# EXPERIMENT 3: INTRODUCTION TO COMBINATIONAL LOGIC AND BUS STRUCTURE

### PURPOSE

This experiment investigates several important aspects of combinational logic circuit design. Readings provide insight into finding minimal realizations of combinational logic, not only in terms of logic gates, but also chip count. In lab a combinational circuit will be investigated. Cascading smaller combinational circuits into larger circuits as well as the basics of a bus structure will be investigated.

### INTRODUCTION

Many logically equivalent realizations exist for any combinational function. It is the engineer's responsibility to make the best choice consistent with the imposed design restrictions. We now discuss some of the criteria used to evaluate a particular design.

### COST

One design criterion is cost. Sum of products and product of sums realizations attempt to minimize the cost of implementing combinational circuits. Their objective is to obtain a two-level realization, in a particular form, of any combinational logic function using a minimal number of gates. A secondary goal is to minimize the number of gate inputs.

The sum of products and product of sums realizations, however, do not always achieve the least cost realization for many reasons. Most importantly, such realizations do not necessarily use the minimal number of gates. For example, multi-level implementations may require fewer gates. Consider the following function:

F(A,B,C,D) = E + A'D + B'D + C'D.

You should verify that this expression is a minimal sum of products form. (a) shows the realization of this expression (assuming all input signals and their complements are available), which requires four gates. This expression can be factored as follows:

F(A,B,C,D) = E + D(A' + B' + C').

The realization of this *factored* expression is shown in (b) and requires only three gates. Thus, the standard sum of products realization, in this case, does not yield the least cost design.



Figure 4.1 Different Realizations of F(A,B,C,D,E).

However, factoring yields multi-level circuits rather than the two level circuits realized from the sum of products and product of sums forms. Propagation delay through multi-level circuits is greater than that for two-level circuits. Hence, a decrease in cost is gained at the expense of speed.

Unfortunately, even finding a design with the minimal number of gates does not guarantee the least cost design. Logic gates are available on IC packages, each one containing perhaps up to six of the gates. Each different gate function required by the design implies at least one IC containing that gate. In many cases, only a portion of the gates available on the IC's will actually be used in the design and, hence, many of the gates available on the IC's may remain unused.

In contrast, designs using all identical gates, such as all two-input NAND gates, will require an additional IC for every four NAND gates (assuming the IC's are quadruple two-input NAND gates). The resulting design may use the gates provided by the IC's more efficiently, implying less cost since fewer IC's are required. To illustrate, consider the following function:

F(A,B,C) = A'B + C'.

Assuming that only uncomplemented inputs are available, the straightforward realization, in (a), using NOT, AND, and OR gates requires 3 IC's: one for the two NOT gates, one for the AND gate, and one for the OR gate. If hex inverter and quadruple two-input AND and OR gate IC's are used, only about 30% of the gates are used--the rest are wasted.



(a) NOT-AND-OR realization of F.



1. All NAND realization of F.

Figure 4.2 Comparison of Different Realizations of F.

But the all NAND gate implementation, shown in (b), requires only one quadruple two-input NAND gate IC--no gates are wasted in this design. Thus, the all NAND gate implementation makes much more efficient use of the gates provided by the IC's, and subsequently costs less. (Even if the complements of all the input signals were available, the AND-OR realization would still require two IC's whereas the all NAND implementation requires only one.) This explains one of the reasons NAND gates are so popular in design. Their use is further supported by the one-to-one correspondence that exists between AND-OR and all NAND gate implementations. Hence, both designs require the same number of gates. However, the all NAND implementation generally uses the gates provided by the IC's more efficiently. Furthermore, every logic function can be realized solely with NAND gates.

Fan-in requirements of a design are yet another factor determining the cost of the design. Due to pin limitations, fewer gates can be packaged on IC's having gates with higher fan-in. For example, four two-input NAND gates can be accommodated by a 14 pin IC (two input and one output pin for each gate plus the two pins for power and ground), whereas only three three-input NAND gates (three input and one output pin for each gate plus the power and ground pins) or two four-input NAND gates (four input and one output pin for each gate plus power and ground pins--two pins are unused) can be accommodated by a 14 pin IC. Thus more IC's are required to provide the same number of gates if the gates have greater fan-in.

It should be noted that higher fan-in gates can be used to realize lower fan-in gates by tying inputs together or by connecting unused inputs to a logic 1 in the form of a pullup resistor connected to +5 volts. For example, a two-input AND gate can be realized from a four-input AND gate by tying three of its inputs together and using these three connected inputs along with the remaining separate input as the two inputs. Hence, different fan-in requirements for the same gate functions need not directly increase the number of IC's required. (Note that unused inputs should not be left unconnected; they should be either tied to other inputs or connected to +5 (logic 1) through a resistor.)

### FEASIBILITY

Not all design realizations are feasible. For example, consider the following function:

F(a,b,c,d,e,f,g,h) = abcdefgh.

This is the minimal product of sums and, also, the minimal sum of products form for F. However, you will be hard pressed to find an eight-input AND gate. Instead, this function must be implemented through a multi-level realization such as that in . (Alternatively, F can be realized with an 8-input NAND followed by an inverter.)



Figure 4.3 Multi-level Realization of an Eight-input AND Gate.

In addition, fanout limitations pose further design constraints. Consider an extrapolation of the 3-to-8 line decoder to an 8-to-256 line decoder. Each output is trivially realized using the product of sums and sum of products forms. But, in such a realization, each input line would have to be an input to 128 AND gates and a NOT gate, and the NOT gate's output would be an input to the remaining 128 AND gates. No logic family can provide such a high fanout. Such a realization is not feasible and an alternative design must be considered.

### BUS STRUCTURE

In many applications it is desirable to have a common communication “bus” between several devices. Each of these devices may send or receive data from this common bus. Tri-state buffers, such as the 74LS125, are useful in implementing such a structure. The logic symbol for a tri-state buffer is illustrated in (a) and their use to the tri-state buffer is illustrated in (b). The Y output of the tri-state buffer follows the A input as long as the C input is LOW. When the C input is HIGH, the Y output enters a high impedance state, acting like an open circuit to the bus. Through a predetermined protocol, all the devices’ outputs to the bus are in high impedance state, except at most one--the one which is transmitting data over the bus. This protocol is required to preclude bus contention. If more than one device attempts to send data on the bus, chaos would ensue. The messages of the devices transmitting simultaneously on the common bus would become hopelessly intertwined and rendered unintelligible; not to mention the possible damage to the transmitting devices should such an event occur.

Note that when tri-state buffers are connected to a bus, the control lines to the tri-state buffers select (at most) one of the buffers to be connected to the bus. Effectively one of the data inputs is being connected to the bus. This structure implements a multiplexor.



Figure 4.4 Tri-state Buffers Used to Implement a Bus Structure.

### PRELAB

1. For the circuit of Figure 4.5, draw a 3-variable Karnaugh for each combination of S1 and S0, i.e., for S1 S0 = 00, 01, 10 and 11. From the Karnaugh maps, derive the minimal SOP expressions for F and Co for each combination of S1 and S0.



Figure 4.5 A Logic Circuit for Arithmetic.

2. The circuit of Figure 4.5 is to be realized using 74LS exclusive-or gates, NAND gates, and inverters as necessary. (Assume the circuit inputs are only available in uncomplemented form.) Construct a chip layout diagram for the circuit. First, determine the placement of the chips on your breadboard. Then wire power and ground. You must consult the pin assignment diagrams for each chip in order to locate the appropriate pins. The power supply pin is usually labeled Vcc. Next wire the other pins according to your logic diagram. Double-check all connections. If any pins are unconnected, make sure they are supposed to be unconnected.

3. Draw schematic and chip layout diagrams for the tri-state bus circuit shown in Figure 4.6.

4. On your breadboard construct the tri-state bus circuit in the same manner as part 2.



Figure 4.6 Circuit Used to Simulate a Bus Structure.

### PROCEDURE

Before performing the procedures listed below, read the report section of the experiment to assure you make all required measurements and record all required data.

As each step is done, include the results in the final report

*A. Function*

Using the counter in the logicbox obtain the table of combinations for the outputs F and Co. (This can be done by connecting S1, S0, A, B, Ci to the 5 least significant bits of the counter, connecting F and Co to LEDs, and then single stepping the counter through the 32 combinations.) Verify the function performed by the circuit, i.e., verify your Karnaugh map from the Prelab. Using the pulse generator and the scope, measure the delay through the longest propagation path in the circuit.

*B. Cascading*

On your breadboard construct two of the circuits in Figure 4.5 connected in ‘cascade’. (By cascade, we mean that the S1 and S0 inputs are connected together for the two circuits, the A and B inputs of the two circuits are independent inputs, and Co from the first circuit is connected to Ci of the second circuit. (Thus, the cascade circuit has seven inputs (S1, S0, Ci, two A’s, two B’s) and four outputs (two F’s, two Co’s).) Find an input pattern such that changing the input Ci changes the output Co. Then find the propagation delay between Ci and Co in this cascaded circuit.

*C. Bus structure*

Hardwire the circuit of problem 4 in the prelab and using the pulse generator and the oscilloscope obtain a timing diagram for the situation where the enable is changing and the inputs A and B have opposite values.

# EXPERIMENT 4--INTRODUCTION TO COMBINATIONAL LOGIC AND BUS STRUCTURE

# FINAL REPORT

**I. *Function***

a). Draw the logic diagram for your circuit of part A.1. Indicate on the diagram the longest propagation path through the circuit. Also, give the logic levels on all the lines in the circuit, other than on this path, for an input combination that sensitizes this path. (The path is sensitized if a change on the path input propagates all the way to the output.)

b). Include the Karnaugh Map function expressions from Part 1 of the Prelab.

c). Either sketch or include the captured waveforms observed on the scope in determining the longest propagation delay of the circuit in Part A.



**LOW to HIGH Output Transition**



**HIGH to LOW Output Transition**

d). From your scope waveforms determine the average propagation delay through the longest path of the circuit in part A.

**II. *Cascading***

a). Include your waveform diagram showing the longest propagation time between Ci and Co in the cascade circuit. For this situation, if S1 and S0 are constant, describe in words the function performed by the cascade circuit?

b). If 16 copies of the circuit were cascaded in the same manner, what would be the maximum propagation delay through the circuit?

**III. *Bus Structure***

a). Designate the value on the tri-state bus as OUT. Fill in the table combinations below and derive the switching algebra expression for OUT.

Enable A B OUT

0 0 0

0 0 1

0 1 0

0 1 1

1 0 0

1 0 1

1 1 0

1 1 1

d) Sketch output waveforms observed on the scope from part C.2.



**LOW to HIGH Output Transition**



**HIGH to LOW Output Transition**

e) How could the circuit of Figure 4.6 be modified so that the bus could be placed in a high-impedance state when desired? Draw a circuit diagram for the new design and explain its operation.