Chapter 11

Laboratory Experiments with Standard ICs and FPGAs

11.1 INTRODUCTION TO EXPERIMENTS

This chapter presents 18 laboratory experiments in digital circuits and logic design. The experiments give the student using this book hands-on experience. The digital circuits can be constructed by using standard integrated circuits (ICs) mounted on breadboards that are easily assembled in the laboratory. The experiments are ordered according to the material presented in the book. The last section consists of a number of supplements with suggestions for using the Verilog HDL to simulate and verify the functionality of the digital circuits presented in the experiments. If an FPGA prototyping board is available, the experiments can be implemented in an FPGA as an alternative to standard ICs.

A logic breadboard suitable for performing the experiments must have the following equipment:

1. Light-emitting diode (LED) indicator lamps.
2. Toggle switches to provide logic-1 and logic-0 signals.
3. Pulsers with push buttons and debounce circuits to generate single pulses.
4. A clock-pulse generator with at least two frequencies: a low frequency of about 1 pulse per second to observe slow changes in digital signals and a higher frequency for observing waveforms in an oscilloscope.
5. A power supply of 5 V.
6. Socket strips for mounting the ICs.
7. Solid hookup wires and a pair of wire strippers for cutting the wires.

Digital logic trainers that include the required equipment are available from several manufacturers. A digital logic trainer contains LED lamps, toggle switches, pulsers, a variable clock,

a power supply, and IC socket strips. Some experiments may require additional switches, lamps, or IC socket strips. Extended breadboards with more solderless sockets and plug-in switches and lamps may be needed.

Additional equipment required is a dual-trace oscilloscope (for Experiments 1, 2, 8, and 15), a logic probe to be used for debugging, and a number of ICs. The ICs required for the experiments are of the TTL or CMOS series 7400.

The integrated circuits to be used in the experiments can be classified as small-scale integration (SSI) or medium-scale integration (MSI) circuits. SSI circuits contain individual gates or flip-flops, and MSI circuits perform specific digital functions. The eight SSI gate ICs needed for the experiments—two-input NAND, NOR, AND, OR, and XOR gates, inverters, and three-input and four-input NAND gates—are shown in Fig. 11.1. The pin assignments for the gates are indicated in the diagram. The pins are numbered from 1 to 14. Pin number 14 is marked V_{CC}, and pin number 7 is marked *GND* (ground). These are the supply terminals, which must be connected to a power supply of 5 V for proper operation of the circuit. Each IC is recognized by its identification number; for example, the two-input NAND gates are found inside the IC whose number is 7400.

Detailed descriptions of the MSI circuits can be found in data books published by the manufacturers. The best way to acquire experience with a commercial MSI circuit is to study its description in a data book that provides complete information on the internal, external, and electrical characteristics of integrated circuits. Various semiconductor companies publish data books for the 7400 series. The MSI circuits that are needed for the experiments are introduced and explained when they are used for the first time. The operation of the circuit is explained by referring to similar circuits in previous chapters. The information given in this chapter about the MSI circuits should be sufficient for performing the experiments adequately. Nevertheless, reference to a data book will always be preferable, as it gives more detailed description of the circuits.

We will now demonstrate the method of presentation of MSI circuits adopted here. To illustrate, we introduce the ripple counter IC, type 7493. This IC is used in Experiment 1 and in subsequent experiments to generate a sequence of binary numbers for verifying the operation of combinational circuits.

The information about the 7493 IC that is found in a data book is shown in Figs. 11.2(a) and (b). Part (a) shows a diagram of the internal logic circuit and its connection to external pins. All inputs and outputs are given symbolic letters and assigned to pin numbers. Part (b) shows the physical layout of the IC, together with its 14-pin assignment to signal names. Some of the pins are not used by the circuit and are marked as *NC* (no connection). The IC is inserted into a socket, and wires are connected to the various pins through the socket terminals. When drawing schematic diagrams in this chapter, we will show the IC in block diagram form, as in Fig. 11.2(c). The IC number (here, 7493) is written inside the block. All input terminals are placed on the left of the block and all output terminals on the right. The letter symbols of the signals, such as *A*, *R1*, and *QA*, are written inside the block, and the corresponding pin numbers, such as 14, 2, and 12, are written along the external lines. V_{CC} and *GND* are the power terminals connected to pins 5 and 10. The size of the block may vary to accommodate all input and output terminals. Inputs or outputs may sometimes be placed on the top or the bottom of the block for convenience.

The operation of the circuit is similar to the ripple counter shown in Fig. 6.8(a) with an asynchronous clear to each flip-flop. When input *R1* or *R2* or both are equal to logic 0 (ground), all asynchronous clears are equal to 1 and are disabled. To clear all four flip-flops to 0, the output

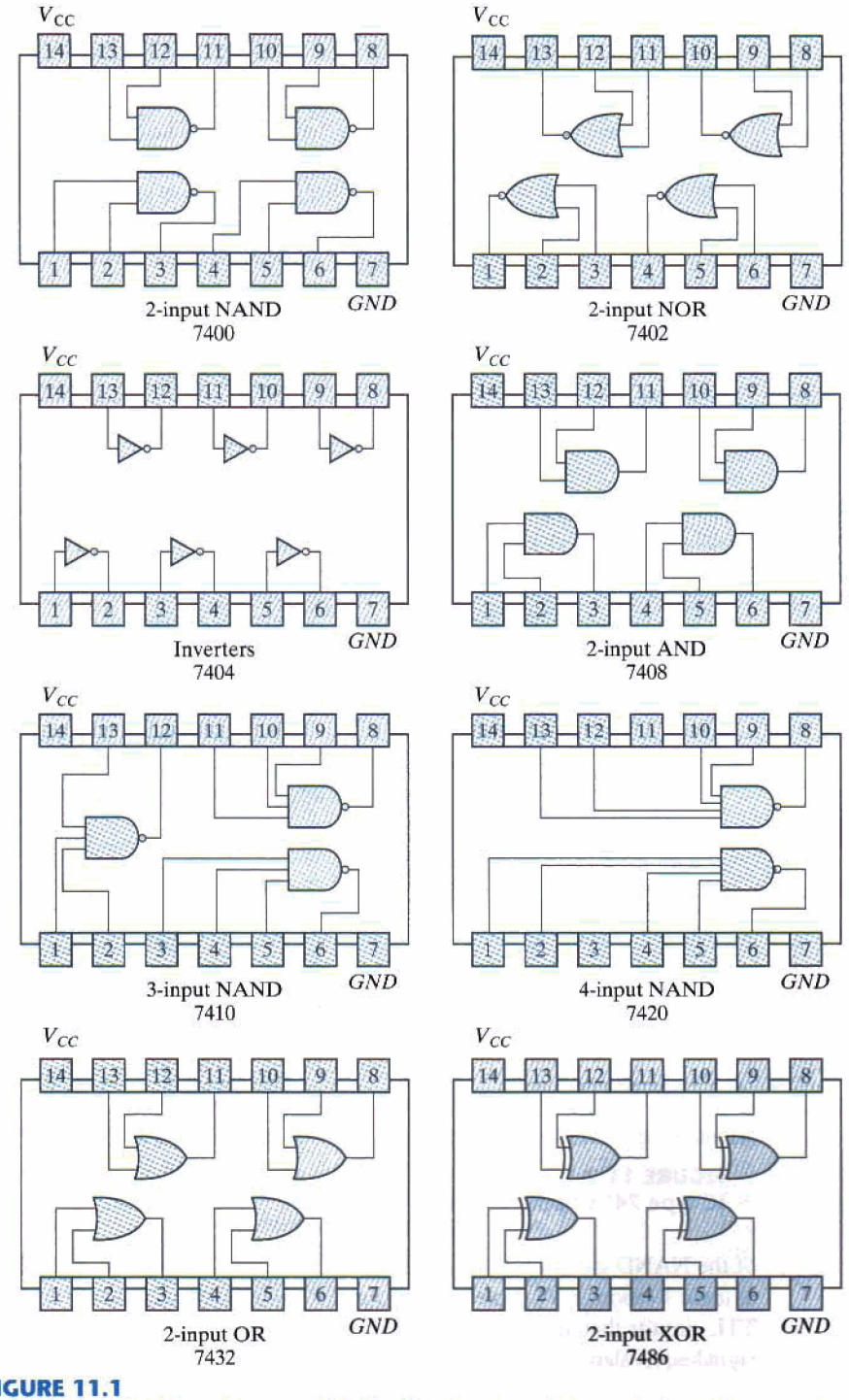
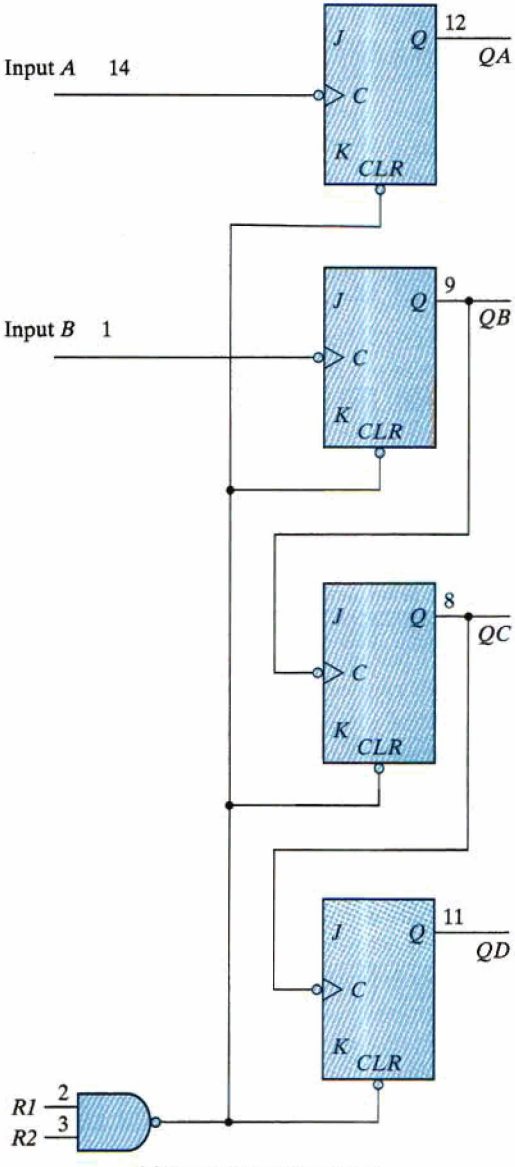
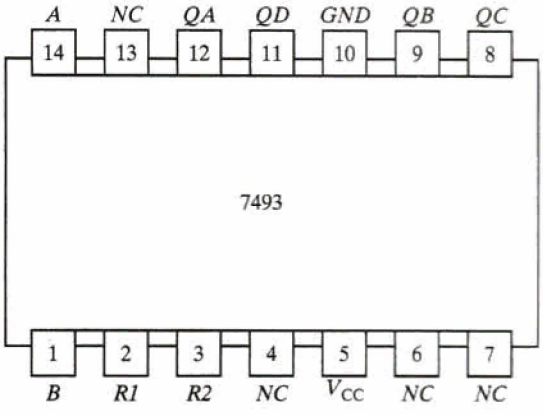


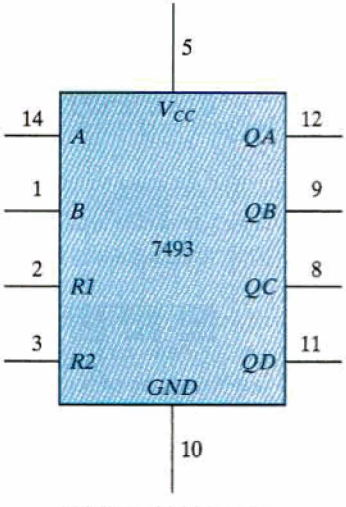
FIGURE 11.1
Digital gates in IC packages with identification numbers and pin assignments



(a) Internal circuit diagram



(b) Physical layout (*NC*: no connection)



(c) Schematic diagram

FIGURE 11.2
IC type 7493 ripple counter

of the NAND gate must be equal to 0. This is accomplished by having both inputs *R1* and *R2* at logic 1 (about 5 V). Note that the *J* and *K* inputs show no connections. It is characteristic of TTL circuits that an input terminal with no external connections has the effect of producing a signal equivalent to logic 1. Note also that output *QA* is not connected to input *B* internally.

The 7493 IC can operate as a three-bit counter using input *B* and flip-flops *QB*, *QC*, and *QD*. It can operate as a four-bit counter using input *A* if output *QA* is connected to input *B*. Therefore, to operate the circuit as a four-bit counter, it is necessary to have an external connection between pin 12 and pin 1. The reset inputs, *R1* and *R2*, at pins 2 and 3, respectively, must be grounded. Pins 5 and 10 must be connected to a 5-V power supply. The input pulses must be applied to input *A* at pin 14, and the four flip-flop outputs of the counter are taken from *QA*, *QB*, *QC*, and *QD* at pins 12, 9, 8, and 11, respectively, with *QA* being the least significant bit.

Figure 11.2(c) demonstrates the way that all MSI circuits will be symbolized graphically in this chapter. Only a block diagram similar to the one shown in this figure will be given for each IC. The letter symbols for the inputs and outputs in the IC block diagram will be according to the symbols used in the data book. The operation of the circuit will be explained with reference to logic diagrams from previous chapters. The operation of the circuit will be specified by means of a truth table or a function table.

Other possible graphic symbols for the ICs are presented in Chapter 12. These are standard graphic symbols approved by the Institute of Electrical and Electronics Engineers and are given in IEEE Standard 91-1984. The standard graphic symbols for SSI gates have rectangular shapes, as shown in Fig. 12.1. The standard graphic symbol for the 7493 IC is shown in Fig. 12.13. This symbol can be substituted in place of the one shown in Fig. 11.2(c). The standard graphic symbols of the other ICs that are needed to run the experiments are presented in Chapter 12. They can be used to draw schematic diagrams of the logic circuits if the standard symbols are preferred.

Table 11.1 lists the ICs that are needed for the experiments, together with the numbers of the figures in which they are presented in this chapter. In addition, the table lists the numbers of the figures in Chapter 12 in which the equivalent standard graphic symbols are drawn.

Table 11.1
Integrated Circuits Required for the Experiments

		Graphic Symbol	
IC Number	**Description**	**In Chapter 11**	**In Chapter 12**
	Various gates	Fig. 11.1	Fig. 12.1
7447	BCD-to-seven-segment decoder	Fig. 11.8	—
7474	Dual *D*-type flip-flops	Fig. 11.13	Fig. 12.9(b)
7476	Dual *JK*-type flip-flops	Fig. 11.12	Fig. 12.9(a)
7483	Four-bit binary adder	Fig. 11.10	Fig. 12.2
7493	Four-bit ripple counter	Fig. 11.2	Fig. 12.13
74151	8 × 1 multiplexer	Fig. 11.9	Fig. 12.7(a)
74155	3 × 8 decoder	Fig. 11.7	Fig. 12.6
74157	Quadruple 2 × 1 multiplexers	Fig. 11.17	Fig. 12.7(b)
74161	Four-bit synchronous counter	Fig. 11.15	Fig. 12.14
74189	16 × 4 random-access memory	Fig. 11.18	Fig. 12.15
74194	Bidirectional shift register	Fig. 11.19	Fig. 12.12
74195	Four-bit shift register	Fig. 11.16	Fig. 12.11
7730	Seven-segment LED display	Fig. 11.8	—
72555	Timer (same as 555)	Fig. 11.21	—

The next 18 sections present 18 hardware experiments requiring the use of digital integrated circuits. Section 11.20 outlines HDL simulation experiments requiring a Verilog HDL compiler and simulator.

11.2 EXPERIMENT 1: BINARY AND DECIMAL NUMBERS

This experiment demonstrates the count sequence of binary numbers and the binary-coded decimal (BCD) representation. It serves as an introduction to the breadboard used in the laboratory and acquaints the student with the cathode-ray oscilloscope. Reference material from the text that may be useful to know while performing the experiment can be found in Section 1.2, on binary numbers, and Section 1.7, on BCD numbers.

Binary Count

IC type 7493 consists of four flip-flops, as shown in Fig. 11.2. They can be connected to count in binary or in BCD. Connect the IC to operate as a four-bit binary counter by wiring the external terminals, as shown in Fig. 11.3. This is done by connecting a wire from pin 12 (output *QA*) to pin 1 (input *B*). Input *A* at pin 14 is connected to a pulser that provides single pulses. The two reset inputs, *R1* and *R2*, are connected to ground. The four outputs go to four indicator lamps, with the low-order bit of the counter from *QA* connected to the rightmost indicator lamp. Do not forget to supply 5 V and ground to the IC. All connections should be made with the power supply in the off position.

Turn the power on and observe the four indicator lamps. The four-bit number in the output is incremented by 1 for every pulse generated in the push-button pulser. The count goes to binary

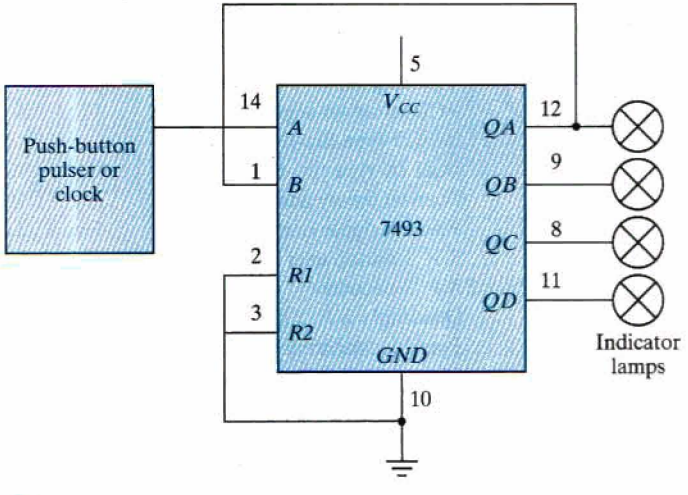


FIGURE 11.3
Binary counter

15 and then back to 0. Disconnect the input of the counter at pin 14 from the pulser, and connect it to a clock generator that produces a train of pulses at a low frequency of about 1 pulse per second. This will provide an automatic binary count. Note that the binary counter will be used in subsequent experiments to provide the input binary signals for testing combinational circuits.

Oscilloscope Display

Increase the frequency of the clock to 10 kHz or higher and connect its output to an oscilloscope. Observe the clock output on the oscilloscope and sketch its waveform. Using a dual-trace oscilloscope, connect the output of *QA* to one channel and the output of the clock to the second channel. Note that the output of *QA* is complemented every time the clock pulse goes through a negative transition from 1 to 0. Note also that the clock frequency at the output of the first flip-flop is one-half that of the input clock frequency. Each flip-flop in turn divides its incoming frequency by 2. The four-bit counter divides the incoming frequency by 16 at output *QD*. Obtain a timing diagram showing the relationship of the clock to the four outputs of the counter. Make sure that you include at least 16 clock cycles. The way to proceed with a dual-trace oscilloscope is as follows: First, observe the clock pulses and *QA*, and record their timing waveforms. Then repeat by observing and recording the waveforms of *QA* together with *QB*, followed by the waveforms of *QB* with *QC* and then *QC* with *QD*. Your final result should be a diagram showing the relationship of the clock to the four outputs in one composite diagram having at least 16 clock cycles.

BCD Count

The BCD representation uses the binary numbers from 0000 to 1001 to represent the coded decimal digits from 0 to 9. IC type 7493 can be operated as a BCD counter by making the external connections shown in Fig. 11.4. Outputs *QB* and *QD* are connected to the two reset inputs,

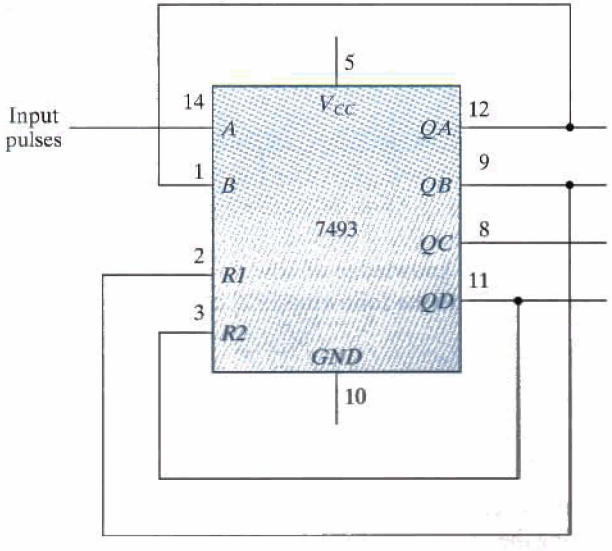


FIGURE 11.4
BCD counter

R1 and *R2*. When both *R1* and *R2* are equal to 1, all four cells in the counter clear to 0 irrespective of the input pulse. The counter starts from 0, and every input pulse increments it by 1 until it reaches the count of 1001. The next pulse changes the ouput to 1010, making *QB* and *QD* equal to 1. This momentary output cannot be sustained, because the four cells immediately clear to 0, with the result that the output goes to 0000. Thus, the pulse after the count of 1001 changes the output to 0000, producing a BCD count.

Connect the IC to operate as a BCD counter. Connect the input to a pulser and the four outputs to indicator lamps. Verify that the count goes from 0000 to 1001.

Disconnect the input from the pulser and connect it to a clock generator. Observe the clock waveform and the four outputs on the oscilloscope. Obtain an accurate timing diagram showing the relationship between the clock and the four outputs. Make sure to include at least 10 clock cycles in the oscilloscope display and in the composite timing diagram.

Output Pattern

When the count pulses into the BCD counter are continuous, the counter keeps repeating the sequence from 0000 to 1001 and back to 0000. This means that each bit in the four outputs produces a fixed pattern of 1's and 0's that is repeated every 10 pulses. These patterns can be predicted from a list of the binary numbers from 0000 to 1001. The list will show that output *QA*, being the least significant bit, produces a pattern of alternate 1's and 0's. Output *QD*, being the most significant bit, produces a pattern of eight 0's followed by two 1's. Obtain the pattern for the other two outputs and then check all four patterns on the oscilloscope. This is done with a dual-trace oscilloscope by displaying the clock pulses in one channel and one of the output waveforms in the other channel. The pattern of 1's and 0's for the corresponding output is obtained by observing the output levels at the vertical positions where the pulses change from 1 to 0.

Other Counts

IC type 7493 can be connected to count from 0 to a variety of final counts. This is done by connecting one or two outputs to the reset inputs, *R1* and *R2*. Thus, if *R1* is connected to *QA* instead of to *QB* in Fig. 11.4, the resulting count will be from 0000 to 1000, which is 1 less than 1001 ($QD = 1$ and $QA = 1$).

Utilizing your knowledge of how *R1* and *R2* affect the final count, connect the 7493 IC to count from 0000 to the following final counts:

(a) 0101

(b) 0111

(c) 1011

Connect each circuit and verify its count sequence by applying pulses from the pulser and observing the output count in the indicator lamps. If the initial count starts with a value greater than the final count, keep applying input pulses until the output clears to 0.

11.3 EXPERIMENT 2: DIGITAL LOGIC GATES

In this experiment, you will investigate the logic behavior of various IC gates:

7400 quadruple two-input NAND gates

7402 quadruple two-input NOR gates

7404 hex inverters

7408 quadruple two-input AND gates

7432 quadruple two-input OR gates

7486 quadruple two-input XOR gates

The pin assignments to the various gates are shown in Fig. 11.1. "Quadruple" means that there are four gates within the package. The digital logic gates and their characteristics are discussed in Section 2.8. A NAND implementation is discussed in Section 3.7.

Truth Tables

Use one gate from each IC listed and obtain the truth table of the gate. The truth table is obtained by connecting the inputs of the gate to switches and the output to an indicator lamp. Compare your results with the truth tables listed in Fig. 2.5.

Waveforms

For each gate listed, obtain the input–output waveform of the gate. The waveforms are to be observed in the oscilloscope. Use the two low-order outputs of a binary counter (Fig. 11.3) to provide the inputs to the gate. As an example, the circuit and waveforms for the NAND gate are illustrated in Fig. 11.5. The oscilloscope display will repeat this waveform, but you should record only the nonrepetitive portion.

Propagation Delay

Connect the six inverters inside the 7404 IC in cascade. The output will be the same as the input, except that it will be delayed by the time it takes the signal to propagate through all six inverters. Apply clock pulses to the input of the first inverter. Using the oscilloscope, determine

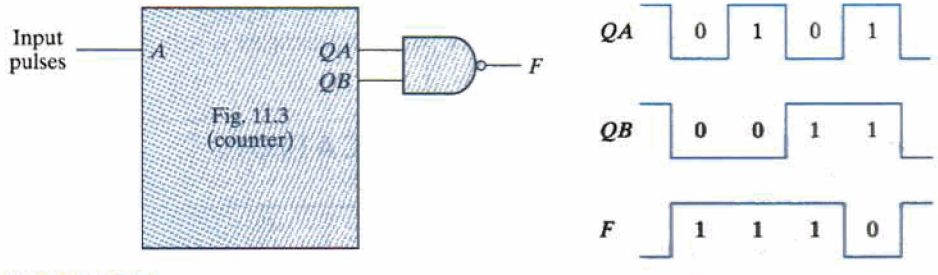


FIGURE 11.5
Waveforms for NAND gate

the delay from the input to the output of the sixth inverter during the upswing of the pulse and again during the downswing. This is done with a dual-trace oscilloscope by applying the input clock pulses to one of the channels and the output of the sixth inverter to the second channel. Set the time-base knob to the lowest time-per-division setting. The rise or fall time of the two pulses should appear on the screen. Divide the total delay by 6 to obtain an average propagation delay per inverter.

Universal NAND Gate

Using a single 7400 IC, connect a circuit that produces

(a) an inverter.

(b) a two-input AND.

(c) a two-input OR.

(d) a two-input NOR.

(e) a two-input XOR. (See Fig. 3.32.)

In each case, verify your circuit by checking its truth table.

NAND Circuit

Using a single 7400 IC, construct a circuit with NAND gates that implements the Boolean function

$$F = AB + CD$$

1. Draw the circuit diagram.
2. Obtain the truth table for F as a function of the four inputs.
3. Connect the circuit and verify the truth table.
4. Record the patterns of 1's and 0's for F as inputs A, B, C, and D go from binary 0 to binary 15.
5. Connect the four outputs of the binary counter shown in Fig. 11.3 to the four inputs of the NAND circuit. Connect the input clock pulses from the counter to one channel of a dual-trace oscilloscope and output F to the other channel. Observe and record the 1's and 0's pattern of F after each clock pulse, and compare it with the pattern recorded in step 4.

11.4 EXPERIMENT 3: SIMPLIFICATION OF BOOLEAN FUNCTIONS

This experiment demonstrates the relationship between a Boolean function and the corresponding logic diagram. The Boolean functions are simplified by using the map method, as discussed in Chapter 3. The logic diagrams are to be drawn with NAND gates, as explained in Section 3.7.

The gate ICs to be used for the logic diagrams must be those from Fig. 11.1 which contain the following NAND gates:

7400 two-input NAND

7404 inverter (one-input NAND)

7410 three-input NAND

7420 four-input NAND

If an input to a NAND gate is not used, it should not be left open, but instead should be connected to another input that is used. For example, if the circuit needs an inverter and there is an extra two-input gate available in a 7400 IC, then both inputs of the gate are to be connected together to form a single input for an inverter.

Logic Diagram

This part of the experiment starts with a given logic diagram from which we proceed to apply simplification procedures to reduce the number of gates and, possibly, the number of ICs. The logic diagram shown in Fig. 11.6 requires two ICs—a 7400 and a 7410. Note that the inverters for inputs x, y, and z are obtained from the remaining three gates in the 7400 IC. If the inverters were taken from a 7404 IC, the circuit would have required three ICs. Note

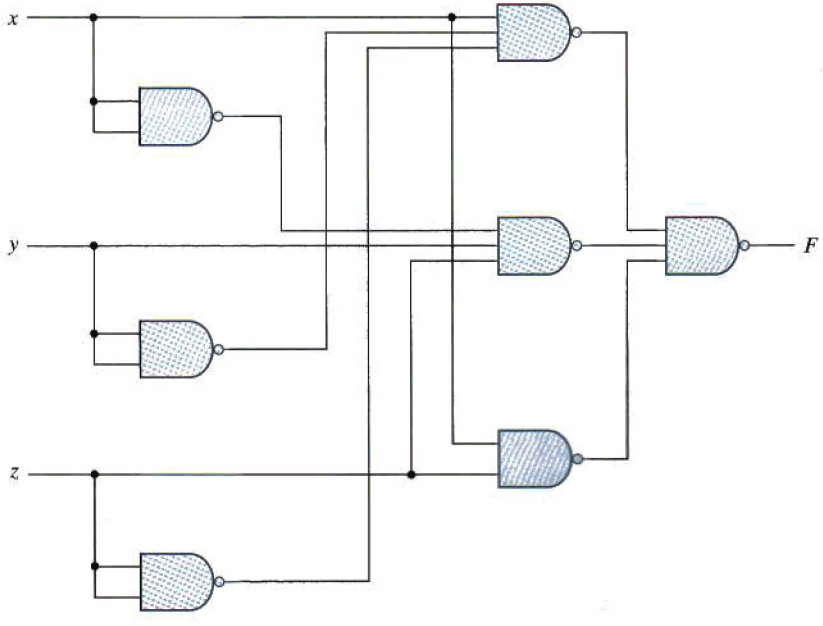


FIGURE 11.6
Logic diagram for Experiment 3

also that, in drawing SSI circuits, the gates are not enclosed in blocks as is done with MSI circuits.

Assign pin numbers to all inputs and outputs of the gates, and connect the circuit with the x, y, and z inputs going to three switches and the output F to an indicator lamp. Test the circuit by obtaining its truth table.

Obtain the Boolean function of the circuit and simplify it, using the map method. Construct the simplified circuit without disconnecting the original circuit. Test both circuits by applying identical inputs to each and observing the separate outputs. Show that, for each of the eight possible input combinations, the two circuits have identical outputs. This will prove that the simplified circuit behaves exactly like the original circuit.

Boolean Functions

Consider two Boolean functions in sum-of-minterms form:

$$F_1(A, B, C, D) = (0, 1, 4, 5, 8, 9, 10, 12, 13)$$
$$F_2(A, B, C, D) = (3, 5, 7, 8, 10, 11, 13, 15)$$

Simplify these functions by means of maps. Obtain a composite logic diagram with four inputs, A, B, C, and D, and two outputs, F_1 and F_2. Implement the two functions together, using a minimum number of NAND ICs. Do not duplicate the same gate if the corresponding term is needed for both functions. Use any extra gates in existing ICs for inverters when possible. Connect the circuit and check its operation. The truth table for F_1 and F_2 obtained from the circuit should conform with the minterms listed.

Complement

Plot the following Boolean function in a map:

$$F = A'D + BD + B'C + AB'D$$

Combine the 1's in the map to obtain the simplified function for F in sum-of-products form. Then combine the 0's in the map to obtain the simplified function for F', also in sum-of-products form. Implement both F and F' with NAND gates, and connect the two circuits to the same input switches, but to separate output indicator lamps. Obtain the truth table of each circuit in the laboratory and show that they are the complements of each other.

11.5 EXPERIMENT 4: COMBINATIONAL CIRCUITS

In this experiment, you will design, construct, and test four combinational logic circuits. The first two circuits are to be constructed with NAND gates, the third with XOR gates, and the fourth with a decoder and NAND gates. Reference to a parity generator can be found in Section 3.9. Implementation with a decoder is discussed in Section 4.9.

Design Example

Design a combinational circuit with four inputs—A, B, C, and D—and one output, F. F is to be equal to 1 when $A = 1$, provided that $B = 0$, or when $B = 1$, provided that either C or D is also equal to 1. Otherwise, the output is to be equal to 0.

1. Obtain the truth table of the circuit.
2. Simplify the output function.
3. Draw the logic diagram of the circuit, using NAND gates with a minimum number of ICs.
4. Construct the circuit and test it for proper operation by verifying the given conditions.

Majority Logic

A majority logic is a digital circuit whose output is equal to 1 if the majority of the inputs are 1's. The output is 0 otherwise. Design and test a three-input majority circuit using NAND gates with a minimum number of ICs.

Parity Generator

Design, construct, and test a circuit that generates an even parity bit from four message bits. Use XOR gates. Adding one more XOR gate, expand the circuit so that it generates an odd parity bit also.

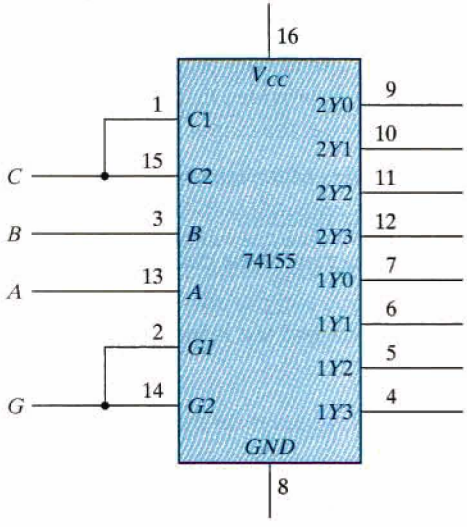
Decoder Implementation

A combinational circuit has three inputs—x, y, and z—and three outputs—F_1, F_2, and F_3. The simplified Boolean functions for the circuit are

$$F_1 = xz + x'y'z'$$
$$F_2 = x'y + xy'z'$$
$$F_3 = xy + x'y'z$$

Implement and test the combinational circuit, using a 74155 decoder IC and external NAND gates.

The block diagram of the decoder and its truth table are shown in Fig. 11.7. The 74155 can be connected as a dual 2 × 4 decoder or as a single 3 × 8 decoder. When a 3 × 8 decoder is desired, inputs $C1$ and $C2$, as well as inputs $G1$ and $G2$, must be connected together, as shown in the block diagram. The function of the circuit is similar to that illustrated in Fig. 4.18. G is the enable input and must be equal to 0 for proper operation. The eight outputs are labeled with symbols given in the data book. The 74155 uses NAND gates, with the result that the selected output goes to 0 while all other outputs remain at 1. The implementation with the decoder is as shown in Fig. 4.21, except that the OR gates must be replaced with external NAND gates when the 74155 is used.



Truth table

Inputs				Outputs							
G	*C*	*B*	*A*	2*Y*0	2*Y*1	2*Y*2	2*Y*3	1*Y*0	1*Y*1	1*Y*2	1*Y*3
1	X	X	X	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	0

FIGURE 11.7
IC type 74155 connected as a 3 × 8 decoder

11.6 EXPERIMENT 5: CODE CONVERTERS

The conversion from one binary code to another is common in digital systems. In this experiment, you will design and construct three combinational-circuit converters. Code conversion is discussed in Section 4.4.

Gray Code to Binary

Design a combinational circuit with four inputs and four outputs that converts a four-bit Gray code number (Table 1.6) into the equivalent four-bit binary number. Implement the circuit with exclusive-OR gates. (This can be done with one 7486 IC.) Connect the circuit to four switches and four indicator lamps, and check for proper operation.

9's Complementer

Design a combinational circuit with four input lines that represent a decimal digit in BCD and four output lines that generate the 9's complement of the input digit. Provide a fifth output that detects an error in the input BCD number. This output should be equal to logic 1 when the four inputs have one of the unused combinations of the BCD code. Use any of the gates listed in Fig. 11.1, but minimize the total number of ICs used.

Seven-Segment Display

A seven-segment indicator is used to display any one of the decimal digits 0 through 9. Usually, the decimal digit is available in BCD. A BCD-to-seven-segment decoder accepts a decimal digit in BCD and generates the corresponding seven-segment code, as is shown pictorially in Problem 4.9.

Figure 11.8 shows the connections necessary between the decoder and the display. The 7447 IC is a BCD-to-seven-segment decoder/driver that has four inputs for the BCD digit. Input D is the most significant and input A the least significant. The four-bit BCD digit is converted to a seven-segment code with outputs a through g. The outputs of the 7447 are applied to the inputs of the 7730 (or equivalent) seven-segment display. This IC contains the seven light-emitting diode (LED) segments on top of the package. The input at pin 14 is the common anode (CA) for all the LEDs. A 47-Ω resistor to V_{CC} is needed in order to supply the proper current to the selected LED segments. Other equivalent seven-segment display ICs may have additional anode terminals and may require different resistor values.

Construct the circuit shown in Fig. 11.8. Apply the four-bit BCD digits through four switches, and observe the decimal display from 0 to 9. Inputs 1010 through 1111 have no meaning in BCD.

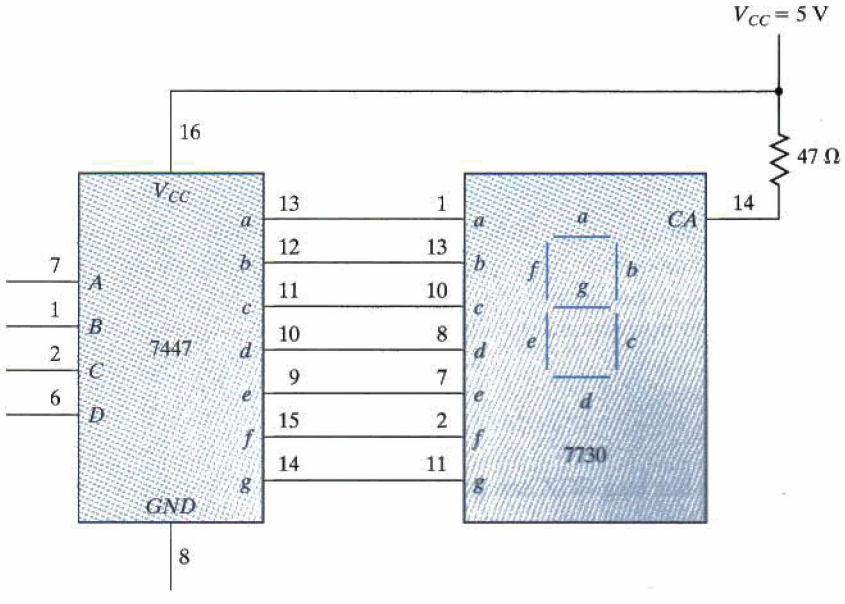
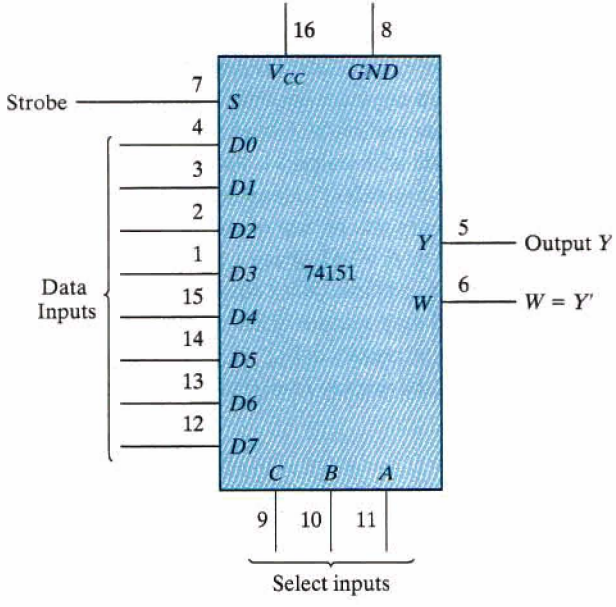


FIGURE 11.8
BCD-to-seven-segment decoder (7447) and seven-segment display (7730)

Depending on the decoder, these values may cause either a blank or a meaningless pattern to be displayed. Observe and record the output patterns of the six unused input combinations.

11.7 EXPERIMENT 6: DESIGN WITH MULTIPLEXERS

In this experiment, you will design a combinational circuit and implement it with multiplexers, as explained in Section 4.11. The multiplexer to be used is IC type 74151, shown in Fig. 11.9. The internal construction of the 74151 is similar to the diagram shown in Fig. 4.25, except that



Function table

Strobe	Select			Output
S	*C*	*B*	*A*	*Y*
1	X	X	X	0
0	0	0	0	*D0*
0	0	0	1	*D1*
0	0	1	0	*D2*
0	0	1	1	*D3*
0	1	0	0	*D4*
0	1	0	1	*D5*
0	1	1	0	*D6*
0	1	1	1	*D7*

FIGURE 11.9
IC type 74151 38 × 1 multiplexer

there are eight inputs instead of four. The eight inputs are designated *D0* through *D7*. The three selection lines—*C*, *B*, and *A*—select the particular input to be multiplexed and applied to the output. A strobe control *S* acts as an enable signal. The function table specifies the value of output *Y* as a function of the selection lines. Output *W* is the complement of *Y*. For proper operation, the strobe input *S* must be connected to ground.

Design Specifications

A small corporation has 10 shares of stock, and each share entitles its owner to one vote at a stockholder's meeting. The 10 shares of stock are owned by four people as follows:

Mr. W: 1 share

Mr. X: 2 shares

Mr. Y: 3 shares

Mrs. Z: 4 shares

Each of these persons has a switch to close when voting yes and to open when voting no for his or her shares.

It is necessary to design a circuit that displays the total number of shares that vote yes for each measure. Use a seven-segment display and a decoder, as shown in Fig. 11.8, to display the required number. If all shares vote no for a measure, the display should be blank. (Note that binary input 15 into the 7447 blanks out all seven segments.) If 10 shares vote yes for a measure, the display should show 0. Otherwise, the display shows a decimal number equal to the number of shares that vote yes. Use four 74151 multiplexers to design the combinational circuit that converts the inputs from the stock owners' switches into the BCD digit for the 7447. Do not use 5 V for logic 1. Use the output of an inverter whose input is grounded.

11.8 EXPERIMENT 7: ADDERS AND SUBTRACTORS

In this experiment, you will construct and test various adder and subtractor circuits. The subtractor circuit is then used to compare the relative magnitudes of two numbers. Adders are discussed in Section 4.3. Subtraction with 2's complement is explained in Section 1.6. A four-bit parallel adder–subtractor is shown in Fig. 4.13, and the comparison of two numbers is explained in Section 4.8.

Half Adder

Design, construct, and test a half-adder circuit using one XOR gate and two NAND gates.

Full Adder

Design, construct, and test a full-adder circuit using two ICs, 7486 and 7400.

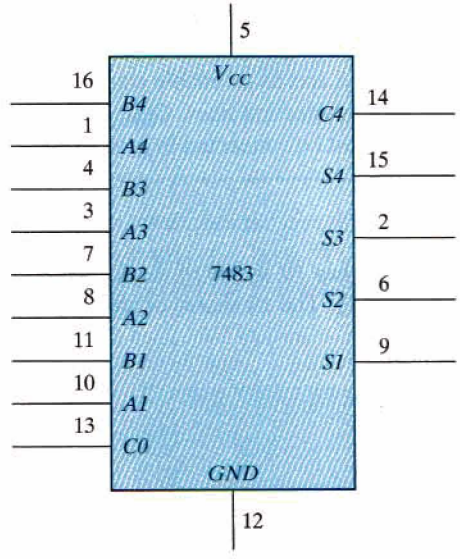


FIGURE 11.10
IC type 7483 four-bit binary adder

Parallel Adder

IC type 7483 is a four-bit binary parallel adder. The pin assignment is shown in Fig. 11.10. The 2 four-bit input binary numbers are *A1* through *A4* and *B1* through *B4*. The four-bit sum is obtained from *S1* through *S4*. *C0* is the input carry and *C4* the output carry.

Test the four-bit binary adder 7483 by connecting the power supply and ground terminals. Then connect the four *A* inputs to a fixed binary number, such as 1001, and the *B* inputs and the input carry to five toggle switches. The five outputs are applied to indicator lamps. Perform the addition of a few binary numbers and check that the output sum and output carry give the proper values. Show that when the input carry is equal to 1, it adds 1 to the output sum.

Adder–Subtractor

Two binary numbers can be subtracted by taking the 2's complement of the subtrahend and adding it to the minuend. The 2's complement can be obtained by taking the 1's complement and adding 1. To perform $A - B$, we complement the four bits of B, add them to the four bits of A, and add 1 through the input carry. This is done as shown in Fig. 11.11. The four XOR gates complement the bits of B when the mode select $M = 1$ (because $x \oplus 1 = x'$) and leave the bits of B unchanged when $M = 0$ (because $x \oplus 0 = x$). Thus, when the mode select M is equal to 1, the input carry $C0$ is equal to 1 and the sum output is A plus the 2's complement of B. When M is equal to 0, the input carry is equal to 0 and the sum generates $A + B$.

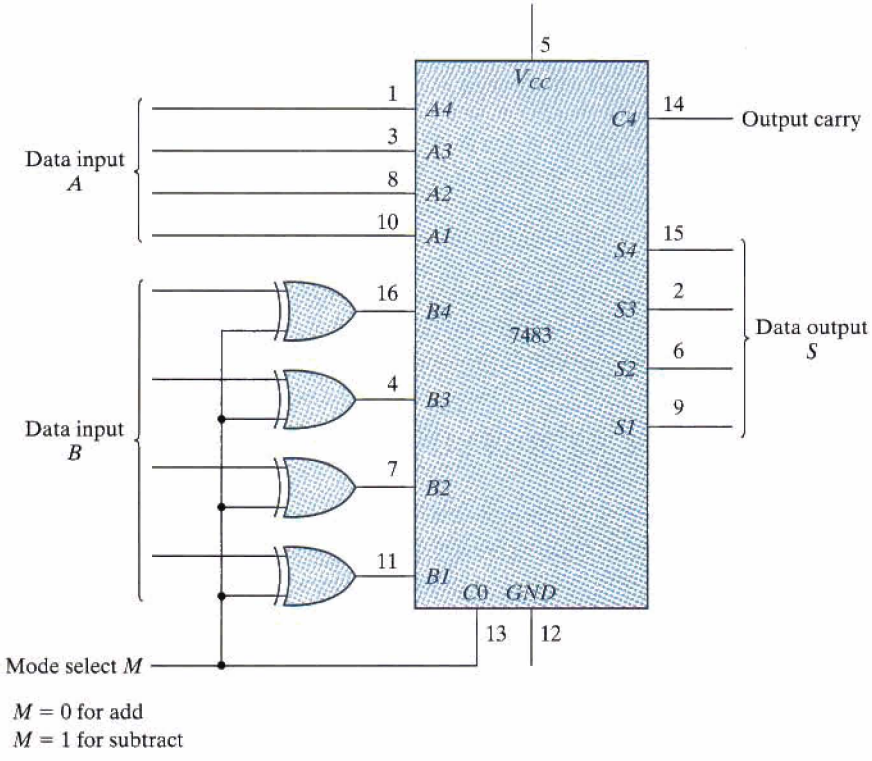


FIGURE 11.11
Four-bit adder–subtractor

Connect the adder–subtractor circuit and test it for proper operation. Connect the four A inputs to a fixed binary number 1001 and the B inputs to switches. Perform the following operations and record the values of the output sum and the output carry $C4$:

$$\begin{array}{ll} 9 + 5 & 9 - 5 \\ 9 + 9 & 9 - 9 \\ 9 + 15 & 9 - 15 \end{array}$$

Show that during addition, the output carry is equal to 1 when the sum exceeds 15. Also, show that when $A \geq B$, the subtraction operation gives the correct answer, $A - B$, and the output carry $C4$ is equal to 1, but when $A < B$, the subtraction gives the 2's complement of $B - A$ and the output carry is equal to 0.

Magnitude Comparator

The comparison of two numbers is an operation that determines whether one number is greater than, equal to, or less than the other number. Two numbers, A and B, can be compared by first subtracting $A - B$ as is done in Fig. 11.11. If the output in S is equal to zero, then $A = B$. The output carry from $C4$ determines the relative magnitudes of the numbers: When $C4 = 1$, $A \geq B$; when $C4 = 0$, $A < B$; and when $C4 = 1$ and $S \neq 0$, $A > B$.

It is necessary to supplement the subtractor circuit of Fig. 11.11 to provide the comparison logic. This is done with a combinational circuit that has five inputs—*S1* through *S4* and *C4*—and three outputs, designated by x, y, and z, so that

$$x = 1 \quad \text{if } A = B \quad (S = 0000)$$
$$y = 1 \quad \text{if } A < B \quad (C4 = 0)$$
$$z = 1 \quad \text{if } A > B \quad (C4 = 1 \text{ and } S \neq 0000)$$

The combinational circuit can be implemented with the 7404 and 7408 ICs.

Construct the comparator circuit and test its operation. Use at least two sets of numbers for A and B to check each of the outputs x, y, and z.

11.9 EXPERIMENT 8: FLIP-FLOPS

In this experiment, you will construct, test, and investigate the operation of various latches and flip-flops. The internal construction of latches and flip-flops can be found in Sections 5.3 and 5.4.

SR Latch

Construct an *SR* latch with two cross-coupled NAND gates. Connect the two inputs to switches and the two outputs to indicator lamps. Set the two switches to logic 1, and then momentarily turn each switch separately to the logic-0 position and back to 1. Obtain the function table of the circuit.

D Latch

Construct a *D* latch with four NAND gates (only one 7400 IC) and verify its function table.

Master–Slave Flip-Flop

Connect a master–slave *D* flip-flop using two *D* latches and an inverter. Connect the *D* input to a switch and the clock input to a pulser. Connect the output of the master latch to one indicator lamp and the output of the slave latch to another indicator lamp. Set the value of the input to the complement value of the output. Press the push button in the pulser and then release it to produce a single pulse. Observe that the master changes when the pulse goes positive and the slave follows the change when the pulse goes negative. Press the push button again a few times while observing the two indicator lamps. Explain the transfer sequence from input to master and from master to slave.

Disconnect the clock input from the pulser and connect it to a clock generator. Connect the complement output of the flip-flop to the *D* input. This causes the flip-flop to be complemented with each clock pulse. Using a dual-trace oscilloscope, observe the waveforms of the clock and the master and slave outputs. Verify that the delay between the master and the slave outputs is equal to the positive half of the clock cycle. Obtain a timing diagram showing the relationship between the clock waveform and the master and slave outputs.

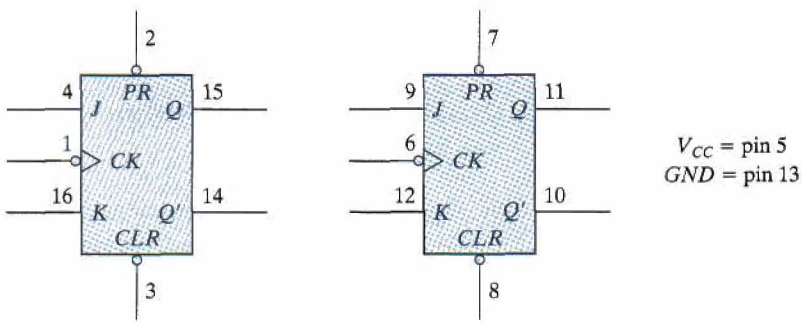
Edge-Triggered Flip-Flop

Construct a *D*-type positive-edge-triggered flip-flop using six NAND gates. Connect the clock input to a pulser, the *D* input to a toggle switch, and the output *Q* to an indicator lamp. Set the value of *D* to the complement of *Q*. Show that the flip-flop output changes only in response to a positive transition of the clock pulse. Verify that the output does not change when the clock input is logic 1, when the clock goes through a negative transition, or when the clock input is logic 0. Continue changing the *D* input to correspond to the complement of the *Q* output at all times.

Disconnect the input from the pulser and connect it to the clock generator. Connect the complement output *Q'* to the *D* input. This causes the output to be complemented with each positive transition of the clock pulse. Using a dual-trace oscilloscope, observe and record the timing relationship between the input clock and the output *Q*. Show that the output changes in response to a positive edge transition.

IC Flip-Flops

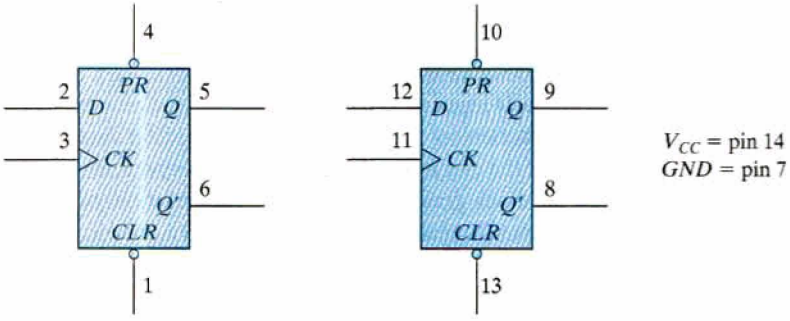
IC type 7476 consists of two *JK* master–slave flip-flops with preset and clear. The pin assignment for each flip-flop is shown in Fig. 11.12. The function table specifies the circuit's operation. The first three entries in the table specify the operation of the asynchronous preset and



Function table

Inputs					Outputs	
Preset	Clear	Clock	*J*	*K*	*Q*	*Q'*
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1	1
1	1	⎍	0	0	No change	
1	1	⎍	0	1	0	1
1	1	⎍	1	0	1	0
1	1	⎍	1	1	Toggle	

FIGURE 11.12
IC type 7476 dual *JK* master–slave flip-flops



Function table

Inputs				Outputs	
Preset	Clear	Clock	D	Q	Q'
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1	1
1	1	↑	0	0	1
1	1	↑	1	1	0
1	1	0	X	No change	

FIGURE 11.13
IC type 7474 dual *D* positive-edge-triggered flip-flops

clear inputs. These inputs behave like a NAND *SR* latch and are independent of the clock or the *J* and *K* inputs. (The X's indicate don't-care conditions.) The last four entries in the function table specify the operation of the clock with both the preset and clear inputs maintained at logic 1. The clock value is shown as a single pulse. The positive transition of the pulse changes the master flip-flop, and the negative transition changes the slave flip-flop as well as the output of the circuit. With $J = K = 0$, the output does not change. The flip-flop toggles, or is complemented, when $J = K = 1$. Investigate the operation of one 7476 flip-flop and verify its function table.

IC type 7474 consists of two *D* positive-edge-triggered flip-flops with preset and clear. The pin assignment is shown in Fig. 11.13. The function table specifies the preset and clear operations and the clock's operation. The clock is shown with an upward arrow to indicate that it is a positive-edge-triggered flip-flop. Investigate the operation of one of the flip-flops and verify its function table.

11.10 EXPERIMENT 9: SEQUENTIAL CIRCUITS

In this experiment, you will design, construct, and test three synchronous sequential circuits. Use IC type 7476 (Fig. 11.12) or 7474 (Fig. 11.13). Choose any type of gate that will minimize the total number of ICs. The design of synchronous sequential circuits is covered in Section 5.7.

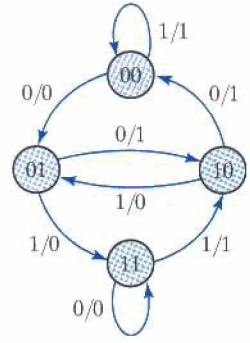


FIGURE 11.14
State diagram for Experiment 9

Up–Down Counter with Enable

Design, construct, and test a two-bit counter that counts up or down. An enable input E determines whether the counter is on or off. If $E = 0$, the counter is disabled and remains at its present count even though clock pulses are applied to the flip-flops. If $E = 1$, the counter is enabled and a second input, x, determines the direction of the count. If $x = 1$, the circuit counts upward with the sequence 00, 01, 10, 11, and the count repeats. If $x = 0$, the circuit counts downward with the sequence 11, 10, 01, 00, and the count repeats. Do not use E to disable the clock. Design the sequential circuit with E and x as inputs.

State Diagram

Design, construct, and test a sequential circuit whose state diagram is shown in Fig. 11.14. Designate the two flip-flops as A and B, the input as x, and the output as y.

Connect the output of the least significant flip-flop B to the input x, and predict the sequence of states and output that will occur with the application of clock pulses. Verify the state transition and output by testing the circuit.

Design of Counter

Design, construct, and test a counter that goes through the following sequence of binary states: 0, 1, 2, 3, 6, 7, 10, 11, 12, 13, 14, 15, and back to 0 to repeat. Note that binary states 4, 5, 8, and 9 are not used. The counter must be self-starting; that is, if the circuit starts from any one of the four invalid states, the count pulses must transfer the circuit to one of the valid states to continue the count correctly.

Check the circuit's operation for the required count sequence. Verify that the counter is self-starting. This is done by initializing the circuit to each unused state by means of the preset and clear inputs and then applying pulses to see whether the counter reaches one of the valid states.

11.11 EXPERIMENT 10: COUNTERS

In this experiment, you will construct and test various ripple and synchronous counter circuits. Ripple counters are discussed in Section 6.3 and synchronous counters are covered in Section 6.4.

Ripple Counter

Construct a four-bit binary ripple counter using two 7476 ICs (Fig. 11.12). Connect all asynchronous clear and preset inputs to logic 1. Connect the count-pulse input to a pulser and check the counter for proper operation.

Modify the counter so that it will count downward instead of upward. Check that each input pulse decrements the counter by 1.

Synchronous Counter

Construct a synchronous four-bit binary counter and check its operation. Use two 7476 ICs and one 7408 IC.

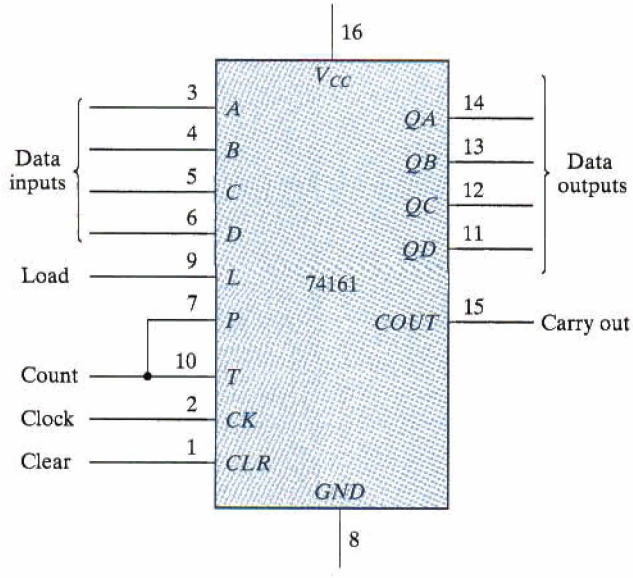
Decimal Counter

Design a synchronous BCD counter that counts from 0000 to 1001. Use two 7476 ICs and one 7408 IC. Test the counter for the proper sequence. Determine whether the counter is self-starting. This is done by initializing the counter to each of the six unused states by means of the preset and clear inputs. The application of pulses will transfer the counter to one of the valid states if the counter is self-starting.

Binary Counter with Parallel Load

IC type 74161 is a four-bit synchronous binary counter with parallel load and asynchronous clear. The internal logic is similar to that of the circuit shown in Fig. 6.14. The pin assignments to the inputs and outputs are shown in Fig. 11.15. When the load signal is enabled, the four data inputs are transferred into four internal flip-flops, *QA* through *QD*, with *QD* being the most significant bit. There are two count-enable inputs called *P* and *T*. Both must be equal to 1 for the counter to operate. The function table is similar to Table 6.6, with one exception: The load input in the 74161 is enabled when equal to 0. To load the input data, the clear input must be equal to 1 and the load input must be equal to 0. The two count inputs have don't-care conditions and may be equal to either 1 or 0. The internal flip-flops trigger on the positive transition of the clock pulse. The circuit functions as a counter when the load input is equal to 1 and both count inputs *P* and *T* are equal to 1. If either *P* or *T* goes to 0, the output does not change. The carry-out output is equal to 1 when all four data outputs are equal to 1. Perform an experiment to verify the operation of the 74161 IC according to the function table.

Show how the 74161 IC, together with a two-input NAND gate, can be made to operate as a synchronous BCD counter that counts from 0000 to 1001. Do not use the clear input. Use the NAND gate to detect the count of 1001, which then causes all 0's to be loaded into the counter.



Function table

Clear	Clock	Load	Count	Function
0	X	X	X	Clear outputs to 0
1	↑	0	X	Load input data
1	↑	1	1	Count to next binary value
1	↑	1	0	No change in output

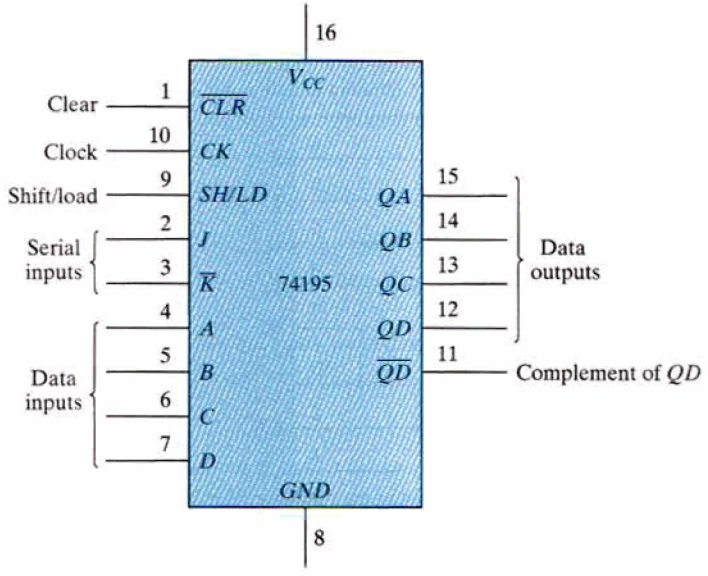
FIGURE 11.15
IC type 74161 binary counter with parallel load

11.12 EXPERIMENT 11: SHIFT REGISTERS

In this experiment, you will investigate the operation of shift registers. The IC to be used is the 74195 shift register with parallel load. Shift registers are explained in Section 6.2.

IC Shift Register

IC type 74195 is a four-bit shift register with parallel load and asynchronous clear. The pin assignments to the inputs and outputs are shown in Fig. 11.16. The single control line labeled *SH/LD* (shift/load) determines the synchronous operation of the register. When $SH/LD = 0$, the control input is in the load mode and the four data inputs are transferred into the four internal flip-flops, *QA* through *QD*. When $SH/LD = 1$, the control input is in the shift mode and the information in the register is shifted right from *QA* toward *QD*. The serial input into *QA* during the shift is determined from the *J* and $\overline{K}$ inputs. The two inputs behave like the *J* and the complement of *K* of a *JK* flip-flop. When both *J* and $\overline{K}$ are equal to 0, flip-flop *QA* is



Function table

Clear	Shift/ load	Clock	J	$\overline{K}$	Serial input	Function
0	X	X	X	X	X	Asynchronous clear
1	X	0	X	X	X	No change in output
1	0	↑	X	X	X	Load input data
1	1	↑	0	0	0	Shift from QA toward QD, $QA = 0$
1	1	↑	1	1	1	Shift from QA toward QD, $QA = 1$

FIGURE 11.16
IC type 74195 shift register with parallel load

cleared to 0 after the shift. If both inputs are equal to 1, QA is set to 1 after the shift. The other two conditions for the J and $\overline{K}$ inputs provide a complement or no change in the output of flip-flop QA after the shift.

The function table for the 74195 shows the mode of operation of the register. When the clear input goes to 0, the four flip-flops clear to 0 asynchronously—that is, without the need of a clock. Synchronous operations are affected by a positive transition of the clock. To load the input data, *SH/LD* must be equal to 0 and a positive clock-pulse transition must occur. To shift right, *SH/LD* must be equal to 1. The J and $\overline{K}$ inputs must be connected together to form the serial input.

Perform an experiment that will verify the operation of the 74195 IC. Show that it performs all the operations listed in the function table. Include in your function table the two conditions for $J\overline{K} = 01$ and 10.

Ring Counter

A ring counter is a circular shift register with the signal from the serial output *QD* going into the serial input. Connect the *J* and $\overline{K}$ input together to form the serial input. Use the load condition to preset the ring counter to an initial value of 1000. Rotate the single bit with the shift condition and check the state of the register after each clock pulse.

A switch-tail ring counter uses the complement output of *QD* for the serial input. Preset the switch-tail ring counter to 0000 and predict the sequence of states that will result from shifting. Verify your prediction by observing the state sequence after each shift.

Feedback Shift Register

A feedback shift register is a shift register whose serial input is connected to some function of selected register outputs. Connect a feedback shift register whose serial input is the exclusive-OR of outputs *QC* and *QD*. Predict the sequence of states of the register, starting from state 1000. Verify your prediction by observing the state sequence after each clock pulse.

Bidirectional Shift Register

The 74195 IC can shift only right from *QA* toward *QD*. It is possible to convert the register to a bidirectional shift register by using the load mode to obtain a shift-left operation (from *QD* toward *QA*). This is accomplished by connecting the output of each flip-flop to the input of the flip-flop on its left and using the load mode of the *SH/LD* input as a shift-left control. Input *D* becomes the serial input for the shift-left operation.

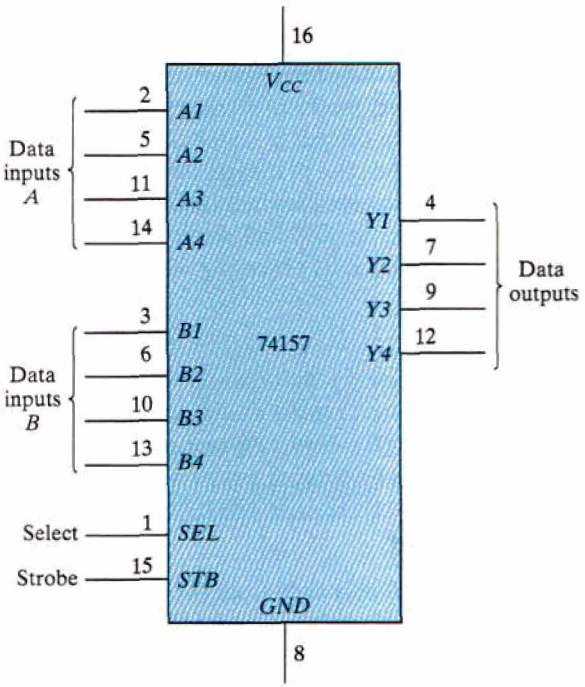
Connect the 74195 as a bidirectional shift register (without parallel load). Connect the serial input for shift right to a toggle switch. Construct the shift left as a ring counter by connecting the serial output *QA* to the serial input *D*. Clear the register and then check its operation by shifting a single 1 from the serial input switch. Shift right three more times and insert 0's from the serial input switch. Then rotate left with the shift-left (load) control. The single 1 should remain visible while shifting.

Bidirectional Shift Register with Parallel Load

The 74195 IC can be converted to a bidirectional shift register with parallel load in conjunction with a multiplexer circuit. We will use IC type 74157 for this purpose. The 74157 is a quadruple two-to-one-line multiplexer whose internal logic is shown in Fig. 4.26. The pin assignments to the inputs and outputs of the 74157 are shown in Fig. 11.17. Note that the enable input is called a strobe in the 74157.

Construct a bidirectional shift register with parallel load using the 74195 register and the 74157 multiplexer. The circuit should be able to perform the following operations:

1. Asynchronous clear
2. Shift right
3. Shift left
4. Parallel load
5. Synchronous clear



Function table

Strobe	Select	Data outputs Y
1	X	All 0's
0	0	Select data inputs A
0	1	Select data inputs B

FIGURE 11.17
IC type 74157 quadruple 2 × 1 multiplexers

Derive a table for the five operations as a function of the clear, clock, and *SH/LD* inputs of the 74195 and the strobe and select inputs of the 74157. Connect the circuit and verify your function table. Use the parallel-load condition to provide an initial value to the register, and connect the serial outputs to the serial inputs of both shifts in order not to lose the binary information while shifting.

11.13 EXPERIMENT 12: SERIAL ADDITION

In this experiment, you will construct and test a serial adder–subtractor circuit. Serial addition of two binary numbers can be done by means of shift registers and a full adder, as explained in Section 6.2.

Serial Adder

Starting from the diagram of Fig. 6.6, design and construct a four-bit serial adder using the following ICs: 74195 (two), 7408, 7486, and 7476. Provide a facility for register B to accept parallel data from four toggle switches, and connect its serial input to ground so that 0's are shifted into register B during the addition. Provide a toggle switch to clear the registers and the flip-flop. Another switch will be needed to specify whether register B is to accept parallel data or is to be shifted during the addition.

Testing the Adder

To test your serial adder, perform the binary addition $5 + 6 + 15 = 26$. This is done by first clearing the registers and the carry flip-flop. Parallel load the binary value 0101 into register B. Apply four pulses to add B to A serially, and check that the result in A is 0101. (Note that clock pulses for the 7476 must be as shown in Fig. 11.12.) Parallel load 0110 into B and add it to A serially. Check that A has the proper sum. Parallel load 1111 into B and add to A. Check that the value in A is 1010 and that the carry flip-flop is set.

Clear the registers and flip-flop and try a few other numbers to verify that your serial adder is functioning properly.

Serial Adder–Subtractor

If we follow the procedure used in Section 6.2 for the design of a serial subtractor (that subtracts $A - B$), we will find that the output difference is the same as the output sum, but that the input to the J and K of the borrow flip-flop needs the complement of QD (available in the 74195). Using the other two XOR gates from the 7486, convert the serial adder to a serial adder–subtractor with a mode control M. When $M = 0$, the circuit adds $A + B$. When $M = 1$, the circuit subtracts $A - B$ and the flip-flop holds the borrow instead of the carry.

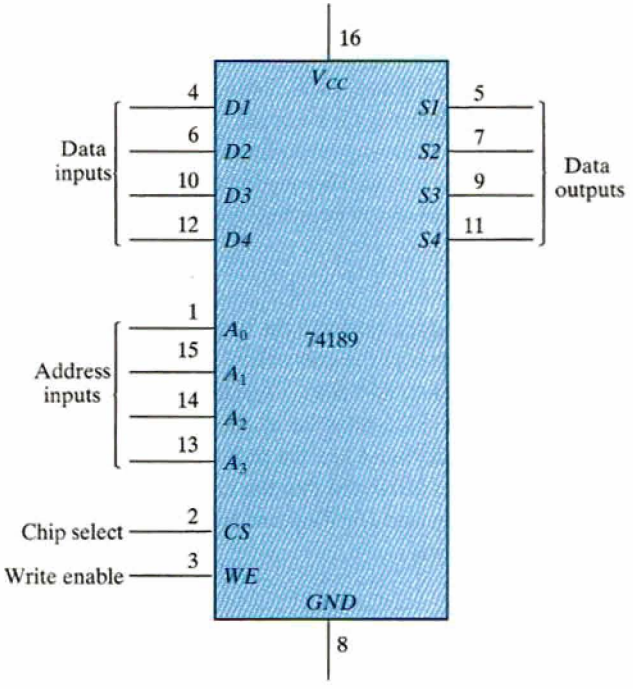
Test the adder part of the circuit by repeating the operations recommended to ensure that the modification did not change the operation. Test the serial subtractor part by performing the subtraction $15 - 4 - 5 - 13 = -7$. Binary 15 can be transferred to register A by first clearing it to 0 and adding 15 from B. Check the intermediate results during the subtraction. Note that -7 will appear as the 2's complement of 7 with a borrow of 1 in the flip-flop.

11.14 EXPERIMENT 13: MEMORY UNIT

In this experiment, you will investigate the behavior of a random-access memory (RAM) unit and its storage capability. The RAM will be used to simulate a read-only memory (ROM). The ROM simulator will then be used to implement combinational circuits, as explained in Section 7.5. The memory unit is discussed in Sections 7.2 and 7.3.

IC RAM

IC type 74189 is a 16×4 random-access memory. The internal logic is similar to the circuit shown in Fig. 7.6 for a 4×4 RAM. The pin assignments to the inputs and outputs are shown in Fig. 11.18. The four address inputs select 1 of 16 words in the memory. The least significant bit



Function table

CS	*WE*	Operation	Data outputs
0	0	Write	High impedance
0	1	Read	Complement of selected word
1	X	Disable	High impedance

FIGURE 11.18
IC type 74189 16 × 4 RAM

of the address is A and the most significant is A_3. The chip select (*CS*) input must be equal to 0 to enable the memory. If *CS* is equal to 1, the memory is disabled and all four outputs are in a high-impedance state. The write enable (*WE*) input determines the type of operation, as indicated in the function table. The write operation is performed when $WE = 0$. This operation is a transfer of the binary number from the data inputs into the selected word in memory. The read operation is performed when $WE = 1$. This operation transfers the complemented value stored in the selected word into the output data lines. The memory has three-state outputs to facilitate memory expansion.

Testing the RAM

Since the outputs of the 74189 produce the complemented values, we need to insert four inverters to change the outputs to their normal value. The RAM can be tested after making the

following connections: Connect the address inputs to a binary counter using the 7493 IC (shown in Fig. 11.3). Connect the four data inputs to toggle switches and the data outputs to four 7404 inverters. Provide four indicator lamps for the address and four more for the outputs of the inverters. Connect input *CS* to ground and *WE* to a toggle switch (or a pulser that provides a negative pulse). Store a few words into the memory, and then read them to verify that the write and read operations are functioning properly. You must be careful when using the *WE* switch. Always leave the *WE* input in the read mode, unless you want to write into memory. The proper way to write is first to set the address in the counter and the inputs in the four toggle switches. Then, store the word in memory, flip the *WE* switch to the write position and return it to the read position. Be careful not to change the address or the inputs when *WE* is in the write mode.

ROM Simulator

A ROM simulator is obtained from a RAM operated in the read mode only. The pattern of 1's and 0's is first entered into the simulating RAM by placing the unit momentarily in the write mode. Simulation is achieved by placing the unit in the read mode and taking the address lines as inputs to the ROM. The ROM can then be used to implement any combinational circuit.

Implement a combinational circuit using the ROM simulator that converts a four-bit binary number to its equivalent Gray code as defined in Table 1.6. This is done as follows: Obtain the truth table of the code converter. Store the truth table into the 74189 memory by setting the address inputs to the binary value and the data inputs to the corresponding Gray code value. After all 16 entries of the table are written into memory, the ROM simulator is set by permanently connecting the *WE* line to logic 1. Check the code converter by applying the inputs to the address lines and verifying the correct outputs in the data output lines.

Memory Expansion

Expand the memory unit to a 32 × 4 RAM using two 74189 ICs. Use the *CS* inputs to select between the two ICs. Note that since the data outputs are three-stated, you can tie pairs of terminals together to obtain a logic OR operation between the two ICs. Test your circuit by using it as a ROM simulator that adds a three-bit number to a two-bit number to produce a four-bit sum. For example, if the input of the ROM is 10110, then the output is calculated to be 101 + 10 = 0111. (The first three bits of the input represent 5, the last two bits represent 2, and the output sum is binary 7.) Use the counter to provide four bits of the address and a switch for the fifth bit of the address.

11.15 EXPERIMENT 14: LAMP HANDBALL

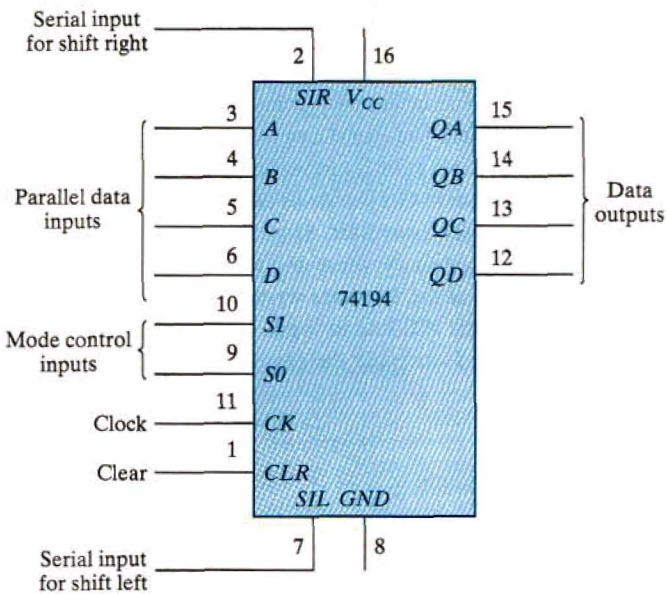
In this experiment, you will construct an electronic game of handball, using a single light to simulate the moving ball. The experiment demonstrates the application of a bidirectional shift register with parallel load. It also shows the operation of the asynchronous inputs of flip-flops. We will first introduce an IC that is needed for the experiment and then present the logic diagram of the simulated lamp handball game.

IC Type 74194

This is a four-bit bidirectional shift register with parallel load. The internal logic is similar to that shown in Fig. 6.7. The pin assignments to the inputs and outputs are shown in Fig. 11.19. The two mode-control inputs determine the type of operation, as specified in the function table.

Logic Diagram

The logic diagram of the electronic lamp handball game is shown in Fig. 11.20. It consists of two 74194 ICs, a dual *D* flip-flop 7474 IC, and three gate ICs: the 7400, 7404, and 7408. The ball is simulated by a moving light that is shifted left or right through the bidirectional shift register. The rate at which the light moves is determined by the frequency of the clock. The



Function table

Clear	Clock	Mode S1	Mode S0	Function
0	X	X	X	Clear outputs to 0
1	↑	0	0	No change in output
1	↑	0	1	Shift right in the direction from *QA* to *QD*. *SIR* to *QA*
1	↑	1	0	Shift left in the direction from *QD* to *QA*. *SIL* to *QD*
1	↑	1	1	Parallel-load input data

FIGURE 11.19
IC type 74194 bidirectional shift register with parallel load

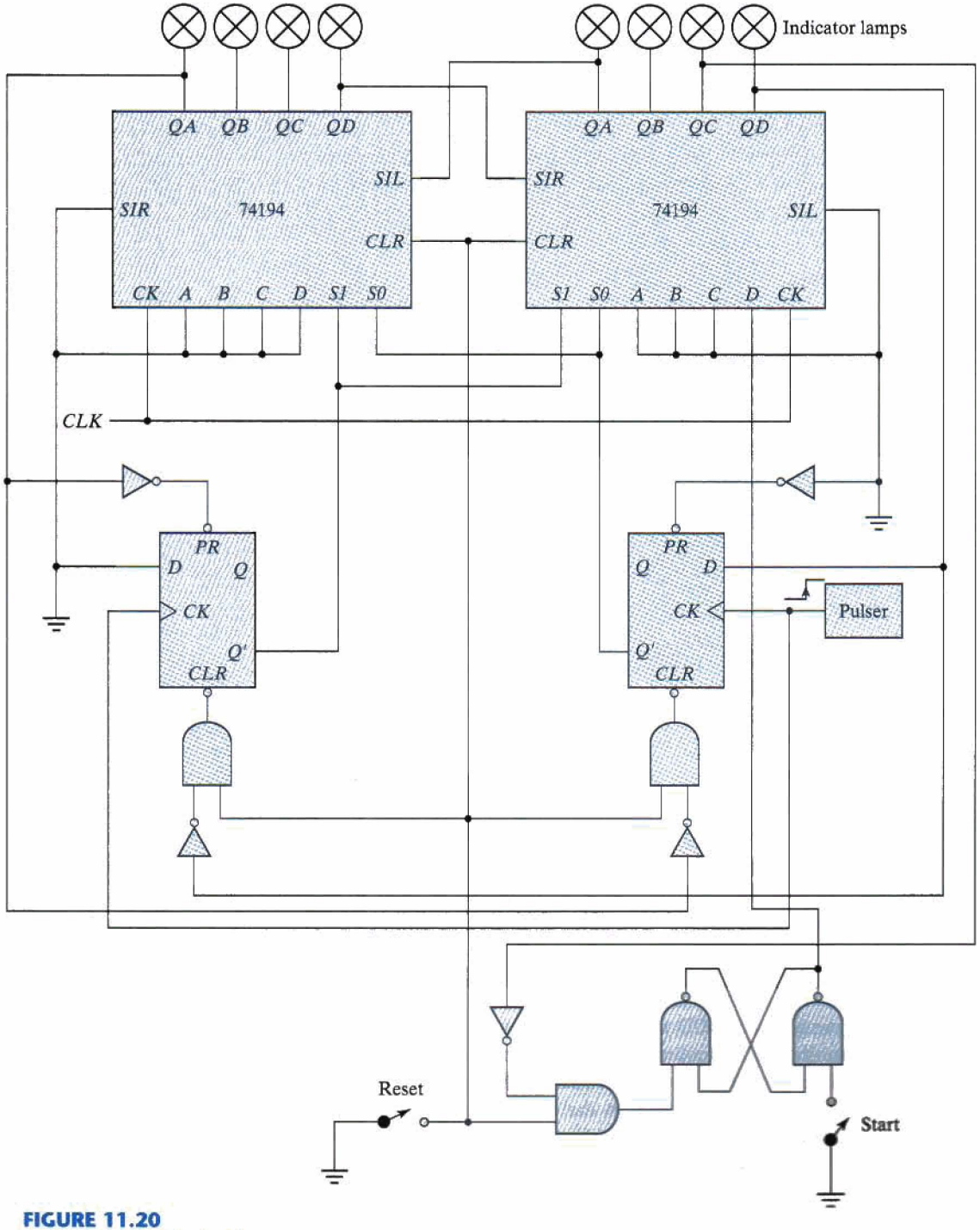


FIGURE 11.20
Lamp handball logic diagram

circuit is first initialized with the *reset* switch. The *start* switch starts the game by placing the ball (an indicator lamp) at the extreme right. The player must press the pulser push button to start the ball moving to the left. The single light shifts to the left until it reaches the leftmost position (the wall), at which time the ball returns to the player by reversing the direction of shift of the moving light. When the light is again at the rightmost position, the player must press the pulser again to reverse the direction of shift. If the player presses the pulser too soon or too late, the ball disappears and the light goes off. The game can be restarted by turning the start switch on and then off. The start switch must be open (logic 1) during the game.

Circuit Analysis

Prior to connecting the circuit, analyze the logic diagram to ensure that you understand how the circuit operates. In particular, try to answer the following questions:

1. What is the function of the reset switch?
2. How does the light in the rightmost position come on when the start switch is grounded? Why is it necessary to place the start switch in the logic-1 position before the game starts?
3. What happens to the two mode-control inputs, *S1* and *S0*, once the ball is set in motion?
4. What happens to the mode-control inputs and to the ball if the pulser is pressed while the ball is moving to the left? What happens if the ball is moving to the right, but has not yet reached the rightmost position?
5. If the ball has returned to the rightmost position, but the pulser has not yet been pressed, what is the state of the mode-control inputs if the pulser is pressed? What happens if it is not pressed?

Playing the Game

Wire the circuit of Fig. 11.20. Test the circuit for proper operation by playing the game. Note that the pulser must provide a positive-edge transition and that both the reset and start switches must be open (i.e., must be in the logic-1 state) during the game. Start with a low clock rate, and increase the clock frequency to make the handball game more challenging.

Counting the Number of Losses

Design a circuit that keeps score of the number of times the player loses while playing the game. Use a BCD-to-seven-segment decoder and a seven-segment display, as in Fig. 11.8, to display the count from 0 through 9. Counting is done with either the 7493 as a ripple decimal counter or the 74161 and a NAND gate as a synchronous decimal counter. The display should show 0 when the circuit is reset. Every time the ball disappears and the light goes off, the display should increase by 1. If the light stays on during the play, the number in the display should not change. The final design should be an automatic scoring circuit, with the decimal display incremented automatically each time the player loses when the light disappears.

Lamp Ping-Pong™

Modify the circuit of Fig. 11.20 so as to obtain a lamp Ping-Pong game. Two players can participate in this game, with each player having his or her own pulser. The player with the right pulser returns the ball when it is in the extreme right position, and the player with the left pulser returns the ball when it is in the extreme left position. The only modification required for the Ping-Pong game is a second pulser and a change of a few wires.

With a second start circuit, the game can be made to start by either one of the two players (i.e., either one serves). This addition is optional.

11.16 EXPERIMENT 15: CLOCK-PULSE GENERATOR

In this experiment, you will use an IC timer unit and connect it to produce clock pulses at a given frequency. The circuit requires the connection of two external resistors and two external capacitors. The cathode-ray oscilloscope is used to observe the waveforms and measure the frequency of the pulses.

IC Timer

IC type 72555 (or 555) is a precision timer circuit whose internal logic is shown in Fig. 11.21. (The resistors, R_A and R_B, and the two capacitors are not part of the IC.) The circuit consists of two voltage comparators, a flip-flop, and an internal transistor. The voltage division from $V_{CC} = 5$ V through the three internal resistors to ground produces $\frac{2}{3}$ and $\frac{1}{3}$ of V_{CC} (3.3 V and 1.7 V, respectively) into the fixed inputs of the comparators. When the threshold input at pin 6 goes above 3.3 V, the upper comparator resets the flip-flop and the output goes low to about 0 V. When the trigger input at pin 2 goes below 1.7 V, the lower comparator sets the flip-flop and the output goes high to about 5 V. When the output is low, Q' is high and the base–emitter junction of the transistor is forward biased. When the output is high, Q' is low and the transistor is cut off. (See Section 10.3.) The timer circuit is capable of producing accurate time delays controlled by an external RC circuit. In this experiment, the IC timer will be operated in the astable mode to produce clock pulses.

Circuit Operation

Figure 11.21 shows the external connections for astable operation of the circuit. Capacitor C charges through resistors R_A and R_B when the transistor is cut off and discharges through R_B when the transistor is forward biased and conducting. When the charging voltage across capacitor C reaches 3.3 V, the threshold input at pin 6 causes the flip-flop to reset and the transistor turns on. When the discharging voltage reaches 1.7 V, the trigger input at pin 2 causes the flip-flop to set and the transistor turns off. Thus, the output continually alternates between two voltage levels at the output of the flip-flop. The output remains high for a duration equal to the charge time. This duration is determined from the equation

$$t_H = 0.693(R_A + R_B)C$$

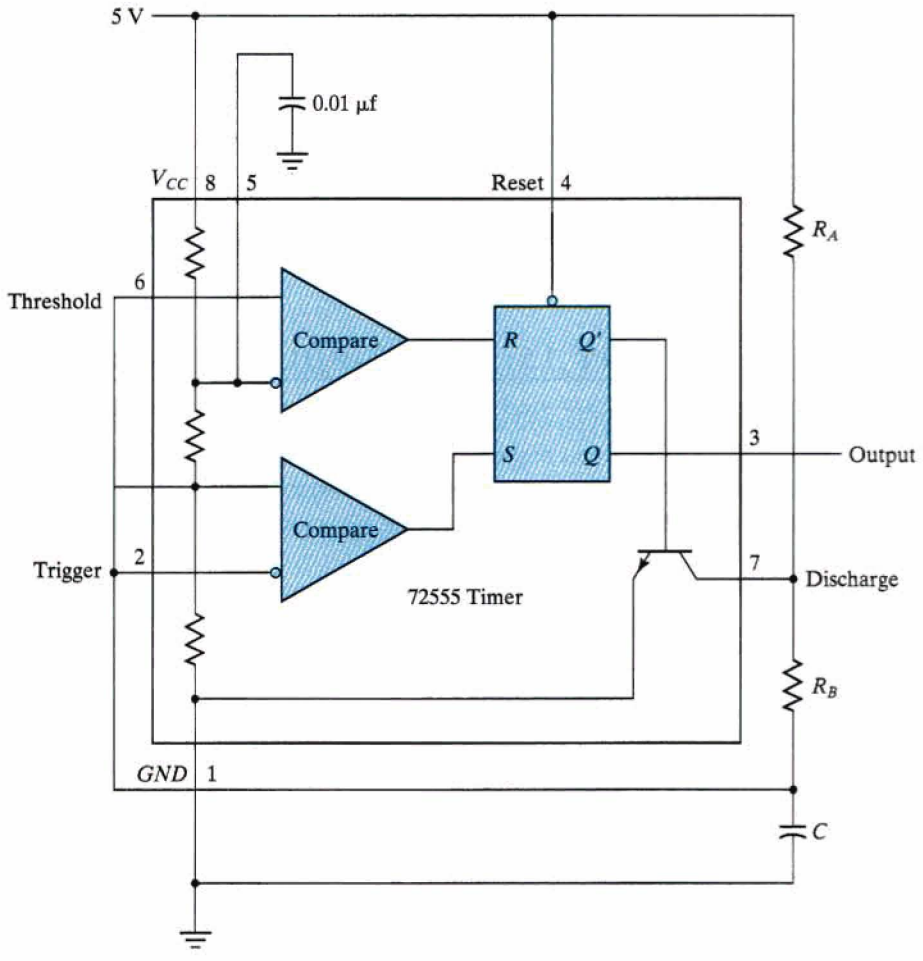


FIGURE 11.21
IC type 72555 timer connected as a clock-pulse generator

The output remains low for a duration equal to the discharge time. This duration is determined from the equation

$$t_L = 0.693 R_B C$$

Clock-Pulse Generator

Starting with a capacitor C of 0.001 μF, calculate values for R_A and R_B to produce clock pulses, as shown in Fig. 11.22. The pulse width is 1 μs in the low level and repeats at a frequency rate of 100 kHz (every 10 μs). Connect the circuit and check the output in the oscilloscope.

Observe the output across the capacitor C, and record its two levels to verify that they are between the trigger and threshold values.

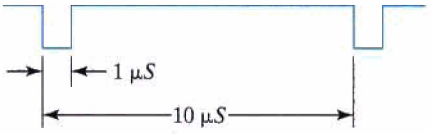


FIGURE 11.22
Output waveform for clock generator

Observe the waveform in the collector of the transistor at pin 7 and record all pertinent information. Explain the waveform by analyzing the circuit's action.

Connect a variable resistor (potentiometer) in series with R_A to produce a variable-frequency pulse generator. The low-level duration remains at 1 μs. The frequency should range from 20 to 100 kHz.

Change the low-level pulses to high-level pulses with a 7404 inverter. This will produce positive pulses of 1 μs with a variable-frequency range.

11.17 EXPERIMENT 16: PARALLEL ADDER AND ACCUMULATOR

In this experiment, you will construct a four-bit parallel adder whose sum can be loaded into a register. The numbers to be added will be stored in a random-access memory. A set of binary numbers will be selected from memory and their sum will be accumulated in the register.

Block Diagram

Use the RAM circuit from the memory experiment of Section 11.14, a four-bit parallel adder, a four-bit shift register with parallel load, a carry flip-flop, and a multiplexer to construct the circuit. The block diagram and the ICs to be used are shown in Fig. 11.23. Information can be written into RAM from data in four switches or from the four-bit data available in the outputs of the register. The selection is done by means of a multiplexer. The data in RAM can be added to the contents of the register and the sum transferred back to the register.

Control of Register

Provide toggle switches to control the 74194 register and the 7476 carry flip-flop as follows:

(a) A LOAD condition transfers the sum to the register and the output carry to the flip-flop upon the application of a clock pulse.

(b) A SHIFT condition shifts the register right with the carry from the carry flip-flop transferred into the leftmost position of the register upon the application of a clock pulse. The value in the carry flip-flop should not change during the shift.

(c) A NO-CHANGE condition leaves the contents of the register and flip-flop unchanged even when clock pulses are applied.

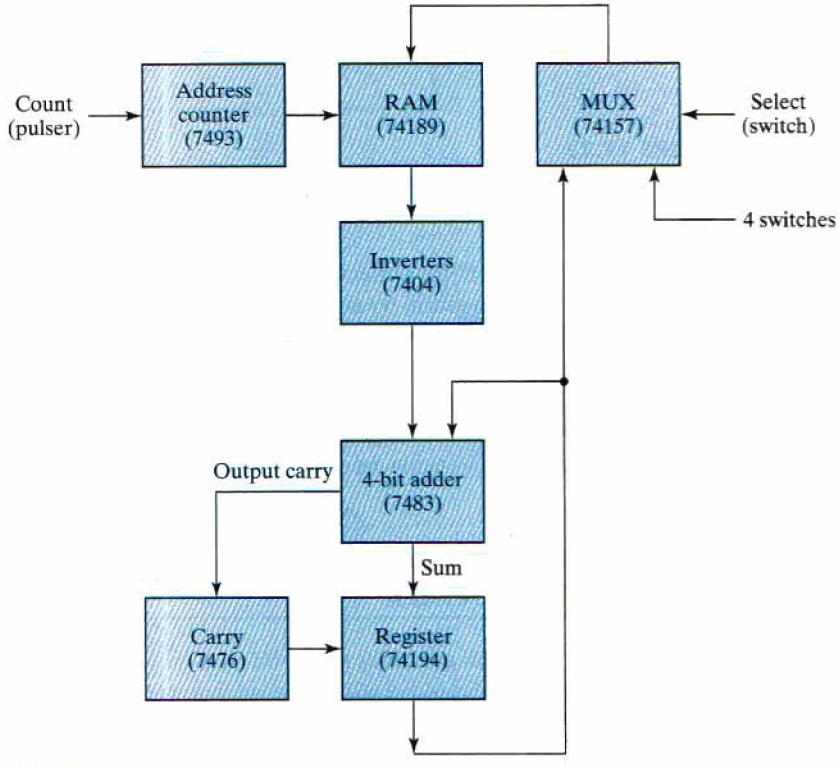


FIGURE 11.23
Block diagram of a parallel adder for Experiment 16

Carry Circuit

To conform with the preceding specifications, it is necessary to provide a circuit between the output carry from the adder and the J and K inputs of the 7476 flip-flop so that the output carry is transferred into the flip-flop (whether it is equal to 0 or 1) only when the LOAD condition is activated and a pulse is applied to the clock input of the flip-flop. The carry flip-flop should not change if the LOAD condition is disabled or the SHIFT condition is enabled.

Detailed Circuit

Draw a detailed diagram showing all the wiring between the ICs. Connect the circuit, and provide indicator lamps for the outputs of the register and carry flip-flop and for the address and output data of the RAM.

Checking the Circuit

Store the numbers 0110, 1110, 1101, 0101, and 0011 in RAM and then add them to the register one at a time. Start with a cleared register and flip-flop. Predict the values in the output of the register and carry after each addition in the following sum, and verify your results:

$$0110 + 1110 + 1101 + 0101 + 0011$$

Circuit Operation

Clear the register and the carry flip-flop to zero, and store the following four-bit numbers in RAM in the indicated addresses:

Address	Content
0	0110
3	1110
6	1101
9	0101
12	0011

Now perform the following four operations:

1. Add the contents of address 0 to the contents of the register, using the LOAD condition.
2. Store the sum from the register into RAM at address 1.
3. Shift right the contents of the register and carry with the SHIFT condition.
4. Store the shifted contents of the register at address 2 of RAM.

Check that the contents of the first three locations in RAM are as follows:

Address	Contents
0	0110
1	0110
2	0011

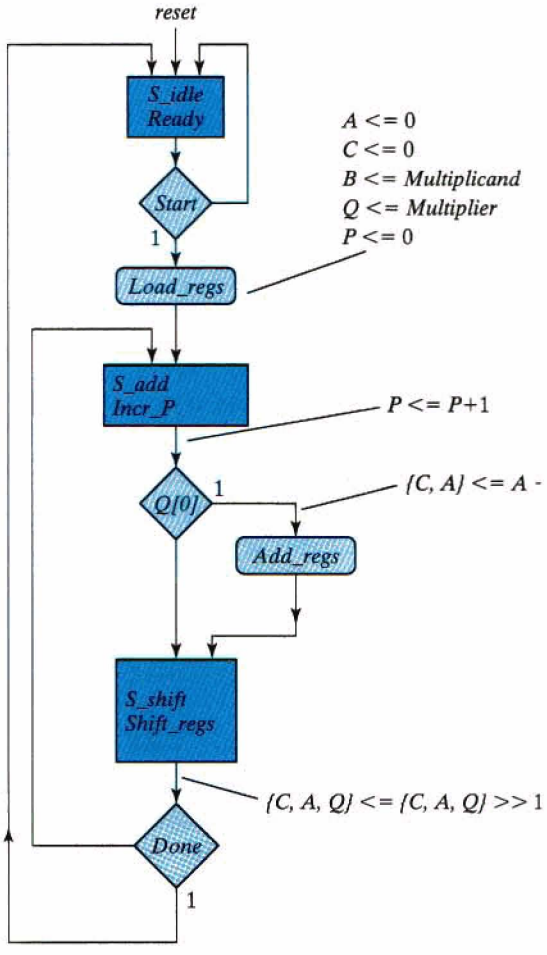
Repeat the foregoing four operations for each of the other four binary numbers stored in RAM. Use addresses 4, 7, 10, and 13 to store the sum from the register in step 2. Use addresses 5, 8, 11, and 14 to store the shifted value from the register in step 4. Predict what the contents of RAM at addresses 0 through 14 would be, and check to verify your results.

11.18 EXPERIMENT 17: BINARY MULTIPLIER

In this experiment, you will design and construct a circuit that multiplies 2 four-bit unsigned numbers to produce an eight-bit product. An algorithm for multiplying two binary numbers is presented in Section 8.7. The algorithm implemented in this experiment differs from the one described in Figures 8.14 and 8.15, by treating only a four-bit datapath and by incrementing, instead of decrementing, a bit counter.

Block Diagram

The ASMD chart and block diagram of the binary multiplier with those ICs recommended to be used are shown in Fig. 11.24(a) and (b). The multiplicand, B, is available from four switches instead of a register. The multiplier, Q, is obtained from another set of four switches. The product is displayed with eight indicator lamps. Counter P is initialized to 0 and then incremented after each partial product is formed. When the counter reaches the count of four, output *Done* becomes 1 and the multiplication operation terminates.

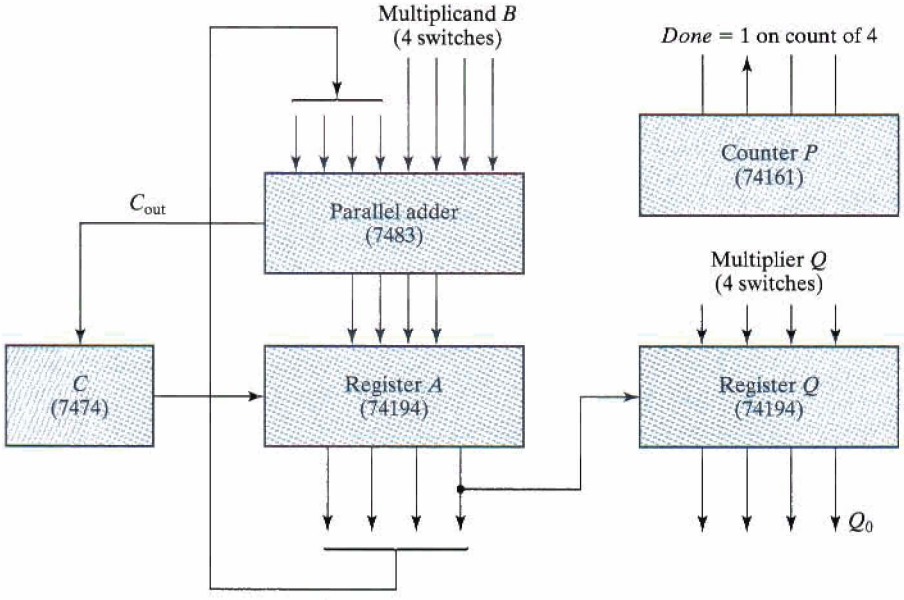


(a) ASMD chart

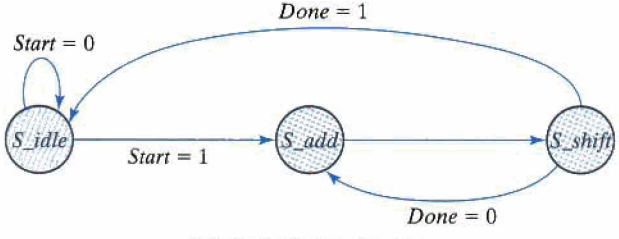
FIGURE 11.24
ASMD chart, block diagram of the datapath, control state diagram, and register operations of the binary multiplier circuit

Control of Registers

The ASMD chart for the binary multiplier in Figure 11.24(a) shows that the three registers and the carry flip-flop of the datapath unit are controlled with signals *Load_regs*, *Incr_P*, *Add_regs*, and *Shift_regs*. The external input signals of the control unit are *clock*, *reset_b* (active-low), and *Start*; another input to the control unit is the internal status signal, *Done*, which is formed by the datapath unit to indicate that the counter has reached a count of four, corresponding to the number of bits in the multiplier. *Load_regs* clears the product register (A) and the carry flip-flop (C), loads the multiplicand into register B, loads the multiplier into register Q, and clears the bit counter. *Incr_P* increments the bit counter concurrently with the accumulation of a partial product. *Add_regs* adds the multiplicand to A, if the least significant bit of the shifted multiplier



(b) Datapath block program



(c) Control state diagram

State Transition		Register Operations	Control signal
From	To		
S_idle		Initial state reached by reset action	
S_idle	*S_add*	$A <= 0, C <= 0, P <= 0$	*Load_regs*
S_add	*S_shift*	$P <= P + 1$ if $(Q[0])$ then $(A <= A + B, C <= C_{out})$	*Incr_P* *Add_regs*
S_shift		shift right $\{CAQ\}, C <= 0$	*Shift_regs*

(d) Register operations

FIGURE 11.24
(Continued)

$(Q[0])$ is 1. Flip-flop C accommodates a carry that results from the addition. The concatenated register CAQ is updated by storing the result of shifting its contents one bit to the right. *Shift_regs* shifts CAQ one bit to the right, which also clears flip-flop C.

The state diagram for the control unit is shown in Fig. 11.24(c). Note that it does not show the register operations of the datapath unit or the output signals that control them. That information is apparent in Figure 11.24(d). Note that *Incr_P* and *Shift_regs* are generated unconditionally in states *S_add* and *S_shift*, respectively. *Load_regs* is generated under the condition that *Start* is asserted conditionally while the state is in *S_idle*; *Add_regs* is asserted conditionally in *S_add* if $Q[0] = 1$

Multiplication Example

Before connecting the circuit, make sure that you understand the operation of the multiplier. To do this, construct a table similar to Table 8.5, but with $B = 1111$ for the multiplicand and $Q = 1011$ for the multiplier. Along with each comment listed on the left side of the table, specify the state.

Datapath Design

Draw a detailed diagram of the datapath part of the multiplier, showing all IC pin connections. Generate the four control signals with switches, and use them to provide the required control operations for the various registers. Connect the circuit and check that each component is functioning properly. With the control signals at 0, set the multiplicand switches to 1111 and the multiplier switches to 1011. Assert the control signals manually by means of the control switches, as specified by the state diagram of Fig. 11.24(c). Apply a single pulse while in each control state, and observe the outputs of registers A and Q and the values in C and P. Compare these outputs with the numbers in your numerical example to verify that the circuit is functioning properly. Note that IC type 74161 has master–slave flip-flops. To operate it manually, it is necessary that the single clock pulse be a negative pulse.

Design of Control

Design the control circuit specified by the state diagram. You can use any method of control implementation discussed in Section 8.8.

Choose the method that minimizes the number of ICs. Verify the operation of the control circuit prior to its connection to the datapath unit.

Checking the Multiplier

Connect the outputs of the control circuit to the datapath unit, and verify the total circuit operation by repeating the steps of multiplying 1111 by 1011. The single clock pulses should now sequence the control states as well. (Remove the manual switches.) The start signal (*Start*) can be generated with a switch that is on while the control is in state *S_idle*.

Generate the start signal (*Start*) with a pulser or any other short pulse, and operate the multiplier with continuous clock pulses from a clock generator. Pressing the pulser for *Start* should initiate the multiplication operation, and upon its completion, the product should be displayed in the A and Q registers. Note that the multiplication will be repeated as long as signal *Start* is enabled. Make sure that *Start* goes back to 0. Then set the switches to two other four-bit numbers

and press *Start* again. The new product should appear at the outputs. Repeat the multiplication of a few numbers to verify the operation of the circuit.

11.19 EXPERIMENT 18: ASYNCHRONOUS SEQUENTIAL CIRCUITS

In this experiment, you will analyze and design asynchronous sequential circuits. These types of circuits are presented in Chapter 9.

Analysis Example

The analysis of asynchronous sequential circuits with *SR* latches is outlined in Section 9.3. Analyze the circuit of Fig. P9.9 (shown with Problem 9.9) by deriving the transition table and output map of the circuit. From the transition table and output map, determine (a) what happens to output Q when input x_1 is a 1 irrespective of the value of input x_2, (b) what happens to output Q when input x_2 is a 1 and x_1 is equal to 0, and (c) what happens to output Q when both inputs go back to 0?

Connect the circuit and show that it operates according to the way you analyzed it.

Design Example

The circuit of a positive-edge-triggered D-type flip-flop is shown in Fig. 5.10. The circuit of a negative-edge T-type flip-flop is shown in Fig. 9.46. Using the six-step procedure recommended in Section 9.8, design, construct, and test a D-type flip-flop that triggers on both the positive and negative transitions of the clock. The circuit has two inputs—D and C—and a single output, Q. The value of D at the time C changes from 0 to 1 becomes the flip-flop output Q. The output remains unchanged irrespective of the value of D, as long as C remains at 1. On the next clock transition, the output is again updated to the value of D when C changes from 1 to 0. The output then remains unchanged as long as C remains at 0.

11.20 VERILOG HDL SIMULATION EXPERIMENTS AND RAPID PROTOTYPING WITH FPGAS

Field programmable gate arrays (FPGAs) are used by industry to implement logic when the system is complex, the time-to-market is short, the performance (e.g., speed) of an FPGA is acceptable, and the volume of potential sales does not warrant the investment in a standard cell-based ASIC. Circuits can be rapidly prototyped into an FPGA using an HDL. Once the HDL model is verified, the description is synthesized and mapped into the FPGA. FPGA vendors provide software tools for synthesizing the HDL description of a circuit into an optimized gate-level description and mapping (fitting) the resulting netlist into the resources of their FPGA. This process avoids the detailed assembly of ICs that is required by composing a circuit on a breadboard, and the process involves significantly less risk of failure, because it is easier and faster to edit an HDL description than to re-wire a breadboard.

Most of the hardware experiments outlined in this chapter can be supplemented by a corresponding software procedure using the Verilog hardware description language (HDL). A Verilog compiler and simulator are necessary for these supplements. The supplemental experiments have two levels of engagement. In the first, the circuits that are specified in the hands-on laboratory experiments can be described, simulated, and verified using Verilog and a simulator. In the second, if a suitable FPGA prototyping board is available (e.g., see www.digilentinc.com), the hardware experiments can be done by synthesizing the Verilog descriptions and implementing the circuits in an FPGA. Where appropriate, the identity of the individual (structural) hardware units (e.g., a 4-bit counter) can be preserved by encapsulating them in separate Verilog modules whose internal detail is described behaviorally or by a mixture of behavioral and structural models.

Prototyping a circuit with an FPGA requires synthesizing a Verilog description to produce a bit stream that can be downloaded to configure the internal resources (e.g., CLBS of a Xilinx FPGA) and connectivity of the FPGA. Three details require attention: (1) The pins of the prototyping board are connected to the pins of the FPGA, and the hardware implementation of the synthesized circuit requires that its input and output signals be associated with the pins of the prototyping board (this association is made using the synthesis tool provided by the vendor of the FPGA (such tools are available free)), (2) FPGA prototyping boards have a clock generator, but it will be necessary, in some cases, to implement a clock divider (in Verilog) to obtain an internal clock whose frequency is suitable for the experiment, and (3) inputs to an FPGA-based circuit can be made using switches and pushbuttons located on the prototyping board, but it might be necessary to implement a pulser circuit in software to control and observe the activity of a counter or a state machine (see the supplement to Experiment 1).

Supplement to Experiment 1 (Section 11.2)

The functionality of the counters specified in Experiment 1 can be described in Verilog and synthesized for implementation in an FPGA. Note that the circuit shown in Fig. 11.3 uses a push-button pulser or a clock to cause the count to increment in a circuit built with standard ICs. A software pulser circuit can be developed to work with a switch on the prototyping board of an FPGA so that the operation of the counters can be verified by visual inspection.

The software pulser has the ASM chart shown in Fig. 11.25, where the external input (*Pushed*) is obtained from a mechanical switch or pushbutton. This circuit asserts *Start* for one cycle of the clock and then waits for the switch to be opened (or the pushbutton to be released) to ensure that each action of the switch or pushbutton will produce only one pulse of *Start*. If the counter, or a state machine, is in the reset state (*S_idle*) when the switch is closed, the pulse will launch the activity of the counter or state machine. It will be necessary to open the switch (or release the pushbutton) before *Start* can be reasserted. Using the software pulser will allow each value of the count to be observed. If necessary, a simple synchronizer circuit can be used with *Pushed*.

Supplement to Experiment 2 (Section 11.3)

The various logic gates and their propagation delays were introduced in the hardware experiment. In Section 3.10, a simple circuit with gate delays was investigated. As an introduction

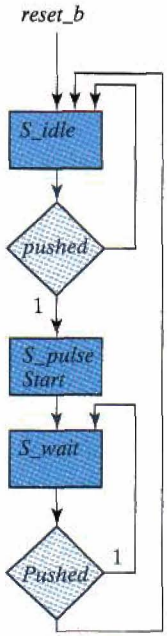


FIGURE 11.25
Pulser circuit for FPGA implementation of Experiment 1

to the laboratory Verilog program, compile the circuit described in HDL Example 3.3 and then run the simulator to verify the waveforms shown in Fig. 3.38.

Assign the following delays to the exclusive-OR circuit shown in Fig. 3.32(a): 10 ns for an inverter, 20 ns for an AND gate, and 30 ns for an OR gate. The input of the circuit goes from $xy = 00$ to $xy = 01$.

(a) Determine the signals at the output of each gate from $t = 0$ to $t = 50$ ns.

(b) Write the HDL description of the circuit including the delays.

(c) Write a stimulus module (similar to HDL Example 3.3) and simulate the circuit to verify the answer in part (a).

(d) Implement the circuit with an FPGA and test its operation.

Supplement to Experiment 4 (Section 11.5)

The operation of a combinational circuit is verified by checking the output and comparing it with the truth table for the circuit. HDL Example 4.10 (Section 4.12) demonstrates the procedure for obtaining the truth table of a combinational circuit by simulating it.

(a) In order to get acquainted with this procedure, compile and simulate HDL Example 4.10 and check the output truth table.

(b) In Experiment 4, you designed a majority logic circuit. Write the HDL gate-level description of the majority logic circuit together with a stimulus for displaying the truth table. Compile and simulate the circuit and check the output response.

(c) Implement the majority logic circuit units an FPGA and test its operation.

Supplement to Experiment 5 (Section 11.6)

This experiment deals with code conversion. A BCD-to-excess-3 converter was designed in Section 4.4. Use the result of the design to check it with an HDL simulator.

(a) Write an HDL gate-level description of the circuit shown in Fig. 4.4.

(b) Write a dataflow description using the Boolean expressions listed in Fig. 4.3.

(c) Write an HDL behavioral description of a BCD-to-excess-3 converter.

(d) Write a test bench to simulate and test the BCD-to-excess-3 converter circuit in order to verify the truth table. Check all three circuits.

(e) Implement the behavioral description with an FPGA and test the operation of the circuit.

Supplement to Experiment 7 (Section 11.8)

A four-bit adder–subtractor is developed in this experiment. An adder–subtractor circuit is also developed in Section 4.5.

(a) Write the HDL behavioral description of the 7483 four-bit adder.

(b) Write a behavioral description of the adder–subtractor circuit shown in Fig. 11.11.

(c) Write the HDL hierarchical description of the four-bit adder–subtractor shown in Fig. 4.13 (including V). This can be done by instantiating a modified version of the four-bit adder described in HDL Example 4.2 (Section 4.12).

(d) Write an HDL test bench to simulate and test the circuits of part (c). Check and verify the values that cause an overflow with $V = 1$.

(e) Implement the circuit of part (c) with an FPGA and test its operation.

Supplement to Experiment 8 (Section 11.9)

The edge-triggered *D* flip-flop 7474 is shown in Fig. 11.13. The flip-flop has asynchronous preset and clear inputs.

(a) Write an HDL behavioral description of the 7474 *D* flip-flop, using only the *Q* output. (Note that when *Preset* = 0, *Q* goes to 1, and when *Preset* = 1 and *Clear* = 0, *Q* goes to 0. Thus, *Preset* takes precedence over *Clear*.)

(b) Write an HDL behavioral description of the 7474 *D* flip-flop, using both outputs. Label the second output *Q_not*, and note that this is not always the complement of Q. (When *Preset* = *Clear* = 0, both *Q* and *Q_not* go to 1.)

Supplement to Experiment 9 (Section 11.10)

In this hardware experiment, you are asked to design and test a sequential circuit whose state diagram is given by Fig. 11.14. This is a Mealy model sequential circuit similar to the one described in HDL Example 5.5 (Section 5.6).

(a) Write the HDL description of the state diagram of Fig. 11.14.

(b) Write the HDL structural description of the sequential circuit obtained from the design. (This is similar to HDL Example 5.7 in Section 5.6.)

(c) Figure 11.24(c) (Section 11.18) shows a control state diagram. Write the HDL description of the state diagram, using the one-hot binary assignment (see Table 5.9 in Section 5.7) and four outputs—T_0, T_1, T_2, and T_3, where T_0 asserts if the state is 4'b0001, T_1 asserts if the state is 4'b0010, etc.

(d) Write a behavioral model of the datapath unit, and verify that the interconnected control unit and datapath unit operate correctly.

(e) Implement the integrated circuit with an FPGA and test its operation.

Supplement to Experiment 10 (Section 11.11)

The synchronous counter with parallel load IC type 74161 is shown in Fig. 11.15. This circuit is similar to the one described in HDL Example 6.3 (Section 6.6), with two exceptions: The load input is enabled when equal to 0, and there are two inputs (P and T) that control the count. Write the HDL description of the 74161 IC. Implement the counter with an FPGA and test its operation.

Supplement to Experiment 11 (Section 11.12)

A bidirectional shift register with parallel load is designed in this experiment by using the 74195 and 74157 IC types.

(a) Write the HDL description of the 74195 shift register. Assume that inputs J and $\overline{K}$ are connected together to form the serial input.

(b) Write the HDL description of the 74157 multiplexer.

(c) Obtain the HDL description of the four-bit bidirectional shift register that has been designed in this experiment. (1) Write the structural description by instantiating the two ICs and specifying their interconnection, and (2) write the behavioral description of the circuit, using the function table that is derived in this design experiment.

(d) Implement the circuit with an FPGA and test its operation.

Supplement to Experiment 13 (Section 11.14)

This experiment investigates the operation of a random-access memory (RAM). The way a memory is described in HDL is explained in Section 7.2 in conjunction with HDL Example 7.1.

(a) Write the HDL description of IC type 74189 RAM, shown in Fig. 11.18.

(b) Test the operation of the memory by writing a stimulus program that stores binary 3 in address 0 and binary 1 in address 14. Then read the stored numbers from the two addresses to check whether the numbers were stored correctly.

(c) Implement the RAM with an FPGA and test its operation.

Supplement to Experiment 14 (Section 11.15)

(a) Write the HDL behavioral description of the 74194 bidirectional shift register with parallel load shown in Fig. 11.19.

(b) Implement the shift register with an FPGA and test its operation.

Supplement to Experiment 16 (Section 11.17)

A parallel adder with an accumulator register and a memory unit is shown in the block diagram of Fig. 11.23. Write the structural description of the circuit specified by the block diagram. The HDL structural description of this circuit can be obtained by instantiating the various components. An example of a structural description of a design can be found in HDL Example 8.4 in Section 8.6. First, it is necessary to write the behavioral description of each component. Use counter 74161 instead of 7493, and substitute the *D* flip-flop 7474 instead of the *JK* flip-flop 7476. The block diagram of the various components can be found from the list in Table 11.1. Write a test bench for each model, and then write a test bench to verify the entire design. Implement the circuit with an FPGA and test its operation.

Supplement to Experiment 17 (Section 11.18)

The block diagram of a four-bit binary multiplier is shown in Fig. 11.24. The multiplier can be described in one of two ways: (1) by using the register transfer level statements listed in part (b) of the figure or (2) by using the block diagram shown in part (a) of the figure. The description of the multiplier in terms of the register transfer level (RTL) format is carried out in HDL Example 8.5 (Section 8.7).

(a) Use the integrated circuit components specified in the block diagram to write the HDL structural description of the binary multiplier. The structural description is obtained by using the module description of each component and then instantiating all the components to show how they are interconnected. (See Section 8.5 for an example.) The HDL descriptions of the components may be available from the solutions to previous experiments. The 7483 is described with a solution to Experiment 7(a), the 7474 with Experiment 8(a), the 74161 with Experiment 10, and the 74194 with Experiment 14. The description of the control is available from a solution to Experiment 9(c). Be sure to verify each structural unit before attempting to verify the multiplier.

(b) Implement the binary multiplier with an FPGA. Use the pulser described in the supplement to Experiment 1.