 0 V. In some parts of our circuits, “H” represents a “1”, while in other parts, “H” represents a “0”. It is up to the circuit designer to keep track of what “H” means on any given signal wire. A signal where “H” represents a “1” and “L” represents a “0” is called a “high asserting” signal; a signal where “H” represents a “0” and “L” represents a “1” is called a low asserting signal.

In this course, we will use switches to control the inputs of our digital systems. Let’s assume that a switch in the “on” or “asserted” position always means “1”. In the first part of the lab, we will build a circuit that produces a high asserting signal that tells us whether a particular switch is asserted (on) or deasserted (off). For the second part of the lab, we will build a circuit that produces a *low asserting* signal that tells us the position of the switch. Finally, for the last part of the experiment, we will apply our high-asserting and low asserting switches to some integrated circuits to construct both voltage tables (Hs and Ls) and logic truth tables (0s and 1s) for these devices. We will determine these truth tables experimentally.

Procedure

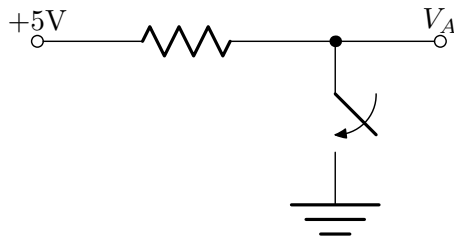
- 1) Construct *two* circuits with high asserting outputs; each of the two circuits should be constructed as shown below:



A circuit with a high asserting output indicating the position of the switch

Use switches numbered “1” and “2” from a DIP switch package. The DIP switch package should straddle the channel in the middle of the breadboard.

- 2) Connect the output of this circuit to one of the “8 channel logic monitor” LEDs on the Cadet Masterlab.
- 3) Verify that your circuit has a high asserting output. The Cadet Masterlab logic monitors will glow red for a “high” and will glow green for a “low.” If they fail to glow at all, you have connected something incorrectly.
- 4) Construct *two* circuits with *low* asserting outputs; each of the two circuits should be constructed as shown below:



A circuit with a low asserting output indicating the position of the switch

Use switches numbered “4” and “5” from a DIP switch package.

- 5) Connect the output of this circuit to one of the “8 channel logic monitor” LEDs on the Cadet Masterlab.
- 6) Verify that your circuit has a low asserting output.
- 7) Place a 74LS00 chip across the channel in your breadboard and connect power (+5V) from row 2 of the power strip to pin 14 of the 74LS00 (your instructor will show you how to find pin 14). Connect ground (0 V) from row 3 of the power strip to pin 7 of the 74LS00. Connect the high asserting output of the switch numbered 1 to the input pin 1 of the 74LS00 and connect the high asserting output of the switch numbered 2 to the input

pin 2 of the 74LS00. Assume the output on pin 3 of the 74LS00 is *low asserting* and complete the following truth table using 0s and 1s to show the correct output:

SW1	SW2	Output
0	0	
0	1	
1	0	
1	1	

- 8) Disconnect the output of switch 1 and switch 2 and instead connect the *low asserting* output of switch 4 and switch 5 to pin 1 and pin 2 of the 74LS00. Assume the output on pin 3 of the 74LS00 is *high asserting* and complete the following truth table using 0s and 1s to show the correct output:

SW4	SW5	Output
0	0	
0	1	
1	0	
1	1	

- 9) Repeat the last two steps using a 74LS02 instead of a 74LS00. *The pins on the 74LS02 are different, however.* Pins 2 and 3 are the input pins and pin 1 is the output pin. As in step 7, first connect the high asserting switch outputs to the input pins 2 and 3 assuming the output on pin 1 is low asserting. Then, as in step 8, connect the low asserting switch outputs to the input pins 2 and 3 assuming the output on pin 1 is high asserting.

Complete the following table for the high asserting inputs and low asserting output combination:

SW1	SW2	Output
0	0	
0	1	
1	0	
1	1	

Complete the following table for the low asserting input and high asserting output combination:

SW4	SW5	Output
0	0	
0	1	
1	0	
1	1	

- 10) Use the information gathered in this exercise to complete the following *voltage tables* for the 74LS00 and the 74LS02:

Voltage table for the 74LS00

V_{SW1}	V_{SW2}	V_{Output}
L	L	
L	H	
H	L	
H	H	

Voltage table for the 74LS02

V_{SW1}	V_{SW2}	V_{Output}
L	L	
L	H	
H	L	
H	H	

Review Questions

- 1) For a given circuit, the voltage table is always the same. (T/F)
- 2) For a given circuit, the truth table depends on the _____ of the input and output signals.

Lab Exercise #2

Analyzing, Building and Troubleshooting Digital Circuits

Objectives

After completing this experiment, you will

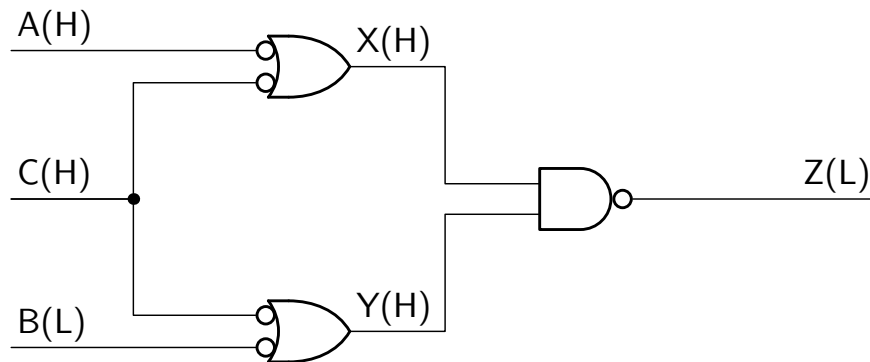
- 1) be able to write expression for the intermediate and final output variables in a combinational logic circuit;
- 2) be able to determine the truth table for a combinational logic circuit by analyzing the circuit diagram; and
- 3) be able to build and test an implementation for a logic diagram using standard 7400 series chips

Pre-lab

There is no prelab for this experiment.

Procedure

This experiment pertains to the following logic circuit:



- 1) Write an expression for X, Y, and Z in terms of A, B, and C
 - a) $X =$ _____
 - b) $Y =$ _____
 - c) $Z =$ _____

2) Complete the following truth table:

A	B	C	X	Y	Z

- 3) Label the circuit diagram with the part numbers of the integrated circuits you intend to use to implement each gate. On each gate, label the pin numbers for the integrated circuit you will use.
- 4) Build high asserting switches for A and C and a low asserting switch for B. Use a single DIP switch for A, B, and C. Use switch #1 for A, switch #2 for B, and switch #3 for C.
- 5) Build and test the circuit. Use LEDs (either high-asserting or low-asserting, as appropriate) to monitor X, Y, and Z. If you have trouble obtaining the truth table you determined in step 2, trace any incorrect signals backward to their point of origin. Resolve any discrepancies between your table and your circuit.

Lab Exercise #3

Designing a Digital Circuit to Implement a Specified Logic Function

Objectives

After completing this experiment, you will

- 1) be able to synthesize a given logic function using 7400 series integrated circuits;
- 2) be able to determine the truth table for a given logic function;
- 3) be able to build and test an implementation of a given logic function using specified 7400 series integrated circuits.

Pre-lab

- 1) Using only 7400, 7404, and 7410 integrated circuits, draw a circuit diagram that implements the logic function:

$$F = (A \vee B) \wedge (\overline{C} \vee \overline{D}) \wedge (\overline{A} \vee \overline{D})$$

Assume that A, B, and C are high asserting signals, and that D and F are low asserting signals.

Label your circuit diagram with chip numbers and pin numbers. Label the output of every gate with the correct logical expression representing its output along with the correct assertion level.

- 2) Repeat using only 7402, 7404, and 7427 integrated circuits. Label your circuit diagram with chip numbers and pin numbers.

Lab Exercise #4

Designing a Digital Circuit to Implement a Specified Truth Table

Objectives

After completing this experiment, you will be able to synthesize a circuit that implements a specified truth table using 7400 series integrated circuits.

Pre-lab

- 1) Determine a *minimal* sum-of-product (SOP) representation for a function that implements the following truth table:

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Hint: Use a K-map to obtain the minimal representation

- 2) Determine a *minimal* product-of-sum (POS) representation for a function that implements the previous truth table.
- 3) Select either your SOP or your POS representation for implementation
- 4) Draw a circuit diagram for your implementation using standard parts (e.g., 7400, 7402, 7404, 7410, etc.).

Assume that A, B, and C are high asserting signals, and that D and F are low asserting signals.

Label your circuit diagram with chip numbers and pin numbers. Label the output of every gate with the correct logical expression representing its output along with the correct assertion level.

Procedure

- 1) Construct high or low asserting switches to use for A, B, C, and D as appropriate. Use a single DIP switch for A, B, C, and D. Use switch #1 for A, switch #2 for B, switch #3 for C, and switch #4 for D.
- 2) Build, troubleshoot, and test the circuit.
- 3) Use LEDs to monitor the output variable.
- 4) Demonstrate your circuit to your instructor when you have it working.

Lab Exercise #5

Arithmetic circuits—a two-bit adder

Objectives

After completing this experiment, you will be able

- 1) to synthesize a full and half adder circuit from truth tables
- 2) to combine single-bit digital circuits (in this case a full and half adder) to produce a multi-bit digital circuit.

Pre-lab

- 1) Determine a *minimal* sum-of-product (SOP) or product-of-sum (POS) representation for a function that implements the following truth table for a full adder:

A	B	C _{in}	S	C _{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Hint: Use a K-map to obtain the minimal representation. Hint, you may want to use an exclusive-or function. When you see $(A \wedge \bar{B}) \vee (\bar{A} \wedge B)$, you may rewrite that as $A \text{ XOR } B$.

In this table, S represents the one bit sum and C_{out} represents the carry.

- 2) Repeat for the truth table of a half adder, which is similar to the full adder, but doesn't have an input carry:

A	B	S	C _{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- 3) Draw a circuit diagram for your implementation of each circuit using standard parts (e.g., 7400, 7402, 7404, 7410, etc.). You may want to use a 7486 for its XORs functions.

Assume that A, B, and C_{in}, S and C_{out} are all high asserting signals.

Label your circuit diagram with chip numbers and pin numbers. Label the output of every gate with the correct logical expression representing its output.

Procedure

- 1) Build, troubleshoot, and test the two circuits (the full adder and the half adder) independently. You may use the pre-built switches at bottom of the Cadet MasterLab kits, which are high-asserting, as input switches. To test the circuits independently, you will want two switches as inputs for the half adder (call them A_0 and B_0) and three switches as inputs for the full adder (A_1 , B_1 , and C_{in}).
- 2) When you are sure that both circuits are working, remove the carry input of the full adder from its switch and connect the carry output of the half adder to the carry input of the full adder. Verify that your two-bit adder works as expected. *Reconnect the switches as follows:* Use switch #1 for A_1 , switch #2 for A_0 , switch #3 for B_1 , and switch #4 for B_0 . The 1 subscripts indicate connections to the full adder and the 0 subscripts indicate connections to the half-adder. Your two-bit adder will add two numbers. The first number is represented by the left two switches and the second number is represented by the right two switches.
- 3) Use one of the BCD seven segment LED displays to display a numeric digit corresponding to your two-bit output. Connect the least significant S bit to the connector labeled A , the most significant S bit to the connector labeled B , and the most significant carry to the connector labeled C .
- 4) Demonstrate your circuit to your instructor when you have it working.

Lab Exercise #6

Decoders

Objectives

After completing this experiment, you will be able to synthesize a truth table using a decoder.

Introduction

A decoder may be used to generate truth tables quite easily because it generates *all* minterms simultaneously. In other words, a three-input decoder having inputs A , B , and C , *simultaneously* produces the minterms as shown in the following table

Table 1—Relationship between input variables and output variables in a decoder

Output	Minterm Expression
Y0	$\bar{C} \wedge \bar{B} \wedge \bar{A}$
Y1	$\bar{C} \wedge \bar{B} \wedge A$
Y2	$\bar{C} \wedge B \wedge \bar{A}$
Y3	$\bar{C} \wedge B \wedge A$
Y4	$C \wedge \bar{B} \wedge \bar{A}$
Y5	$C \wedge \bar{B} \wedge A$
Y6	$C \wedge B \wedge \bar{A}$
Y7	$C \wedge B \wedge A$

The minterms may be used to synthesize any sum-of-products (SOP) function directly from the truth table without using a K-map.

A typical decoder circuit is the 74LS138, which has the pinout diagram shown in Figure 1. *Pay close attention to the assertion levels shown in Figure 1. Note: the G1, G2A, and G2B pins signals are enable inputs and should all be set to a logical one.*

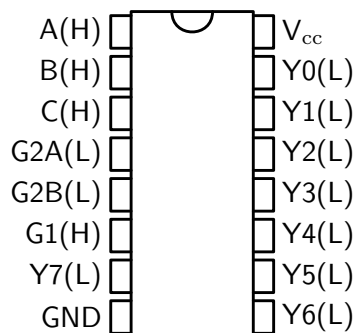


Figure 1—Pinout for 74LS138 circuit

Pre-lab

- 1) Use a 74138 decoder circuit to implement the following truth table for a full adder:

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

In this table, S represents the one bit sum and C_{out} represents the carry.

- 2) Draw a circuit diagram for your implementation using the 74LS138 along with a few additional additional gates (no more than two) obtained from standard parts (e.g., 7400, 7402, 7404, 7410, etc.).

Assume that A, B, and C_{in} , S and C_{out} are all high asserting signals.

Label your circuit diagram with chip numbers and pin numbers. Label the output of every gate with the correct logical expression representing its output.

Procedure

- 1) Build, troubleshoot, and test *two* full adder circuits. You may use the pre-built switches at bottom of the Cadet MasterLab kits, which are high-asserting, as input switches. You will need three inputs for the first full adder (call them A_0 , B_0 , C_0) and three inputs for the second full adder (A_1 , B_1 , and C_{in}).
- 2) When you are sure that both circuits are working, remove the carry input of the full adder from its switch and connect the carry output of the half adder to the carry input of the full adder. Verify that your two-bit adder works as expected. *Reconnect the switches as follows:* Use switch #1 for A_1 , switch #2 for A_0 , switch #3 for B_1 , and switch #4 for B_0 . The 1 subscripts indicate connections to the full adder and the 0 subscripts indicate connections to the half-adder. Your two-bit adder will add two numbers. The first number is represented by the left two switches and the second number is represented by the right two switches.
- 3) Use the BCD seven segment LED displays to display a numeric digit corresponding to your two-bit output. Connect the least significant S bit to the connector labeled A, the most significant S bit to the connector labeled B, and the most significant carry to the connector labeled C.
- 4) Demonstrate your circuit to your instructor when you have it working.

Lab Exercise #7

Introduction to Flip-Flops

Objectives

After completing this experiment, you will be able to design, build, and test circuits containing flip-flops, given a state-transition table.

Introduction

A flip-flop is a memory device. Its current state depends not only on its current inputs, but also on previous inputs that have been applied to it. The JK flip-flop has two control inputs, J and K. The JK flip-flop can change its output *only* on the edge of the CLK signal, when the CLK signal changes from 0 to 1. A JK flip-flop changes its state based on the value of its J and K inputs at the moment it's clocked. The transitions that occur appear in Table 1. The symbol for a JK flip-flop is shown in Figure 1.

Figure 1—Symbol for JK flip-flop

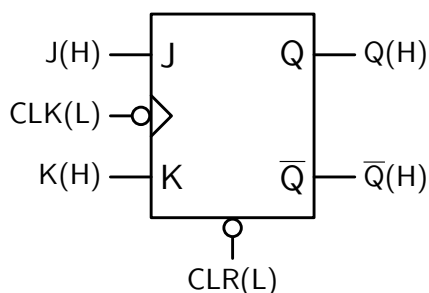


Table 1—Transition table for analysis of JK flip-flops

Current State		Inputs		Next State
Q	J	K	Q ⁺	
0	0	0	0	
0	0	1	0	
0	1	0	1	
0	1	1	1	
1	0	0	1	
1	0	1	0	
1	1	0	1	
1	1	1	0	

This table is useful for telling what a flip-flop will do when the current state and inputs are known. It isn't very useful, however, for *designing* flip-flop circuits. When designing circuits, the designer has to synthesize J and K to effect the desired

transition. You can show that when you are *given* the current state and desired next state, the required control inputs may be obtained from the following table:

Table 2—Transition table for design using JK flip-flops

Current State	Desired Next State	Inputs	
Q	Q ⁺	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

The × symbols in Table 2 indicate irrelevant “don’t care” input values.

Example: Suppose you have a circuit with an input A and you want to produce the state transitions shown in the following table, where the values of J and K are to be determined:

Table 3—State transition table for example

Current State	Circuit Input	Desired Next State	Flip-Flop Inputs	
Q	A	Q ⁺	J	K
0	0	0	?	?
0	1	1	?	?
1	0	1	?	?
1	1	0	?	?

Using Table 2, you can complete Table 3 as follows:

Current State	Circuit Input	Desired Next State	Flip-Flop Inputs	
Q	A	Q ⁺	J	K
0	0	0	0	×
0	1	1	1	×
1	0	1	×	0
1	1	0	×	1

The flip-flop inputs are determined from both the current state and the circuit inputs. By constructing a K-map (this is left as an exercise) for *both* J and K , using Q and A as inputs, you can determine that the optimal solution is

$$J = K = A$$

In other words, you would simply connect the circuit input A to both flip-flop inputs J and K . □

For this lab exercise you will construct a replacement for a mechanical push-on/push-off power switch. In other words, the first push of the switch should turn a LED on, the release of the switch should leave the LED on, the second push of the switch should turn the LED off, and the second release of the switch should leave the LED off. The circuit uses the memory of the flip-flop to remember whether the power is on or off.

The state transition table in Table 4 (which is incomplete) produces this behavior:

Table 4—State transition table for Lab Exercise 7

Current State			Circuit Input		Next State		Flip-Flop Inputs				Output (LED)
Q_1	Q_0	A	Q_1^+	Q_0^+	J_1	K_1	J_0	K_0	Y		
0	0	0	0	0					0		
0	0	1	0	1					0		
0	1	0	1	1					1		
0	1	1	0	1					1		
1	0	0	0	0					0		
1	0	1	1	0					0		
1	1	0	1	1					1		
1	1	1	1	0					1		

Pre-lab

- 1) Complete the J_1 , K_1 , J_0 , and K_0 columns of Table 4.
- 2) Construct K-maps for *each* of the variables J_1 , K_1 , J_0 , K_0 , and Y using Q_1 , Q_0 , and A as input variables. Determine *expressions* for each of the variables J_1 , K_1 , J_0 , K_0 , and Y .
- 3) Draw a complete circuit diagram, showing integrated circuit numbers and pin numbers. Your circuit will need combinational logic that produces J_1 , K_1 , J_0 , K_0 , and Y from Q_1 , Q_0 , and A . The pinout for the 7473 integrated circuit, which contains two JK flip-flops, is shown in Figure 2.

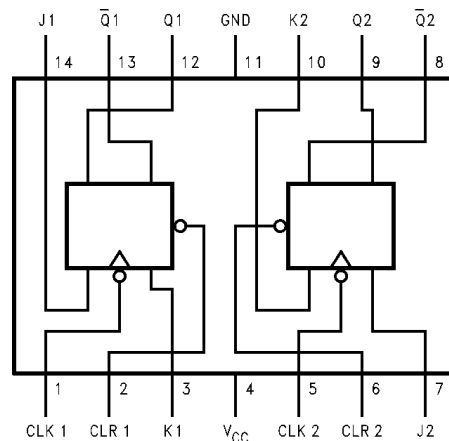


Figure 2—Pinout diagram for 7473 integrated circuit

Pay careful attention to the power connections, which are different from the other integrated circuits you have used this term. As always, pay careful attention to assertion levels.

Procedure

- 1) Build and test your circuit.
- 2) Use a push button as the input A. *Important note: Do not connect the pushbuttons directly to the +5 V supply! This will destroy the pushbuttons. You may not have working pushbuttons if some other student has done this!* If in doubt, ask your instructor how to connect the push buttons.
- 3) When you have your circuit working, demonstrate it to your instructor.

Troubleshooting Tips

- 1) Don't leave the CLR input of the flip-flop unconnected. Make sure it is deasserted.
- 2) JK flip-flops *must* be clocked via their CLK input. Use the built-in clock on the Cadet Masterlab as an input to the CLK input on the 7473. Run the clock at a very slow speed (or better yet, use a debounced push button) for debugging and run it fast when you have your circuit working.

Lab Exercise #8

Sequential logic circuits

Objectives

After completing this experiment, you will be able to design, build, and test a simple counter circuit, based on a verbal description of its operation.

Pre-lab

Design a 3-bit up/down counter with an input D that functions as follows: when D is asserted, the counter counts up (e.g., 000, 001, 010, 011, ...) on each clock pulse; when D is not asserted, the counter counts down on each clock pulse (e.g., 000, 111, 110, 101, ...) Your final circuit should have one input D , and three outputs Q_2 , Q_1 , and Q_0 . All inputs and outputs are assumed to be high asserting. Use JK flip-flops in your design. The three outputs are to be obtained directly from flip-flop outputs.

Perform your design by completing each of the following steps:

- 1) Draw a complete state transition diagram based on the previous description.

2) Complete the following present state/next state table:

State transition table for up/down counter

Current State				Circuit Input	Next State			Flip-Flop Inputs					
Q ₂	Q ₁	Q ₀	D		Q ₂ ⁺	Q ₁ ⁺	Q ₀ ⁺	J ₂	K ₂	J ₁	K ₁	J ₀	K ₀
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										

3) Obtain minimized expressions for J₂, K₂, J₁, K₁, J₀, and K₀.

- 4) Draw a complete circuit diagram showing part numbers and pin numbers.

Procedure

- 1) Build and test your circuit.
- 2) Use a push button as the input D. *Important note: Do not connect the pushbuttons directly to the +5 V supply! This will destroy the pushbuttons. You may not have working pushbuttons if some other student has done this!* If in doubt, ask your instructor how to connect the push buttons.
- 3) Connect your three-bit output to pins C, B, and A of the 7-segment display driver so you can observe the counting. You should be able to visually observe the digit counting upwards or downwards depending on the value of D
- 4) When you have your circuit working, demonstrate it to your instructor.

Troubleshooting Tips

- 1) Use the built-in clock on the Cadet Masterlab as an input to the CLK input on the 7473. Run the clock at a very slow speed (or better yet, use a debounced push button) for debugging and run it fast when you have your circuit working.

Lab Exercise #9

Design of register circuits

Objectives

After completing this experiment, you will be able to design, build, and test a simple register circuit, based on a verbal description of its operation.

Introduction

In this exercise, you will design a 3-bit left shift register that can be loaded in parallel. Each bit of the shift register will appear as follows:

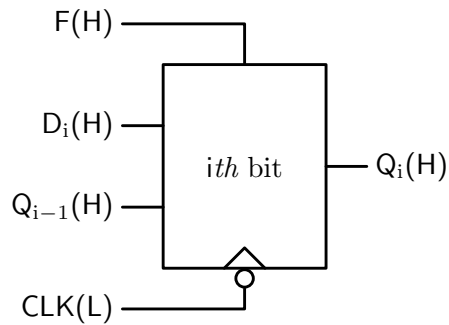


Figure 1—Single bit of a larger shift register

The D_i input is the bit to be loaded in parallel, the Q_{i-1} is the bit to be shifted from the adjacent bit, and the control input F determines whether the shift register will shift ($F = 1$) or load ($F = 0$). In other words, when $F = 1$, the value of Q_i on the next clock will be Q_{i-1} , and when $F = 0$, the value of Q_i on the next clock will be D_i .

Pre-lab

Design one bit of the register by completing each of the following steps:

- 1) Complete the following present state/next state table:

Table 1—State transition table for a single bit in the shift register

Present State	Circuit Input			Next State
Q_i	F	Q_{i-1}	D_i	Q_i^+
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	

- 2) Draw a complete circuit diagram that will implement this state transition table using a single 7474 D type flip-flop (the pinout for the 7474 is on the handout you were provided earlier in the term.)

Procedure

- 1) Build and test *three* identical single bit circuits. For testing, construct separate switch inputs for F, D_i , and for Q_{i-1} .
- 2) When you are sure you have them working make the following interconnections:
 - a) Connect the F inputs of all three bits together, and connect all three to a single input switch.
 - b) Connect each of the three D_i inputs to its own separate input switch.
 - c) Connect each Q_{i-1} input to the Q_i output of the adjacent bit to its right.
 - d) Connect the rightmost Q_{i-1} to a separate switch, because there will be no Q_{i-1} to its right.
 - e) Draw a block diagram showing how you have interconnected the three bits. Your diagram show show the interconnection of three bits, with each bit as depicted in Figure 1.
- 3) Test the combination. Make sure it can both load and shift properly.

When you are finished, you should have five switches connected to the circuit. One switch is connected to the Fs, three are connected to the D_i s, and the fourth is connected to the last Q_{i-1} input.

Troubleshooting Tips

Review the troubleshooting tips for flip-flops/sequential logic circuits from last week's lab.