100

NAME: Don Dvorak

WEEKLY TEST

1. Four BCD counters are cascaded. A 1.2 MHz signal is applied to the input counter. The output of the fourth counter is:

   $\div 10 \quad \div 10 \quad \div 10 \quad \div 10$

   $\frac{1.2^{+6}}{1 \times 10^4} = 120$

   [✓ A.] 120 Hz
   B. 120 KHz
   C. 300 KHz
   D. 4.8 MHz

2. A four bit binary counter contains the number 0100. Nine input pulses occur. The new counter state is:

   4+9 = 13

   4+9 = 13

   A. 0010
   B. 1001
   C. 1011
   [✓ D.] 1101

3. A four bit binary counter contains the number 1010. Nine input pulses are applied. The new counter state is:

   10+9 = 19
   −16
   03

   A. 0001
   [✓ B.] 0011
   C. 1100
   D. 1111

4. A binary counter made up of 3 JK flip-flops will divide an input frequency by

   2^3

   A. 5
   [✓ B.] 8
   C. 16
   D. 32

5. A four bit down counter is in the state 0110 state. Fourteen pulses occur. What is the new output state?

   14

   6 5 4 3 2 1 0 15 14 13 12 11 10 9 8

   A. 0110 [✓ C.] 1000
   B. 0100 D. 1110

6. A BCD counter is cascaded with a 2 flip-flop binary ripple counter. The overall frequency division ratio is:

$\div 10, \div 2, \div 2 = \div 40$

   A. 20
   B. 30
   [✓ C.] 40
   D. 80

7. The two types of MOS shift registers are Static and Dynamic.

8. What determines the frequency of oscillation of most clock circuits?

   A. Crystal
   B. Power supply voltage
   [✓ C.] RC time constant

9. What is the output frequency of the circuit shown on the board?

$\div 2, \div 10, \div 2, \div 2$

$\div 80$

$\frac{500^{+3}}{80} =$ 6.25 Khz

NAME Don Dvorak

97

DATE 2-11-82

DIGITAL TECHNIQUES

UNIT 1 EXAM

1. The two basic types of electronic circuits and signals are Analog and Digital.

2. Tell whether the below signals are analog or digital.

$\frac{1}{2^0} = \frac{1}{1}$

$\frac{1}{2^1} = \frac{1}{2} = .5$

A. Dice digital
B. Typewriter digital
C. Slide Rule analog,
D. Weather Vane analog
E. Camera Shutter digital

3. The most widespread use of digital techniques is in Computers

4. Convert the following binary to decimal.

A. 10010.11 18.75
B. 1110110010.0101 946.3125
C. 11100.1001 28.5625
D. 1011111100.011 764.375

5. Convert the following decimal numbers to binary.

A. 301 100101101 9 bits
B. 32.4375 100000.0111
C. 126 1111110 7 bits

6. Convert the following decimal numbers to BCD.

A. 30.97 0011 0000 . 1001 0111
B. 2486 0010 0100 1000 0110
C. 741 0111 0100 0001
D. 1032 0001 0000 0011 0010

7. Convert the following 8421 BCD to decimal numbers.

A. 1000 0110 0010 0101 8625
B. 0001 1001 0111 0100 1974
C. 0010 0110 0101.0110 0001 265.61
D. 1000 0011 0101 835

.5
.0625
.5625

.25
.125
.375

2
16
32
128
256
512
946.25
.0625
.3125

.25
.0625
.3125

.4375
.25
.1872
.125
.0622
.03125

126
64 1
62
32 1
30
16 1
14
8 1
6
4 1
2
2 1
0

301
256
45
32
13
8
5
4
1

1
0
0
1
0
1
1
0
1

.4375
.25
.1875
.125
.0625

8. What is the ASCII code for the following?

		MSD LSD
A.	DLE	0010000
B.	=	0111101
C.	NAK	0010101
D.	8	0111000
E.	U	1010101

9. Serial data transmission is slower than parallel data transfers.

10. How many digits will there be for converting 932,000,000 to binary.

~~28~~ 29

17	65,536
18	131,072
19	262,144
20	524,288
21	1,048,576
22	2,097,152
22	4,194,304
23	16,772,216
24	33,554,432
25	67,108,864
26	134,217,728
27	268
28	536
29	1,072,

134

WEEKLY TEST- DIGITAL TECHNIQUES

NAME Don Dvorak

100

1. The most popular and widely used logic type is
   [✓ A. TTL]
   B. CMOS
   C. MOS
   D. ECL

2. The fastest logic type is
   A. TTL
   B. CMOS
   C. MOS
   [✓ D. ECL]

3. The logic type with the lowest power consumption is
   A. TTL
   [✓ B. CMOS]
   C. MOS
   D. ECL

4. The logic type with the best noise immunity is
   A. TTL
   [✓ B. CMOS]
   C. MOS
   D. ECL

5. Three CMOS logic gates with an average propagation delay of 70 nano-seconds are cascaded. The output stage will change how many nanosecond after a change in input state?
   A. 23.3
   B. 70
   C. 140
   [✓ D. 210]

6. The power dissipated by a logic gate is proportional to its
   A. complexity
   B. noise immunity
   [✓ C. speed]
   D. fan out

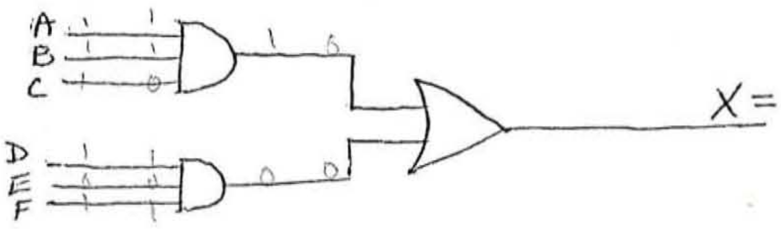
7. Refer to Figure 4-32. Which type of logic gate has the best speed / power trade off?
   A. Standard TTL
   [✓ B. Schottky TTL (low power)]
   C. CMOS
   D. ECL

8. What type of logic circuit would you select for a battery operated uni in a noisy environment?
   A. TTL
   B. ECL
   [✓ C. CMOS]
   D. MOS

9. The propogation delay of a typical logic gate is usually measured in
   [✓ A. nanoseconds]
   B. microseconds
   C. milliseconds
   D. seconds

10. Which of the following is not a use for Boolean algebra
A. Analyze logic circuits
[✓ B.] Solve binary number problems in logic circuits
C. Design logic circuits
D. Minimize logic circuits

11. What is the Boolean equation for the following logic?

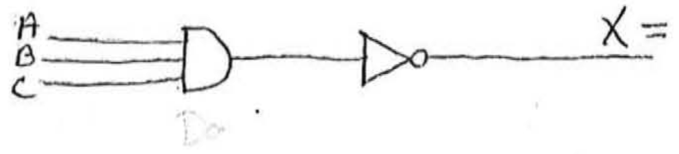


X = ABC + DEF

12. What does x= if A=1,B=1,C=1,D=1,E=0,F=1? 1

13. What does x= if A=1,B=1,C=0,D=1,E=0,F=1? 0

14. Give the Boolean equation for the following logic



$\overline{X}$ = ABC

15. What type of logic gate is drawn in question 14?

3 input Nand

$\overline{A} + \overline{B} + \overline{C} = X$

NAME Don Dvorak

EXAM 2

100

Convert the following binary numbers to decimal.

A. 0.0101 .3125₁₀
B. 1101101101 877₁₀

2. Convert the following decimal numbers to binary (carry fractions to 4 places).

A. 256.75 100000000.11
B. 4092.90625 1000000000000.1110 .9375

3. Convert the following numbers as required. Carry fractions to a maximum 6 places.

A.	91 ₁₀	133 ₈	001011011 ₂	
B.	545.375 ₁₀	1041.3 ₈	1000100001.011 ₂	
C.	5.375 ₁₀	5.3 ₈	101.011 ₂	
D.	AB7 ₁₆	2743 ₁₀	5267 ₈	101010110111 ₂
E.	3F4 ₁₆	1012 ₁₀	1764 ₈	1111110100 ₂

001 111 110 100
0011 1111 0100
3 F 4

4. For the drawings on the board, give the value of

A. X = +
B. Y = +
C. Z = +

5. The transistor drawn on the board is in:

A. cut-off
[✓ B.] conducting in the linear region
C. saturated

6. A logic circuit has 5 inputs. How many possible input combination can it have?

= $2^5 = 32$

A. 2
B. 4
C. 5
D. 16
[✓ E. 32]

1012
512
500
256
244
128
116
64
52
32
20
16
4

1021
512
509
256
253
128
125
64
61
32
29
16

13
8
5
4

A B 7
1010 1011 0111

101 010 110 111
5 2 6 7

2743
2048
659
512
147
128
17
16

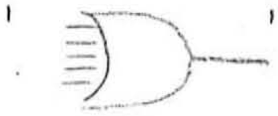
545
512
33
32
1

512
32
1
545

7. A home intrusion alarm system is designed to sound a bell if any one of the following conditions occur: front or back door opens, any window opens, garage door opens. What logic function is implied? or

A. AND
[✓ B.] OR
C. NAND
D. NOR
E. NOT

opens = 1
alarm = 1



8. Fill in the output for the following truth table for a diode AND gate.

INPUTS		OUTPUT
A	B	C
ØV	ØV	∅v
ØV	+5V	∅v
+5V	ØV	∅v
+5V	+5V	+5 V

96

NAME Don Dvorak

WEEKLY EXAM

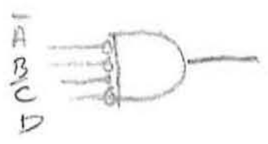
1) A 1 of 16 decoder has how many inputs?

A) 1
[✓ B)] 4
C) 8
D) 16

2) What binary number does the decoder gate on the board recognize? (A is the LSB)

MSB LSB

[✓ A)] 1010
[✓ B)] 0101
C) 0000
D) 1111



LSB
1010

3) A decoder gate to recognize the binary equivalent of the decimal number 289 requires how many inputs?

A) 3
B) 6
[✓ C)] 9
D) 12

9 512
[illegible]

- 4) Which of the following functions cannot be performed by an MSI digital multiplexer?

[✓ A)] Parity generator
B) Parallel to serial converter
C) Boolean function generator
D) Serial word generator

5) An LSI circuit that implements multiple, sum-of-products logic equation is called a:

A) Decoder
B) Multiplexer
C) ROM
[✓ D)] PLA

6) What is the total bit capacity fo a ROM with 6 input bits and 8 output bits?

A) 48
B) 64
C) 256
[✓ D)] 512

$2^6 = 64$
$\times 8$
512

7) In order to get a number 5 to light up on your decimal display, what must your output to your segements be: (A one or zero)

a = 0	c = 0	e = 1	g = 0
b = 1	d = 0	f = 0	

D. Dvorak

LSB

E D C B A

A 1 0 0 1 1

16 2 1

8) What is the decimal equivalent of the state being decoded by the circuit on the board?

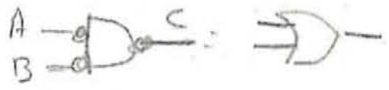
19

9) The basic decoder is a(n) and gate and the basic encoder is a(n) ______ gate. pos NAND/neg NOR

10) Serial to parallel conversion is easily done by a Demultiplexer circuit.

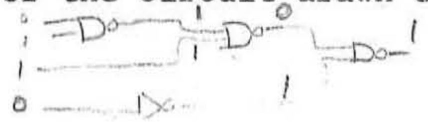
NAME Don Dvorak

DIGITAL TECHNIQUES - WEEKLY EXAM

Write a truth table for a negative AND gate. 

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

2. For the circuit drawn on the board, what is X equal to?

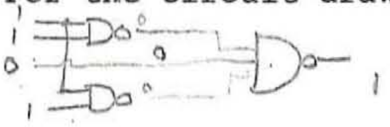


1

3. Digital signal levels of binary 0 = -.7 and binary 1 = -1.7 represent which type of logic?

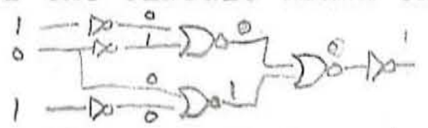
A. positive
[✓ B.] negative

4. For the circuit drawn on the board A=1, B=1, C=0 and D=1; X will be



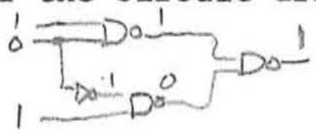
1

5. For the circuit drawn on the board A=1, B=0, and C=1; Y will be



1

6. For the circuit drawn on the board A=1, B=0 and C=1; Z will be



1

7. The complement output of a flip-flop is low. What binary start is stored in the flip-flop?

A. binary 0
[✓ B.] binary 1

HEATHKIT
CONTINUING
EDUCATION

Individual Learning Program In DIGITAL TECHNIQUES

7
SEQUENTIAL
LOGIC CIRCUITS

UNIT 7

SEQUENTIAL LOGIC CIRCUITS: COUNTERS AND SHIFT REGISTERS

INTRODUCTION

In this unit you are going to learn about sequential logic circuits. These are logic circuits that are used for a variety of timing, sequencing and storage functions. The key characteristic of sequential logic circuits is memory. Sequential logic circuits are capable of storing binary data. The output of a sequential logic circuit is a function not only of the various input states applied to the circuit but also the result of previous operations which are stored in the circuit itself.

The main circuit elements of a sequential circuit are flip-flops. These flip-flops store binary data and their states are changed by the logic input signals in accordance with the current information stored in them. The sequential logic operations are generally sequenced by a periodic logic signal known as a clock. The clock is an oscillator that generates rectangular pulses at a fixed frequency.

As with combinational logic circuits, there can exist almost an infinite variety of sequential logic circuits. However, in practice only a few special types seem to regularly reoccur. The two most commonly used sequential circuits are counters and shift registers. Because these two types of sequential circuits are the most widely used, we will emphasize their operation and application in this unit. In addition, clock circuits and special sequential circuit components such as one shot multivibrators and the delay lines will also be considered. Like combinational logic circuits, most sequential circuits are implemented with integrated circuits. In fact, most of the commonly used sequential circuits are available as a single ready to use MSI logic package. We will concentrate on these popular devices in this unit.

UNIT OBJECTIVES

When you complete this unit you will be able to:

1. Name the two most widely used types of sequential logic circuits.
2. Explain the operation of both binary and BCD counters.
3. Determine the maximum count capability of a binary or BCD counter given the number of flip-flops it contains and a logic diagram.
4. Determine the count sequence of a counter from a logic diagram and draw the circuit waveforms.
5. Explain the operation of a shift register.
6. List four applications for shift registers.
7. Explain the purpose of the clock signal and show a method of developing it.
8. Explain the operation of a one shot and list several applications.

UNIT ACTIVITY GUIDE

	Completion Time
☐ Play audio record 6, side 1: Sequential Logic Circuits	______
☐ Read section Binary Counters	______
☐ Answer Self Test Review Questions 1—15	______
☐ Perform Experiment 12	______
☐ Read section BCD Counters	______

Heathkit
ZENITH Educational Systems

UNIT 7

SEQUENTIAL LOGIC CIRCUITS: COUNTERS, SHIFT REGISTERS AND CLOCKS

CONTENTS

- ☐ Answer Self Test Review Questions 16—24 ________
- ☐ Perform Experiment 13 ________
- ☐ Read section Special Counters ________
- ☐ Answer Self Test Review Questions 25 and 26 ________
- ☐ Perform Experiment 14 ________
- ☐ Read sections Shift Register Operation and Bipolar Shift Registers ________
- ☐ Answer Self Test Review Questions 27—32 ________
- ☐ Perform Experiment 15 ________
- ☐ Read section Shift Register Applications ________
- ☐ Answer Self Test Review Questions 33—38 ________
- ☐ Perform Experiment 16 ________
- ☐ Read section MOS Shift Registers ________
- ☐ Answer Self Test Review Questions 39—44 ________
- ☐ Read section Clocks and One Shots ________
- ☐ Answer Self Test Review Questions 45—50 ________
- ☐ Perform Experiment 17 ________
- ☐ Complete Unit Examination ________

COUNTERS

A binary counter is a sequential logic circuit made up of flip-flops that is used to count the number of binary pulses applied to it. The pulses or logic level transitions to be counted are applied to the counter input. These pulses cause the flip-flops in the counter to change state in such a way that the binary number stored in the flip-flops is representative of the number of input pulses that have occurred. By observing the flip-flop outputs you can determine how many pulses were applied to the input.

There are several different types of counters used in digital circuits. The most commonly used is the binary counter. This type of counter counts in the standard pure binary code. BCD counters which count in the standard 8421 BCD code are also widely used. In addition, counters can be developed to count in any of the special binary or BCD codes in common use. Both up and down counters are available.

BINARY COUNTERS

A binary counter is a sequential logic circuit that uses the standard pure binary code. Such a counter is made up by cascading JK flip-flops as shown in Figure 7-1. The normal output of one flip-flop is connected to the toggle (T) input of the next flip-flop. The JK inputs on each flip-flop are open or high. The input pulses to be counted are applied to the toggle input of the A flip-flop.

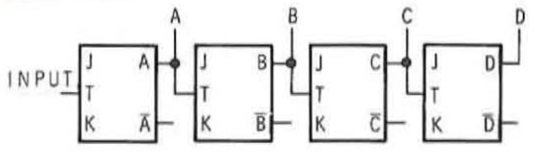


Figure 7-1
Four bit binary counter.

To see how this binary counter operates, remember that a JK flip-flop toggles or changes state each time a trailing edge transition occurs on its T input. The flip-flops will change state when the normal output of the previous flip-flop switches from binary 1 to binary 0. If we assume that the counter is initially reset, the normal outputs of all the flip-flops will be binary 0. When the first input pulse occurs, the A flip-flop will become set. The binary number stored in the flip-flops indicates the number of input pulses that have occurred. To read the number stored in the counter you simply observe the normal outputs of the flip-flops. The A flip-flop is the least significant bit of the word. Therefore, the four bit number stored in the counter is designated DCBA. After the first input pulse, the counter state is 0001. This indicates that one input pulse has occurred.

When the second input pulse occurs the A flip-flop toggles and this time becomes reset. As it resets its normal output switches from binary 1 to binary 0. This causes the B flip-flop to become set. Observing the new output state, you see that it is 0010 or the binary equivalent of the decimal number 2. Two input pulses have occurred.

When the third input pulse occurs the A flip-flop will again set. The normal output switches from binary 0 to binary 1. This transition is ignored by the T input of the B flip-flop. The number stored in the counter at this time then is 0011 or the number 3 indicating that three input pulses have occurred.

When the fourth input pulse occurs, the A flip-flop is reset. Its normal output switches from binary 1 to binary 0 thereby toggling the B flip-flop. This causes the B flip-flop to reset. As it does, its normal output switches from binary 1 to binary 0 causing the C flip-flop to become set. The number now in the counter is 0100 or a decimal 4. This process continues as the input pulses occur. The count sequence is the standard 4 bit binary code as indicated in Figure 7-2.

An important point to consider is the action of the circuit when the number stored in the counter is 1111. This is the maximum value of a four bit number and the maximum count capacity of the circuit. When the next input pulse is applied, all flip-flops will change state. As the A flip-flop resets, the B flip-flop resets. As the B flip-flop resets it, in turn, resets the C flip-flop. As the C flip-flop resets, it toggles the D flip-flop which is also reset to zero. The result is that the contents of the counter becomes 0000. As you can see from Figure 7-2, when the maximum content of the counter is reached it simply recycles and starts its count again.

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

RECYCLE

Figure 7-2
Count sequence of four bit binary counter.

The complete operation of the four bit binary counter is illustrated by the input and output waveforms in Figure 7-3. The upper waveform is a

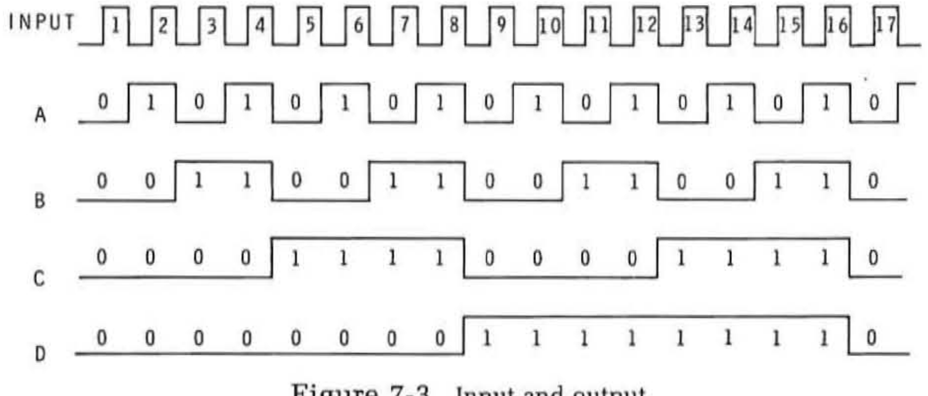


Figure 7-3 Input and output waveforms of a 4 bit binary counter.

series of input pulses to be counted. Here they are shown as a periodic binary waveform but of course it is not necessary for the input signal to be of a constant frequency or have equally spaced input pulses. The waveforms also show the normal output of each flip-flop.

In observing the waveforms in Figure 7-3, you should note several important things. First, all the flip-flops toggle (change state) on the trailing edge or the binary 1 to binary 0 transition of the previous flip-flop. With this in mind you can readily trace the output of the first (A) flip-flop by simply observing when the trailing edges of the input occur.

The output of the B flip-flop is a function of its input which is the output of the A flip-flop. Note that its state change occurs on the trailing edge of the A output. The same is true of the C and D flip-flops. The binary code after each input pulse is indicated on the waveforms. Of course, this corresponds to the binary count sequence in Figure 7-2.

Frequency Divider. Another important fact that is clear from the waveforms in Figure 7-3 is that the binary counter is also a frequency divider. The output of each flip-flop is one half the frequency of its input. If the input is a 100 kHz square wave, the outputs of the flip-flops are:

A — 50 kHz
B — 25 kHz
C — 12.5 kHz
D — 6.25 kHz

The output of a pure binary counter is always some sub-multiple of two. The four bit counter divides the input by 16, (100 kHz ÷ 16 = 6.25 kHz).

Maximum Count. The maximum count capability of a binary counter is a function of the number of flip-flops in the counter. The maximum number that can be contained in a binary counter before it recycles is determined in the same way that we can determine the maximum binary number that can be represented by a word with a specific number of bits. The formula below expresses the relationship between the number of flip-flops in a counter and its maximum count capability.

$$N = 2^n - 1$$

Here N is the maximum number that occurs prior to the counter recycling. The number of flip-flops is designated by n. For example, the maximum number that can be contained in a counter using four flip-flops is

$$N = 2^4 - 1 = 16 - 1 = 15 \text{ (binary 1111)}$$

Another example is a binary counter with nine flip-flops. A maximum count capability here then is

$$N = 2^9 - 1 = 512 - 1 = 511 \text{ (binary 111111111)}$$

To determine the number of flip-flops required to implement a counter with a known or required count capability, use the formula given below.

$$n = 3.32 \log_{10} N$$

For example, if you wish to implement a binary counter capable of counting to 100, you could determine the number of flip-flops as follows:

$$n = 3.32 \log_{10} 100 = 3.32\ (2) = 6.64$$
$$n = 7$$

Since there is no such thing as a fractional part of a flip-flop, the next higher whole number value is used. A counter with 7 flip-flops has a maximum count capability of

$$2^7 - 1 = 127.$$

When the binary counter is used as a frequency divider, the factor by which the counter divides is a function of the number of flip-flops used. To determine the divide ratio, the expression below is used.

$$N = 2^n$$

For example, if the counter contained six flip-flops it would divide an input signal by

$$N = 2^6 = 64$$

What this means is that the output of the sixth flip-flop will be $1/64$ the frequency of the input signal applied to the first flip-flop. With a binary counter, the frequency division ratio is always some power of 2. As you will see later, it is possible to implement frequency dividers that can divide the frequency by any integer value.

Down Counters. The binary counter just described is referred to as an up-counter. Each time that an input pulse occurs, the binary number in the counter is increased by one. We say that the input pulses increment the counter. It is also possible to produce a down counter where the input pulses cause the binary number in the counter to decrease by one. The input pulses are said to decrement the counter.

A four bit binary down counter is shown in Figure 7-4. It is practically identical to the up counter described earlier. The only difference is that

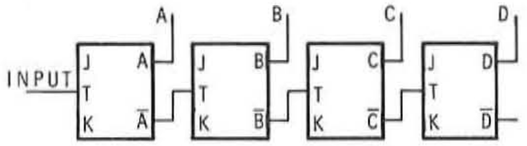


Figure 7-4
Four bit binary down counter.

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

RECYCLE

Figure 7-5 Count sequence for four bit down counter.

the complement output rather than the normal output of each flip-flop is connected to the toggle input of the next flip-flop in sequence. This causes the count sequence to be the exact reverse of the up counter. The count sequence is illustrated in Figure 7-5. The waveforms associated with this counter are shown in Figure 7-6.

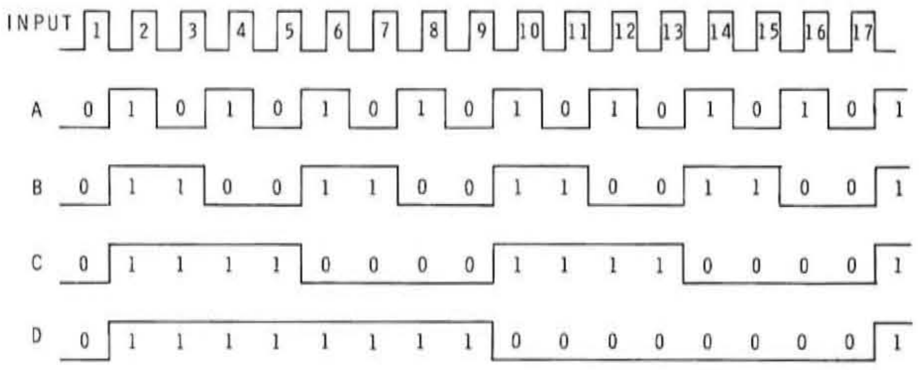


Figure 7-6
Input and output waveforms
of a four bit binary down counter.

In analyzing the operation of this counter, keep in mind that we still determine the contents of the counter by observing the normal outputs of the flip-flops as we did with the up-counter. Assuming the counter is initially reset and its contents is 0000, the application of an input pulse will cause all flip-flops to become set. With the A flip-flop reset, its complement output is high. When the first input pulse is applied, the A flip-flop will set. As it does its complement output will switch from binary 1 to binary 0 thereby toggling the B flip-flop. The B flip-flop becomes set and its complement output also switches from binary 1 to binary 0. This causes the C flip-flop to set. In the same way the complement output of the C flip-flop switches from high to low thereby setting the D flip-flop. The counter recycles from 0000 to 1111.

When the next input pulse arrives, the A flip-flop will again be complemented. It will reset. As it resets the complement output will switch from binary 0 to binary 1. The B flip-flop ignores this transition. No further state changes take place. The content of the counter then is 1110. As you can see this input pulse causes the counter to be decremented from 15 to 14.

Applying another input pulse again complements the A flip-flop. It now sets. As it sets, its complement output switches from binary 1 to binary 0. This causes the B flip-flop to reset. As it resets, the complement output switches from binary 0 to binary 1. The C flip-flop ignores this transition. The new counter contents then is 1101 or 13. Again the input pulse caused the counter to be decremented by one. By using the table in Figure 7-5 and the waveforms in Figure 7-6, you can trace the complete operation of the 4 bit binary down counter.

Up-Down Counter. The up counting and down counting capabilities can be combined within a single counter as illustrated in Figure 7-7. AND and OR gates are used to couple the flip-flops. The normal output of each flip-flop is applied to gate 1. The complement output of each flip-flop is connected to gate 2. These gates determine whether the normal or complement signals toggle the next flip-flop in sequence. The count control line determines whether the counter counts up or down.

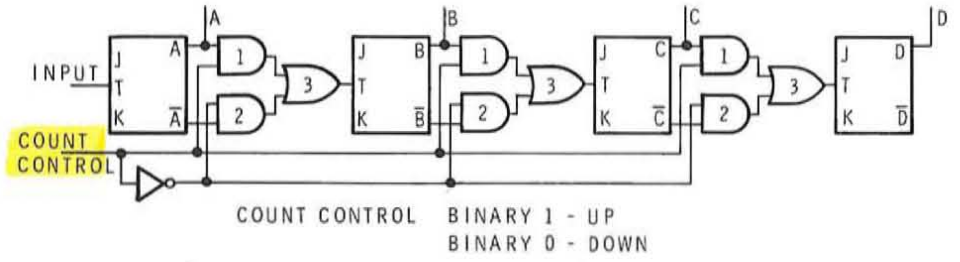


Figure 7-7
Binary Up/Down Counter.

If the count control input is binary 1, all gate 1s are enabled. The normal output of each flip-flop then is coupled through gates 1 and 3 to the T input of the next flip-flop. The counter therefore counts up. During this time, all gate 2s are inhibited.

By making the count control line binary 0, all gate 2s are enabled. The complement output of each flip-flop is coupled through gates 2 and 3 to the next flip-flop in sequence. With this arrangement, the counter counts down.

ripple - asynchronous

Synchronized

Synchronous Counters. The counters that we have discussed so far are known as ripple counters or asynchronous counters. The term ripple is derived from the fact that the flip-flops in the counter are cascaded with the output of one driving the input of the next. As the count pulses are applied to the first or input flip-flop, the count, in effect, ripples through the flip-flops. The term asynchronous comes as a result of the flip-flops not being controlled by a single common clock pulse. Synchronous digital circuits are ones in which all elements are synchronized to a master timing signal known as a clock.

The primary advantage of a ripple counter is its simplicity. Its primary limitation is its counting speed. The counting speed of a binary counter is limited by the propagation delay of the flip-flops in the counter. In a ripple counter, the propagation delay of the flip-flops is additive. Since each flip-flop in the counter is triggered by the preceding circuit, it can take a significant amount of time for an impulse to ripple through all of the flip-flops and change the state of the last flip-flop in the chain. This worse case condition occurs when all flip-flops in the counter change state simultaneously. Referring back to the waveforms for the four bit binary counter in Figure 7-3 you can see that this worse case condition occurs in two places. It occurs when the count changes from 0111 to 1000 and when the count changes from 1111 to 0000. If each flip-flop in the four bit circuit has a propagation delay of 50 nanoseconds, it can take as long as $4 \times 50 = 200$ nanoseconds for the D flip-flop to change state upon the application of an input pulse. Should the input pulses occur at a rate faster than 200 nanoseconds, the binary number stored in the counter will not truly represent the number of input pulses that have occurred. The counter state will lag the input signal. The upper frequency limit (f) of the ripple counter is approximately equal to

$$f = \frac{1}{n\,t} \times 10^9$$

where n is the number of flip-flops in the counter and t is the propagation delay time of a single flip-flop in nanoseconds.

For a flip-flop with a 35 nanosecond propagation delay in a four bit counter, the maximum counting speed is

$$f = \frac{1}{4(35)} \times 10^9 = 7.1428 \text{ MHz}$$

This means that the counter can count at speeds up to about 7 MHz without counting errors. At higher frequencies, the states of the flip-flops cannot keep pace with the rapid occurrence of the input pulses. The counter may actually lag several pulses depending upon the input frequency and the exact propagation delay. Counting errors can occur if the counting is periodically stopped and continued.

The direct solution to this problem, of course, is to use flip-flops with a lower propagation delay. Flip-flops are available to count at high frequencies up to approximately 500 MHz. The propagation delay is very small. Such flip-flops generally employ a non-saturating logic circuit such as ECL to achieve this fast counting rate. Such circuits are expensive and have high power consumption and are undesirable for many applications.

It is possible to reduce the propagation delay effects and increase the counting speed of a binary counter by using a special circuit arrangement. Counters employing this technique are known as synchronous counters.

A synchronous counter is one where all of the flip-flops are triggered simultaneously by a clock pulse or the signal to be counted. Since all flip-flops change state at the same time, the total propagation delay for the circuit is essentially equal to that of a single flip-flop. The propagation delays are not additive and therefore much higher counting speeds can be achieved.

A typical synchronous binary counter is shown in Figure 7-8. Notice that all of the flip-flop T inputs are connected together to a common count input line. This connection is what makes the counter synchronous. All of the flip-flops are synchronized to the input signal to be counted.

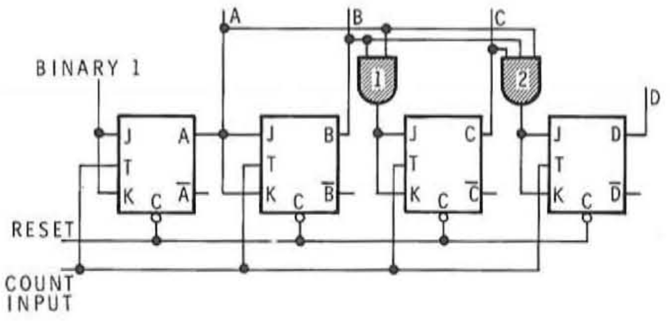


Figure 7-8
A synchronous binary counter.

The operation of the flip-flops is controlled by the states of the JK inputs. As you recall, the J and K inputs can be used as controls for the flip-flop. Up to this point in our discussion of counters we have assumed that the JK inputs were open or connected to a binary 1 level. This essentially enables the flip-flop and permits it to be toggled or complemented each time the trailing edge of an input signal appears on the T input line. If the JK inputs are brought to binary 0, the signal applied to the T input will be ignored. The flip-flop will simply remain in the state to which it was set prior to making the JK input lines low. By using the JK input lines we can then enable or inhibit the toggling of the flip-flops.

In Figure 7-8 the JK inputs on the A flip-flop are connected to binary 1 to enable the flip-flop permanently. Each time a count input signal appears the flip-flop will toggle or change state just as the A flip-flop on the ripple counters discussed earlier operated.

The JK inputs to the B flip-flop are controlled by the normal output of the A flip-flop. This means that the only time that the B flip-flop can change state is when the output of the A flip-flop is binary 1. The JK inputs to the C flip-flop are controlled by the normal outputs of both the A and the B flip-flops. The A and B signals are ANDed together in gate 1 and its output used to control the JK inputs. In order for the C flip-flop to change state, both A and B must be set. The C flip-flop will change state at the first count pulse occurring after A and B are high.

The D flip-flop is controlled by the A, B and C flip-flops. The A, B and C signals are ANDed in gate 2 and the output of gate 2 used to control the JK inputs.

With the circuit arrangement shown in Figure 7-8, the counting sequence is identical to that given in the Table of Figure 7-2. The waveforms of Figure 7-3 are also applicable. In other words, this circuit is still a binary counter but its mode of operation is somewhat different.

The benefit of this binary counter can be best seen by analyzing the state changes in the flip-flop. Assume that the counter contains the number 0111. This means that the A, B and C flip-flops are set. The JK inputs to the B, C and D flip-flops are enabled. This means that upon the occurrence of the next count input pulse, all flip-flops will toggle. When this pulse occurs, flip-flops A, B and C will reset. The D flip-flop will be set. The new number in the counter will be 1000. The important point to note here is that all flip-flops change state simultaneously. The maximum delay between the occurrence of the count input pulse and the change of state of the outputs is only as long as the longest propagation time of the flip-flops in the circuit. All flip-flops of the same type will have approximately the same propagation delay.

Considering this same state change in the binary ripple counter, we can illustrate the effect of the accumulative propagation delay. With the number 0111 stored in the ripple counter of Figure 7-1, the output states will change as follows when an input count pulse is applied. The A flip-flop will change state first. As it does it will toggle the B flip-flop. The B flip-flop must then change state and it in turn will then toggle the C flip-flop. The C flip-flop will change state a short time later and it in turn will set the D flip-flop. Because of the finite propagation delay of each flip-flop the effect of an input count pulse does indeed ripple through the counter and it takes a specific amount of time for the correct binary number to appear in the counter.

In summary then we can say that the advantages of the synchronous counter over the ripple or asynchronous counter are as follows:

1. The synchronous counter is much faster. For the same type of flip-flop, the counting speed of the synchronous counter is significantly higher than that of the ripple counter.
2. All flip-flops in the synchronous counter change states at the same time. As the counter changes from one state to the next, there are no ambiguous states that occur because of the accumulative propagation delays.

Like the ripple counter, the synchronous counter can also be expanded to as many bits as required by the application. Each flip-flop in the counter must be controlled by all previous flip-flops in the counter through an AND gate as indicated. The higher order flip-flops will require AND gates with as many inputs as there are previous flip-flops. Keep in mind that the propagation delays of these gates while small will also have a minor effect on the counting speed of the circuit. Generally, the propagation delay of a gate is significantly lower than that of a flip-flop. This technique can also be extended to down counters as well.

Counter Control Functions. There are several common control functions that are often associated with the use of binary counters. These are reset and preset. Resetting a counter is a process of putting all of the flip-flops in the binary 0 state. In many counter applications it is necessary to clear, reset or zero the counter prior to the start of a counting operation. This process ensures that the counter starts its count sequence with no prior counts stored in the flip-flops. It ensures an accurate count of the input.

Resetting a counter is easily accomplished when JK flip-flops are used. The asynchronous clear input on the flip-flops as you recall are normally used to put the flip-flop into its binary 0 state. By bringing the clear input low, the flip-flop is reset. By connecting all of the asynchronous clear inputs together, all of the flip-flops will be reset to binary 0 simultaneously when a reset pulse is applied. Figure 7-9 shows a binary up-counter with all of the asynchronous clear inputs connected together. In order for the counter to perform normally, the reset line will rest in a high or binary 1 state. To reset the counter we momentarily bring this line to a binary 0. For typical TTL JK flip-flops, a pulse whose duration is 100 nanoseconds or more can be used to reset the counter.

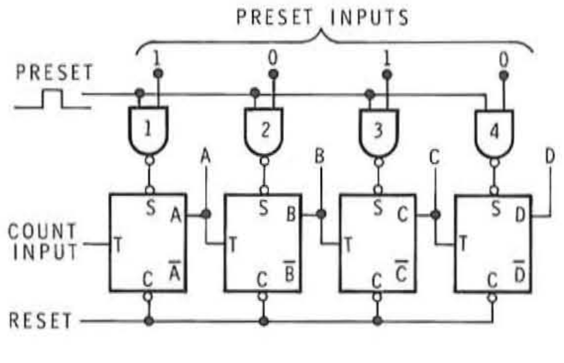


Figure 7-9 A binary counter with reset and preset inputs.

Presetting a counter is a process of loading some binary number into the counter prior to the count sequence. It is sometimes desirable to program a counter to start counting at a particular point. The point at which the counter is to start is determined and then loaded into the counter prior to the start of the count operations.

A counter can be preset by using the asynchronous set input when JK flip-flops are used. When this input on a JK flip-flop is made binary 0, the flip-flop is set. By first clearing the counter and then setting the desired flip-flops, any binary number can be preset into the counter. The circuit shown in Figure 7-9 can be used for this purpose. In order to preset the counter to a given number, the counter is first reset by applying a binary 0 to the reset input line. Next, the desired binary number is applied to the preset inputs. There is one input for each of the flip-flops in the counter. A parallel binary number from any source can be used. When the preset input line is made high, the outputs of the gates to which the parallel input number are applied will cause the asynchronous set lines on the JK flip-flops to assume the correct states to preset the number into the counter. For example, assume that we wish to preset the number 5 into the counter. To do this we would apply the binary number 0101 to the counter as indicated. The counter is then reset, and the preset line is brought high momentarily. The parallel inputs that are binary 0 are applied to gates 2 and 4. These binary 0 inputs hold the outputs of gates 2 and 4 high regardless of the state of the preset input. This keeps the set inputs to the B and D flip-flops high. With these inputs high the flip-flops are not affected. The binary 1 states of the desired input number are applied to gates 1 and 3. When the preset input goes high, the outputs of gates 1 and 3 will go low. This will cause flip-flops A and C to become set. The number 0101 is then stored in the counter.

The method of presetting a counter shown in Figure 7-9 is somewhat awkward in that it requires two operations. First the counter must be reset and then the desired preset number is loaded. It is desirable to have the preset operation take place with a single operation. This can be accomplished by the circuits shown in Figure 7-10. Only one JK flip-flop is shown to simplify the discussion. One of these circuit arrangements would be used on each flip-flop in a counter if presetting were desired. In Figure 7-10A, gates 1 and 2 are connected to the asynchronous set and clear inputs of the JK flip-flop. The desired parallel input (IN) is applied to gate 1. A preset line is tied to both gates 1 and 2. To the input line is applied a binary 0 or binary 1 state which will specify the state of the flip-flop after the preset input is enabled. When the preset input goes high, the JK flip-flop is preset to the desired state. For example, assume that the input is binary 1. When the preset line goes high, the output of gate 1 will go low. This will cause the set(s) input of the JK flip-flop to go low thereby setting the flip-flop. The low on the output of gate 1 will cause the output of gate 2 to remain high. This has no effect on the C input. A low input to gate 1 will reset the flip-flop. With a low input the output of gate 1 will be high. This high output does not affect the S input to the JK flip-flop. It does however enable gate 2. When the preset line goes high, the output of gate 2 will go low. This causes the JK flip-flop to be put in a binary 0 state. The preset operation takes place with only the single operation of applying the preset input pulse. Note that since the asynchronous inputs are used, the presetting operation will over-ride all other flip-flop operations.

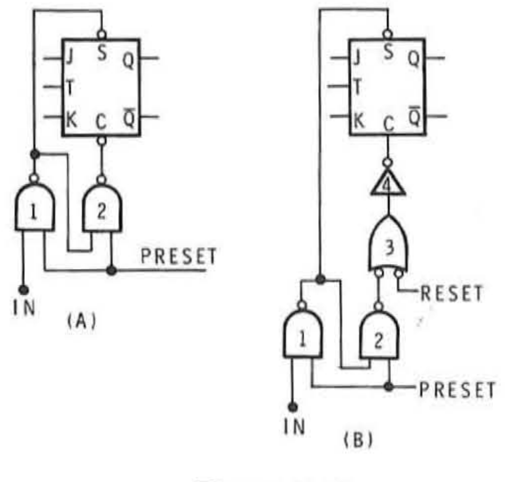


Figure 7-10
Methods of presetting a counter
(A) preset only (B) preset with reset.

Sometimes it is desirable to combine both the reset and preset functions in a counter. This can be accomplished with the circuit shown in Figure 7-10B. Here gates 1 and 2 perform the same basic operation as they do in the circuit of Figure 7-10A. They are used to preset the flip-flops to the desired state. Gate 3 and inverter 4 provide an ORing function to permit the reset function to be incorporated. If the reset input line is brought low, the output of gate 3 will go high and the output of inverter 4 will go low. This will reset the flip-flop. This occurs regardless of the conditions of the preset inputs.

To preset the flip-flop, the desired binary state is applied to the input line on gate 1. When the preset line is brought high, the flip-flop will be put into the states specified by the input. The operation is identical to the circuit in Figure 7-10A. The important point to note about this circuit is that the reset and preset operations are independent of one another. It is not necessary to reset the flip-flop prior to presetting it as was the case in the circuit of Figure 7-9.

An important point to remember is that these circuits for resetting and presetting flip-flops can be applied to any type of counter: synchronous, asynchronous, up-counter, down-counter, binary or BCD counter.

Typical Integrated Circuit Counters

While it is still sometimes necessary and desirable to implement counters with individual IC JK flip-flops, most counter applications can be met with a variety of available MSI integrated circuit counters. All of the most often used binary counters have been implemented in integrated circuit form thereby eliminating the necessity for designing such counters for each application. A variety of counter types and specifications are available. In designing digital equipment, it is desirable to first investigate the types of MSI IC counters available. In most cases you will find one suitable for your application. Only in rare cases where an unusual or peculiar type of counter for unique applications is required will it be necessary for you to design a special counter. However, for such applications, versatile integrated circuit JK flip-flops are available.

In this section we are going to consider one of the most popular and widely used integrated circuit binary counters available.

Figure 7-11 shows the logic diagram of the 74193 TTL MSI IC counter. This is a four bit synchronous counter that can be used for either up or down counting. It also has separate clear or reset inputs as well as the ability to be preset from some external four bit parallel source. In other words, this particular device incorporates all of the features we have considered in a binary counter up to this point.

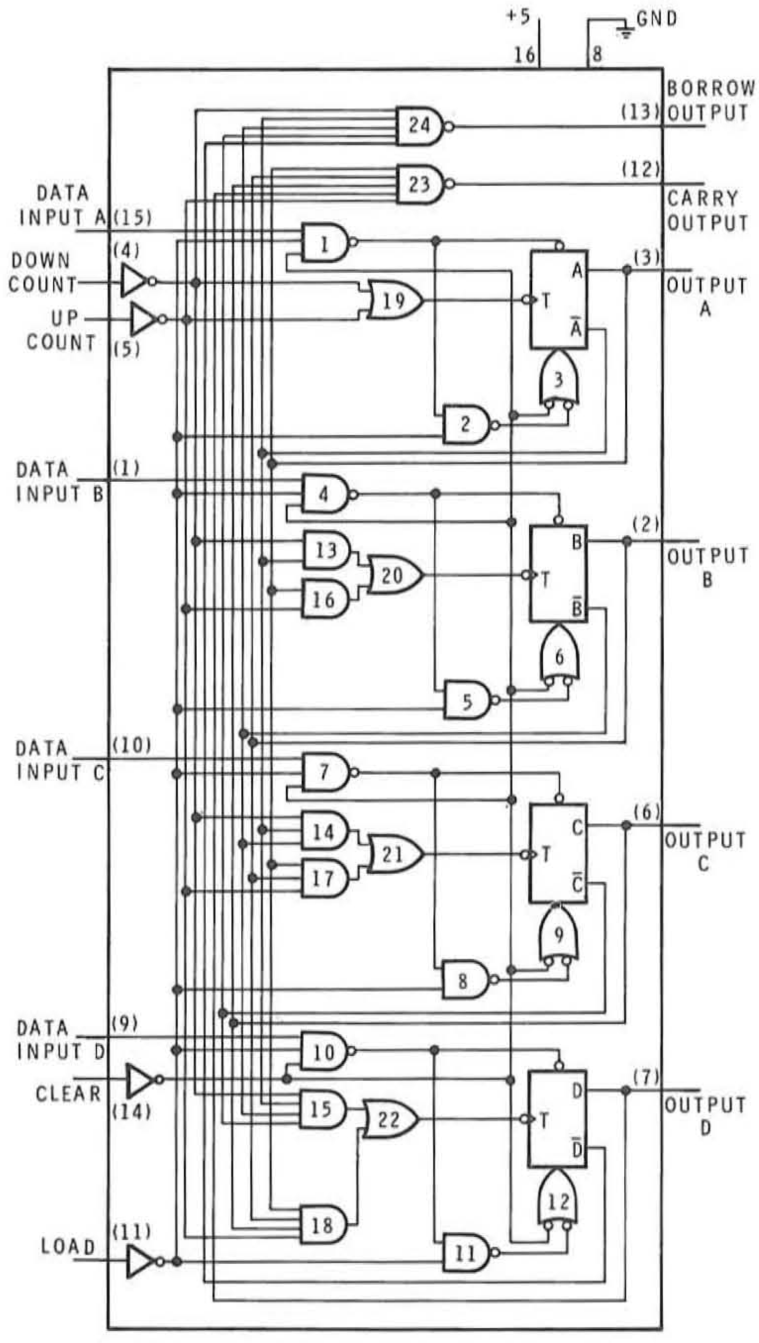


Figure 7-11
Type 74193 TTL MSI binary counter.

As shown in the diagram in Figure 7-11, the counter consists of four JK flip-flops. Gates 1 through 12 make up the logic circuitry used in the reset and preset operations. This circuitry is similar to the reset and preset circuit operations discussed previously. To reset this counter, you apply a high or binary 1 level to the clear input line. This forces all four flip-flops into the binary 0 state. The clear operation is asynchronous and overrides all other counter functions.

The counter can be preset by applying a parallel 4-bit binary number to the data inputs. Data input A is the LSB. When the load input is brought low, the 4-bit input number will be loaded into the flip-flops. This preset function is also asynchronous and will override any synchronous counting functions that occur.

The input pulses to be counted are applied to either the up-count input or the down-count inputs. Instead of having a single count input and an up/down control line as described in the previous discussion of up/down counters, this IC counter uses two separate inputs. To increment the counter, pulses are applied to the up-count input. To decrement the counter, pulses are applied to the down-count input. The counter changes state on the leading edge of the applied input pulse. In other words, it is the binary 0-to-binary 1 transition at the count input that causes the counter to change state. In order for the counter to operate properly, the unused count input must be in the high or binary 1 state while count pulses are applied to the other input. The up and down count sequences are identical to those considered previously.

Synchronous operation is used in this counter by having the input count pulses clock all flip-flops simultaneously so that all outputs change coincidentally with one another. Instead of controlling each flip-flop by use of the JK inputs as in the previously discussed synchronous counter, gates are used ahead of the T inputs to the flip-flops for this purpose. The outputs of the flip-flops control the states of the gates ahead of the T inputs to permit the count pulse to be applied at the appropriate time. Gates 16, 17, and 18 are used to control the application of the up count pulses to the T inputs. The up count input is applied to these gates simultaneously. Note that the outputs of the previous flip-flops are connected to the inputs of these gates in order to control when the count pulse is allowed to toggle the flip-flop. Gates 13, 14 and 15 perform the same function for the down-count operation. Gates 20, 21, and 22 are simply OR gates that permit either the up or down count pulses to appear at the flip-flop T inputs.

This counter has both carry and borrow output gates which are used for cascading these counters. When it is necessary to use a counter with more than 16 states, several of these ICs can be cascaded to provide counters whose lengths are some multiple of 4.

The carry output is developed by gate 23. This NAND gate monitors the normal outputs of the flip-flops. When all of the normal outputs are binary 1 and the up-count pulse occurs, the carry output line will go low. The duration of the carry output pulse is equal to the duration of the count input pulse. This pulse indicates that the counter is in its maximum count state (1111) and that the next count input pulse will cause it to recycle to 0000. This carry output pulse is connected to the count up input of the next counter in sequence when counters are cascaded.

The borrow output is produced by gate 24. This gate monitors the complement outputs of the four flip-flops. The down count input is also applied to gate 24. When the counter has been decremented to the 0000 state and with the down count input high, the output of gate 24 will go low. This indicates the counter is in its lowest count state (0000) and that upon application of the next count input pulse it will recycle to 1111. To cascade 74193 counters for down counting applications, the borrow output is connected to the down count input of the next counter in series.

As you can see, this device is a very flexible counting unit. It can perform nearly any of the required basic counting functions often encountered in digital work. Figure 7-12 illustrates the operation of this counter. The waveform for the clear, preset, count up and count down operations are shown.

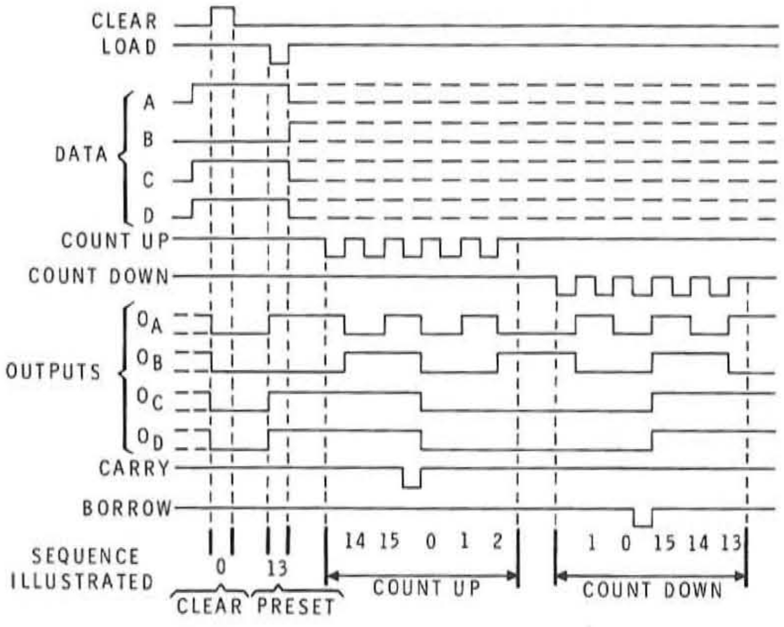


Figure 7-12
Typical clear, load, and count sequences for the 74193 counter.

Self Test Review

1. In a binary counter using JK flip-flops, the counter state will change when the T input changes from
   a. high to low
   b. low to high
   c. both a. and b.
2. A four bit binary counter contains the number 0100. Nine input pulses occur. The new counter state is:
   a. 0010
   b. 1001
   c. 1011
   d. 1101
3. A four bit binary counter contains the number 1010. Seven input pulses are applied. The new counter state is:
   a. 0001
   b. 0101
   c. 1100
   d. 1111
4. A binary counter constructed with two 74193 ICs has a maximum count capability of
   a. 15
   b. 16
   c. 255
   d. 256
5. A binary counter made up of 5 JK flip-flops will divide an input frequency by
   a. 5
   b. 8
   c. 16
   d. 32
6. Input pulses applied to a down counter cause it to be ______________.
7. Clearing a counter to zero is known as ______________.
8. Setting a counter to a desired state is called ______________.

9. A four bit down counter is in the 0110 state. Fourteen input pulses occur. What is the new output state?
   a. 0110
   b. 0100
   c. 1000
   d. 1110
10. The borrow output on the 74193 counter detects the counter state ______________.
11. An asynchronous counter is faster than a synchronous counter assuming the same flip-flops are used to implement both.
    a. True.
    b. False
12. The state of a binary counter is determined by monitoring the ______________ outputs of the flip-flops.
13. The state of the input or first flip-flop in a counter represents the ______________ of the number stored in the counter.
14. Synchronous operation of the flip-flops in the 74193 counter is obtained by controlling the
    a. JK inputs
    b. T inputs
    c. direct set and clear inputs.
15. In the 74193 IC counter, the counter is incremented or decremented by the
    a. leading edge
    b. trailing edge

    of the input pulse. The counter is reset by a

    c. binary 0
    d. binary 1

    on pin 14. The counter is preset by a

    e. binary 0
    f. binary 1

    on pin 11.

Answers

1. a high to low (1 to 0)
2. d. 1101 (4 + 9 = 13)
3. a. 0001 (10 + 7 = 17) The maximum count capability of a 4 bit counter is 15. With 10 in the counter initially, it will reach maximum counter (1111) after the fifth input is applied. The sixth input pulse will recycle the counter to 0000. The seventh input pulse will put the counter into the 0001 state.
4. c. 255. Each 74193 has four flip-flops.
   $2^8 - 1 = 256 - 1 = 255$
5. d. 32 $2^n = 2^5 = 32$
6. decremented
7. resetting
8. presetting
9. c. 1000 The first six input pulses, decrement the counter to 0000. The seventh pulse recycles the counter to 1111. The next seven pulses decrement the counter to 1000.
10. 0000
11. b. False. Synchronous counters are always faster than asynchronous counters.
12. normal
13. LSB
14. b. T inputs. Gates ahead of the T input controlled by the flip-flop states determine when the flip-flops toggle.
15. a. leading edge (0 to 1 transition)
    d. binary 1
    e. binary 0

EXPERIMENT 12

BINARY COUNTERS

OBJECTIVE: To demonstrate the operation and characteristics of a binary counter.

Materials Required

Heathkit ET-3200 Digital Design Experimenter

DC Voltmeter

2 — 7476 dual JK Flip-flop IC (443-16)
1 — 74193 synchronous up/down binary counter IC (443-612)
1 — 1 kΩ resistor

Procedure

1. On your ET-3200 Digital Design Experimenter, construct the four bit binary counter circuit shown in Figure 7-13. The pin connections for the 7476 ICs are given in Figure 7-14. Take care in wiring the circuit to avoid errors. Be sure to connect pin 5 of each IC to +5 volts and pin 13 of each IC to ground (GND). You will monitor the counter outputs on the LED indicators. You will step the counter with the A logic switch.

   Study the counter circuit in Figure 7-13.

   What type of counter is this? ______________.

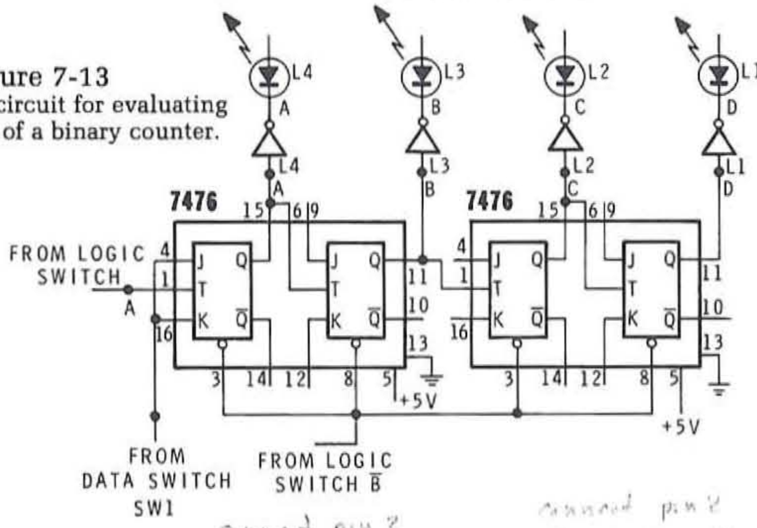


Figure 7-13
Experimental circuit for evaluating the operation of a binary counter.

Connect pin 8 & pin 7 = H

Connect pin 8 & pin 7 = H

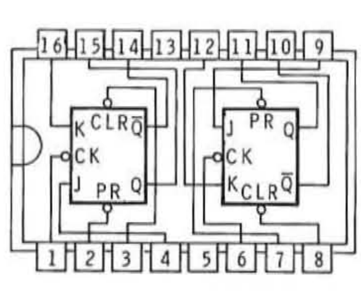


Figure 7-14 Pin connection for dual JK flip-flop 7476.

2. Set the switch SW1 to the high or up position. Apply power to the circuit. Note the states of the LED indicators. Record the binary number stored in the counter. Note: A (L4) = LSB, D (L1) = MSB
   DCBA = 1111.
3. Depress the B logic switch. Note the states of the LEDs. Record the value of the binary word stored in the counter.
   DCBA = 0000.
   Does the change that takes place in the outputs occur on the leading or trailing edge of the $\overline{B}$ signal? leading
4. Record the initial counter state obtained in Step 3 in the first (0) position of Table I. Using the A logic switch, step the counter. Each time you actuate the A logic switch, observe the LED outputs and record the counter contents in Table I.

   Does the counter state change when you depress or release the A logic switch? What does this tell you? Yes.
5. Observe your data in Table I. What kind of counter did you construct? Binary. Does this data confirm your answer given in Step 1? When the counter contents is DCBA = 1111, what happens when you depress the A logic switch? The counter output becomes what?
   DCBA = 0000
6. Remove the counter input from the A output and connect it to the CLK output. Set the clock frequency to 1 Hz. Depress the B pushbutton and hold it. Observe the LED indicators. Then release the B pushbutton and let the counter count. As it counts slowly, verify its outputs against your data in Table I. Let the counter run until you fully understand the count sequence.
7. As the counter is counting, set data switch SW1 to the low or down position and observe the result. Repeat several times to be sure you understand what happens. What effect does SW1 have? ________. Depress the B logic switch while the counter is counting. What happens? ________. Return the counter input to the A output of the logic switch.

TABLE I

	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Discussion of Steps 1 through 7

In Step 1 you constructed a four bit binary counter using JK flip-flops. This is a binary up counter of the asynchronous or ripple type since the normal output of one flip-flop is connected to the toggle (T) input of the next flip-flop.

When you apply power to the circuit, the flip-flops can come up in any random state. In Step 3 you used the B logic switch to reset the counter. The $\overline{B}$ output normally rests high so that it has no effect on the asynchronous clear inputs (C) of the flip-flops. When B is depressed, $\overline{B}$ goes low thereby putting all flip-flops into the binary 0 state. The counter resets on the leading edge of the $\overline{B}$ signal.

Next, you stepped the counter with the A logic switch, noted the output states on the LED indicators and recorded them in Table I. The counter is stepped or incremented on the trailing edge of the A input signal. The count input from A is normally low. When you depress the A logic switch, A goes high. A leading edge is generated. When you release the A logic switch, A goes low. A trailing edge is generated and the counter is incremented. By studying the data in Table I you can see that the counter generates the pure 8421 binary code.

An important observation you made in Step 5 was the recycling of the counter from state 1111 to 0000 when the 16th input pulse was applied.

Next in Step 6 you let the 1 Hz clock signal step the counter automatically. The states change slow enough for you to see each one and thus become familiar with this very common count sequence and the recycling step.

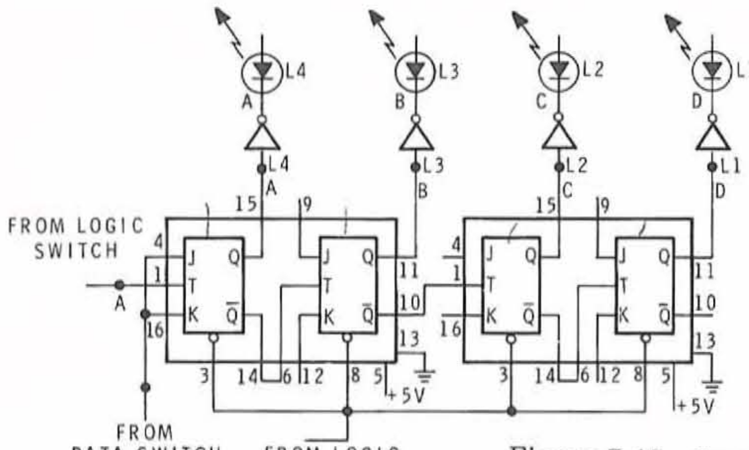
You should have found in Step 7 that you could stop the counter from counting in two ways. First, by setting data switch SW1 to the binary 0 position, the counter stops. What you did was to make the J and K inputs of the A flip-flop binary 0 and thereby inhibit its operation. Since the counter is the ripple type, naturally if A doesn't toggle, neither will any of the other flip-flops. The counter simply retains the last state it was in prior to making the JK inputs low. Putting SW1 back in the binary 1 states simply causes the counter to resume counting.

You can also stop the counter by resetting it. When you depressed the B logic switch, the counter is cleared. As long as the B switch is actuated, the clock pulses have no effect. The asynchronous clear inputs override the effect of the clock.

Procedure (continued)

8. Modify the counter to conform to the circuit in Figure 7-15. Be careful in making your wiring changes to avoid errors.

   Study the counter circuit you have just wired. What kind of counter is it? ______________.



[illegible] 5V

[illegible] 5V

Figure 7-15 Experimental circuit for Steps 8, 9, and 10.

9. Apply power and note the counter output state.
   DCBA = 0 1 1 1
   Now momentarily depress the B logic switch. Record the counter output state below.
   DCBA = 0 0 0 0
   What happened here? ______________.
10. Record the counter state obtained in the latter part of Step 9 in the first position of Table II. Use the A logic switch and step the counter. In Table II, record the counter state after each actuation of switch A. Continue stepping the counter until you fill Table II.
11. Convert the binary numbers you recorded in Table II into their decimal equivalents and record them in the far right hand column of Table II.
12. Observe your data in Table II. What kind of counter is this? ______________. Does this confirm your answer in Step 8?
13. Connect the counter input to the CLK output (1 Hz) as you did in Step 6. Let the circuit count. As you do, observe the output states and compare them to your data in Table II. Let the counter run for a while until you see the count pattern or sequence.
14. While the counter is stepping, set the SW1 logic switch to binary 0. Note the result. Next, while the counter is stepping, depress the B logic switch. What happens in each case? ______________

TABLE II

D	C	B	A	Decimal
0	0	0	0	0
1	1	1	1	15
1	1	1	0	14
1	1	0	1	13
1	1	0	0	12
1	0	1	1	11
1	0	1	0	10
1	0	0	1	9
1	0	0	0	8
0	1	1	1	7
0	1	1	0	6
0	1	0	1	5
0	1	0	0	4
0	0	1	1	3
0	0	1	0	2
0	0	0	1	1

Discussion of Steps 8 through 14

In Step 8 you constructed a binary down counter. The complement output of each flip-flop is connected to the T input of the next in sequence. As input pulses are applied, the counter is decremented. Each input pulse decreases the number in the counter by one. When the count is reduced to 0000, it will recycle to its maximum count (1111) when the next clock pulse is applied.

You should have found in Step 14 that this down counter circuit responds to the B logic switch (reset) and SW1 data switch exactly like the binary up counter.

Procedure (continued)

15. Wire the counter circuit shown in Figure 7-16. Use a type 74193 IC (443-612). Be sure to connect +5 volts to pin 16 and ground to pin 8. The pin connections for the 74193 IC are shown in Figure 7-17. You will step the counter with the A logic switch and reset it with the B logic switch. The data switches SW1 through SW4 serve as a parallel data source for presetting the counter.

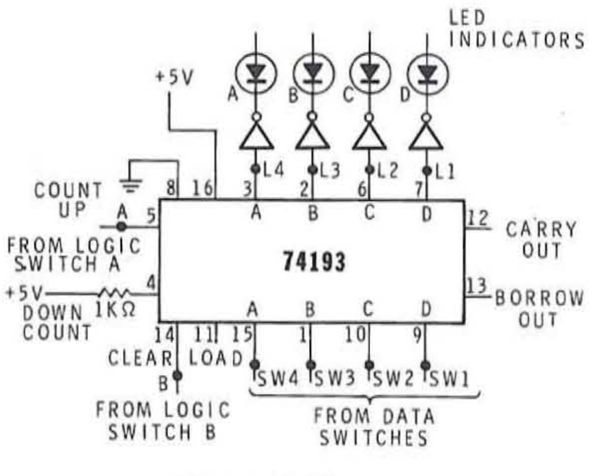


Figure 7-16
Counter circuit using a 74193 IC.

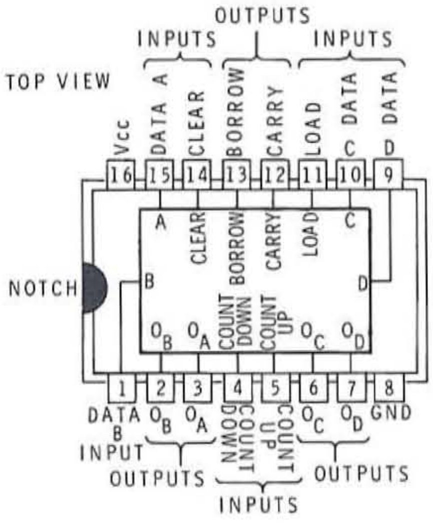


Figure 7-17
Pin connections for type 74193 IC binary up/down counter.

16. Apply power to the circuit. If the state of the counter is other than 0000, reset it with the B pushbutton. Step the counter 16 times with the A pushbutton. Note the output states on the LED indicators after each step. Compare the states to those you recorded in Table I.

    What type of count sequence does this IC counter generate? Binary count up.

17. Reverse the wires at pins 4 and 5 of the IC. Reset the counter with the B logic switch. Apply 16 pulses with the A logic switch and observe the counter output states. Compare them to the data you recorded in Table II. What kind of count sequence is generated? Binary count down.

18. Remove the wires connected to pins 4 and 5 of the IC. Connect pin 4 to +5 V through the 1 kΩ resistor. Connect pin 5 of the IC to CLK. Be sure the clock frequency is set to 1 Hz. Connect a DC voltmeter between GND and pin 12 of the IC. Set the meter for a reading in the zero to +5 volt range.

19. Note the voltmeter reading as the counter is stepped by the clock. Record below. At some point in the count cycle, the voltage at pin 12 will change momentarily. When it does, note the new output voltage and the binary state of the counter during the change. Record below.

    Voltage at pin 12 before the change 3.7 volts. (during counting)
    Voltage at pin 12 after the change 2.ø volts. (momentary)
    Binary code DCBA = 1111. (during momentary change)

20. Reverse the wires at pins 4 and 5 of the IC. Connect the voltmeter to pin 13. Repeat step 19. Record the data below.

    Voltage at pin 13 before change 3.7 volts. (during counting)
    Voltage at pin 13 after change 2.ø volts. (momentary)
    Binary code (during momentary change) DCBA = 0000.

21. Remove the wires at pins 4 and 5 of the IC. Connect pin 4 to +5 V through the 1 kΩ resistor. Connect pin 5 to the A output of logic switch A. Remove the wire between B and pin 14 on the IC. Connect pin 14 to ground. Connect $\overline{B}$ to pin 11. The DC voltmeter can also be removed at this time.

22. Set all of the data switches (SW1 — SW4) to binary 1. Depress the B logic switch. Note the output state of the counter.

    DCBA = 1111

    Next, set all data switches to binary 0. Depress the B logic switch and observe the LED indicators.

    DCBA = 0000

    Set the data switches to the words indicated below.

SW4	SW3	SW2	SW1
1	0	1	0
0	1	0	1
1	0	0	1
0	1	1	0

    After each word is set into the switches, depress B and note the counter state of the LED indicators. Compare this state to the data switch settings.

    What conclusion can you draw from this step? What function is taking place? Data is loaded.

23. Leave the data switches set to 0110. Depress the B logic switch. Note the counter state. Then start stepping the counter with the A logic switch.

    What happens? starts to count from data loaded

Discussion of Steps 15 through 23

The 74193 is a TTL MSI up/down counter. It operates synchronously and can be preset (parallel loaded) from an external 4 bit data source. In Step 15 you wired this IC as an up counter. The counter is cleared when B switches from low to high. The counter is incremented when the A logic switch is actuated. The state change occurs on the binary 0 to binary 1 (leading edge) transition of the A signal. As you determined by observing the outputs, the 74193 counts in the pure binary code. The outputs should be identical to those you recorded in Table I.

In Step 17 you applied the A count pulse to the down count input. With this connection the counter is decremented each time you press A. Your count sequence should have been similar to that you recorded in Table II.

In Steps 18, 19 and 20, you determined the operation of the carry and borrow outputs. The counter was stepped by the 1 Hz clock and you monitored the outputs with a dc voltmeter. During the up count sequence in Step 19, the carry output should have been high (about +3.5 volts). When the 1111 state occurs, the carry output should go low (about +0.1 volt) momentarily. The carry output indicates that the maximum counter value has been reached.

In the down count sequence in Step 20, you monitored the borrow output at pin 13 with the voltmeter. This output should also be high during the count. But when the 0000 state is reached, the borrow output goes low momentarily. The borrow output detects the minimum counter value. To cascade 74193 counters, the carry and borrow outputs of one counter are connected to the up and down count inputs respectively of the next counter in sequence.

In Steps 22 and 23 you demonstrated how the counter could be preset. The data switches served as your 4 bit parallel source, and you wired the B pushbutton to the load input control (pin 11). You should have found that the counter state became the same as the data switch state when the B logic switch was depressed. You parallel loaded the data switch word into the counter. The two important points to remember are that the counter assumed the state of the parallel inputs regardless of its previous contents. In other words, you did not have to reset the counter prior to presetting it. Second, the loading occurs when the load input at pin 11 goes low.

Finally, in Step 23 you stepped the counter after presetting it to 0110. Each actuation of the A logic switch should have incremented the counter. This illustrates that the count sequence simply starts at the preset point and continues in the normal binary sequence. The same applies for down counting.

Remember these operating details of the 74193 counter as you will use it later in demonstrating counter applications.

BCD COUNTERS

A BCD counter is a sequential circuit that counts by tens. The BCD counter has ten discrete states which represent the decimal numbers 0 through 9. Because of its ten state nature, a BCD counter is also sometimes referred to as a decade counter.

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

RECYCLE

Figure 7-18

Count sequence of 8421 BCD counter.

The most commonly used BCD counter counts in the standard 8421 binary code. The table in Figure 7-18 shows the count sequence. Note that a four bit number is required to represent the ten states 0 through 9. These ten four bit codes are the first ten of the standard pure binary code. As count pulses are applied to the binary counter, the counter will be incremented as indicated in the table. Upon the application of a tenth input pulse, the counter will recycle from the 1001 (9) state to the 0000 state.

An asynchronous 8421 BCD counter constructed with JK flip-flops is shown in Figure 7-19. This counter will generate the BCD code given in the Table of Figure 7-18. Note that the counter consists of four flip-flops like the four bit pure binary counter discussed earlier. The output of one flip-flop drives the T input to the next in sequence thereby, making this BCD counter a ripple or asynchronous type. Unlike the binary counter discussed earlier, however, this circuit has several modifications which permits it to count in the standard 8421 BCD sequence. The differences consist of a feedback path from the complement output of the D flip-flop back to the J input of the B flip-flop. Also a two input AND gate monitors the output states of flip-flops B and C and generates a control signal that is used to operate the J input to the D flip-flop. These circuit modifications in effect *trick* the standard four bit counter and cause it to recycle every ten input pulses.

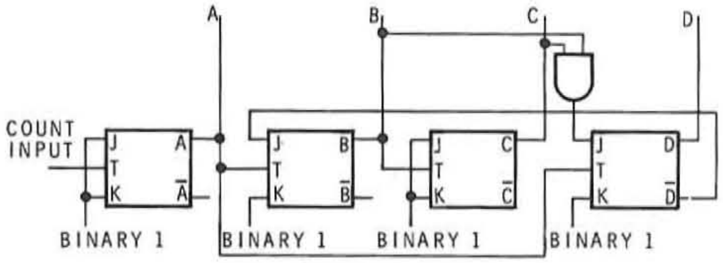


Figure 7-19
An asynchronous 8421 BCD counter.

The waveforms shown in Figure 7-20 illustrate the operation of the 8421 BCD counter. The count sequence is identical to that of the standard four bit pure binary counter discussed earlier for the first 8 input pulses. The operations that occur during the 9th and 10th pulses are unique to the BCD counter.

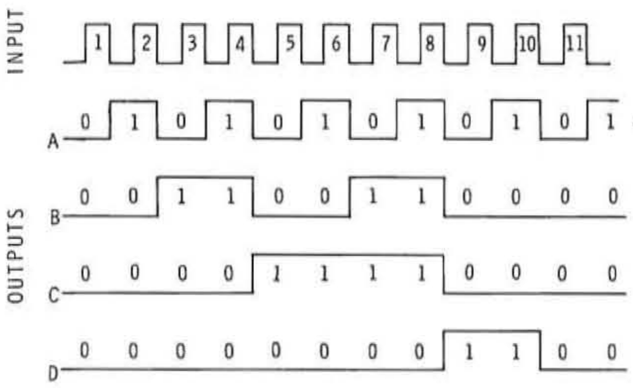


Figure 7-20
Waveforms of the 8421 BCD counter.

Assume that the counter in Figure 7-19 is initially reset. The outputs of flip-flops B and C will be binary 0 at this time. This makes the output of the AND gate low and causes the J input of the D flip-flop to be held low. The D flip-flop cannot be set by the toggle input from the A flip-flop until the J input goes high. Note also that the complement output of the D flip-flop which is binary 1 during the reset state is applied to the J input of the B flip-flop. This enables the B flip-flop permitting it to toggle when the A flip-flop changes state.

If count pulses are now applied, the states of the flip-flops will change as indicated in Figure 7-20. The count ripples through the first three flip-flops in sequence as in the standard 4 bit binary counter considered earlier. However, consider the action of the counter upon the application of the 8th input pulse. With flip-flops A, B, and C set and D reset, the B and C outputs are high thereby enabling the AND gate and the J input to the D flip-flop. This means that upon the application of the next count input that all flip-flops will change state. The A, B and C flip-flops will be reset while the D flip-flop is set. The counter state changes from 0111 to 1000 when the trailing edge of the 8th input pulse occurs.

In this new state, the B and C outputs are low therefore causing the J input to the D flip-flop to again be binary 0. With the J input 0 and the K input binary 1 and the D flip-flop set, the conditions are right for this flip-flop to be reset when the T input switches from binary 1 to binary 0. In addition, the complement output of the D flip-flop is low at this time thereby keeping the J input to the B flip-flop low. The B flip-flop is reset at this time and therefore the occurrence of a clock pulse at the T input will not affect the B flip-flop.

When the 9th input pulse occurs, the A flip-flop sets. No other state changes occur at this time. The binary number in the counter is now 1001. The transition of the A flip-flop switching from binary 0 to binary 1 is ignored by the T input of the D flip-flop.

When the 10th input pulse occurs, the A flip-flop will toggle and reset. The B flip-flop will not be affected at this time since its J input is low. No state change occurs in the C flip-flop since the B flip-flop remains reset. The changing of the state of the A flip-flop however, does cause the D flip-flop to reset. With its J input binary 0 and K input binary 1, this flip-flop will reset when the A flip-flop changes state. This 10th input pulse therefore causes all flip-flops to become reset. As you can see by the waveforms in Figure 7-20, the counter recycles from the 1001 (9) state to the 0000 state on the 10th input pulse.

Numerous variations of the basic BCD counter in Figure 7-19 are possible. Using the same basic count modifying techniques, synchronous BCD counter can be constructed. All of the flip-flops are toggled simultaneously by the common count input. As in the binary counter, the counting speed of the BCD counter can be significantly increased by this synchronous technique. In addition, it is also possible to construct a BCD down counter. Each time an input pulse is applied, the BCD counter is decremented. The count sequence is from 9 through 0.

Cascading BCD Counters. A single BCD counter has a maximum of ten discrete states and therefore can only represent the numbers 0 through 9. When the counter must count more than ten pulses, several BCD counters must be cascaded. Each BCD counter in the counting chain will represent one decimal digit. The number of BCD counters used determines the maximum count capabilities.

Figure 7-21 shows a counter chain with four BCD counters. Each BCD counter is represented by a single block to simplify the drawing. The count input line and the four flip-flop output lines are designated for each counter. In each case, the A output is the least significant bit and the D output is the most significant bit of that counter. The input BCD counter contains the least significant digit of the count contained in the counter. The most significant digit is represented by the counter on the far right. Since this counter contains four BCD counters, the maximum count capability is 9999.

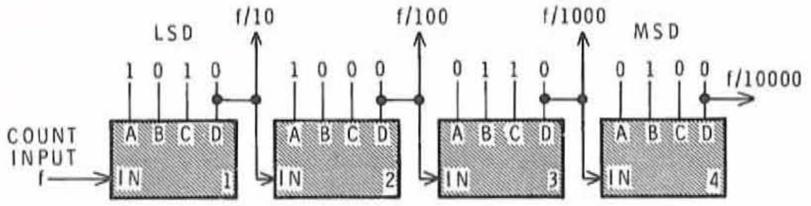


Figure 7-21 Cascading BCD counters to increase count capability.

As count input pulses are applied to BCD counter number 1, it will be incremented as indicated previously. The output states will change in accordance with the 8421 BCD code. Note that the most significant bit output (D) of this first counter is connected to the count input of the second BCD counter. Each time the input counter counts ten pulses and recycles it will trigger the next counter in sequence. By referring back to the waveforms in Figure 7-20, you can see that the trailing edge of the D output occurs on the trailing edge of the 10th input pulse. As this 10th input pulse occurs, the input counter recycles to 0 and the trailing edge increments the next counter in sequence to 1. The remaining counters in the chain are connected in the same way. As you can see then the counter does perform a decimal counting function with each BCD counter representing one of the decimal digits. The decimal contents of the counter can be determined by observing the flip-flop outputs. In Figure 7-21, the counter contains the decimal number 2615. This means that 2615 pulses have occurred at the input assuming the counter was initially reset.

The BCD Counter as a Frequency Divider. Like any counter, the BCD counter can also be used as a frequency divider. Since the BCD counter has ten discrete states, it will divide the input frequency by ten. The output of the most significant bit flip-flop in the BCD counter will be one tenth of the input frequency. From Figure 7-19, you can see that only a single output pulse occurs at the D output for every ten input pulses. While the D output does not have a 50 percent duty cycle, the frequency of the signal is nevertheless one tenth of the input frequency.

By cascading BCD counters, the input frequency can be reduced by any desired factor of ten. For example, in the counter of Figure 7-21, the output of the 4th BCD counter will be 1/10000th of the input frequency. The output of the third counter will be 1/1000th of the input frequency. The output of the second, of course, will be 1/100th of the input frequency. If an input signal of 2 MHz is applied to the counter in Figure 7-21, the D output of the MSD counter will be 200 Hz. When used as a frequency divider, the BCD counter is often referred to as a decade scaler.

Typical Integrated Circuit BCD Counter. The most widely used integrated circuit BCD counter is the type 7490A. This TTL MSI counter is an asynchronous or ripple counter that counts in the standard 8421 BCD code. The logic diagram of this counter is shown in Figure 7-22. Logically, it is identical to the BCD counter discussed earlier. The counter is made up of four JK flip-flops and the associated gating to permit the 8421 BCD sequence.

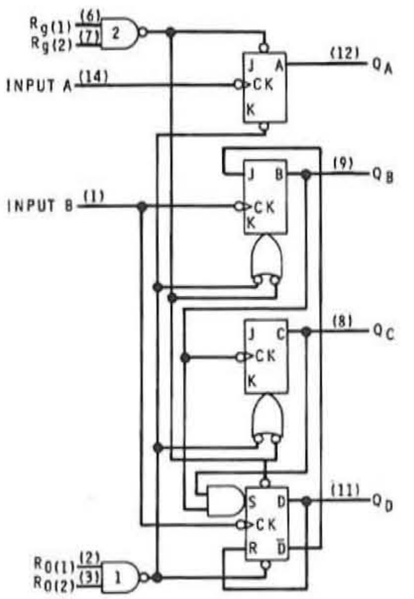


Figure 7-22
Logic diagram of 7490A BCD counter.

A look at the logic diagram of the 7490A counter in Figure 7-22 shows that the A flip-flop is not internally connected to the other three flip-flops. In order to produce an 8421 BCD count, the A output must be connected to the B input. This must be done externally. The input pulses to be counted then are applied to input A.

Gate 1 in Figure 7-22 is used to reset the flip-flop. When both inputs to gate 1 are high, all four flip-flops in the counter will be put in the binary 0 state. This permits two or more inputs to control the resetting of the flip-flop. Normally, only one input will be necessary and therefore both of the reset inputs can be simply tied together.

Gate 2 in the 7490A counter is used to preset the counter to the binary state 1001 or 9. When both inputs to gate 2 are high, a 9 will be preset into the counter. This particular function is useful in applications requiring arithmetic operations to be performed with BCD counters.

Despite the fact that the 7490A counter is an asynchronous counter, its maximum count frequency is approximately 32 MHz. This TTL MSI counter comes in a 14 pin DIP and is widely used in scaling and counting applications.

Self Test Review

16. A BCD counter can assume ______________ discrete states.
17. A BCD counter is in the 0111 state. How many input pulses were applied to it after it was reset?
    a. 3
    b. 6
    c. 7
    d. 12
18. A BCD counter divides its input signal frequency by ______________.
19. A 7490A IC is preset to 1001. Six count pulses are then applied. What is the counter state?
    a. 0000
    b. 0101
    c. 0110
    d. 1001
20. The BCD counter in Figure 7-21 has the following outputs.
    (1.) 1001 (2.) 0010 (3.) 1000 (4.) 0101
    How many input pulses does this represent? ______________.
21. If a 5 MHz signal was applied to the BCD counter of Figure 7-21, the output of counter 3 would be:
    a. 500 Hz
    b. 5 kHz
    c. 50 kHz
    d. 500 kHz
22. When used as a frequency divider, the BCD counter is referred to as a ______________ ______________.
23. A chain of 6 decade counters has a maximum count capacity of ______________.
24. A BCD counter is cascaded with a 3 flip-flop binary ripple counter. The overall frequency division ratio is:
    a. 20
    b. 30
    c. 60
    d. 80

Answers

16. 10
17. c. 7
18. 10

19. b. 0101 The first input pulse recycles the counter from 1001 to 0000. The next five pulses increment it to 0101.
20. 5829 (counter 4 is the MSD.)
21. b. 5 kHz Each counter divides by 10. The third counter in the chain has an output that is 10^3 lower than the input of 5 or MHz ÷ 1000 = kHz.
22. decade scaler.
23. 999999
24. d. 80 The BCD counter divides by ten. The 3 flip-flop binary ripple counter divides by $2^3 = 8$. The total division is the product of the two dividers or $8 \times 10 = 80$.

EXPERIMENT 13

THE BCD COUNTER

OBJECTIVE: To demonstrate the operation of an integrated circuit BCD counter.

Materials Required

Heathkit Digital Design Experimenter ET-3200

1 — 7490A (443-7) TTL BCD counter

Procedure:

1. Wire the circuit shown in Figure 7-23. You will use a 7490A TTL MSI decade counter. You will step the counter from logic switch A. The

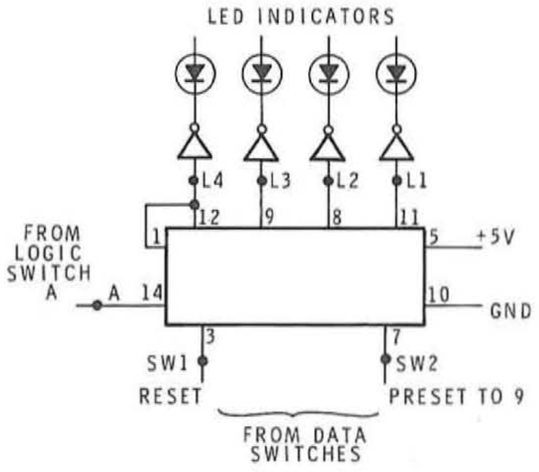


Figure 7-23
Circuit for BCD counter demonstration.

counter will be reset by using data switch SW1. You will demonstrate the preset to nine operation using data switch SW2. The counter states will be shown on the LED indicators. Be sure to connect +5 volts to pin 5 and ground to pin 10 on the IC. The pin connections for this device are shown in Figure 7-24.

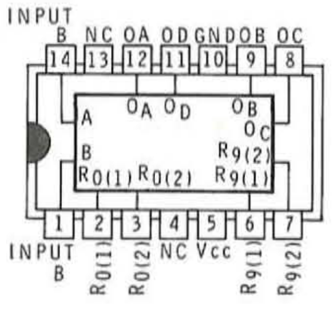


Figure 7-24
Pin connections 7490A counter.

TABLE III

D	C	B	A	Decimal
0	0	0	0	0
1	1	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

2. Be sure the data switches SW1 and SW2 are in the binary 0 position. Next, apply power to the circuit. Observe the states on the LED indicators. Momentarily move SW1 to the binary 1 position and then back to binary 0. Note the effect on the counter state. Record the number in the binary counter before and after you actuate data switch SW1.

DCBA = 0010 (before)
DCBA = 0000 (after)

3. In the first position of Table III, record the counter state that you observed after actuating SW1 in Step 2 above. Then step the counter with the A logic switch. After each actuation of logic switch A, note the LED indicator states and record them in Table III. Apply a total of ten input pulses with the A logic switch and complete Table III. Note particularly the counter state change when the 10th input pulse is applied.
4. Convert the binary numbers you recorded in Table III into their decimal equivalent and write them into the spaces provided in Table III. Then observing the data in Table III, verify the operation of the counter. What type of counting function does this IC perform? decade.
5. Momentarily move SW1 to binary 1 position and return it immediately to binary 0. Then set SW2 to the binary 1 position momentarily and then return it to binary 0. Note the effect on the output state and record the counter contents below.

DCBA = 0000 (SW1)
DCBA = 1001 (SW2)

Alternately set SW1 then SW2 to binary 0 several times to be sure that you see what effect that these two inputs have.

6. Remove the wire connecting pin 14 of the IC to the output of the A logic switch. Connect pin 14 to the CLK output. Set the clock frequency to 1 Hz. Let the clock step the counter. Observe the counter steps and follow the sequence by referring to the states you obtained and record it in Table III. Let the counter cycle for awhile to be sure that you see and understand the count sequence that it produces.

7. Remove the connections from the counter to LED indicators L2, L3, and L4. Only indicator L1 should be connected to the IC counter. When you're removing these connections, be sure to retain the connection between pins 1 and 12 on the IC. The L1 indicator that you have left connected is monitoring the most significant bit of the counter. Remove the connection between pin 14 on the IC and the CLK output. Connect pin 14 to the A output terminal of logic switch A.
8. Momentarily set SW1 to the binary 1 position. Then apply logic count pulses to the counter by actuating the A logic switch. Count the number of pulses that you apply to the circuit. While you are doing this, monitor the output state of L1. Each time that L1 turns on and then off, indicate its occurrence by marking a 1 in the margin of the page. At that time also note the number of input pulses that have been applied to the counter up to that point. Each time L1 turns on and then off, start the input count over again. How many input pulses occur for each single output pulse? 10.

Discussion

In this experiment, you demonstrated the operation of the 7490A TTL MSI BCD counter. As you should have discovered from evaluating the count sequence you recorded in Table III, this circuit counts in the standard 8421 BCD code. When the circuit reaches its maximum count of 9 (1001), the next input pulse causes the counter to recycle to 0000. The states 1010 through 1111 are invalid in a BCD counter.

You used data switch SW1 to reset the counter to 0. When power is first applied to the counter, it can come up into any state. By momentarily putting SW1 in the binary 1 position, the counter should reset to 0.

You demonstrated how data switch SW2 could be used to preset the counter to 9. When SW2 is momentarily moved to the binary 1 position, the counter is preset to 9 (1001). The preset to 9 operation is not a widely used counter function. However, it is used in some applications where certain arithmetic operations with BCD numbers are carried out with the counter.

In Steps 8 and 9 you demonstrated that the frequency counter divides by 10. Such a counter is generally referred to as a decade counter. The divide by 10 action is demonstrated by the fact that you should have recorded a change in the L1 output indicator for every 10 input pulses. With the counter starting at binary 0, the L1 indicator which monitors the D flip-flop output remains reset for the first 8 input pulses. On the ninth input pulse, the L1 indicator lights indicating that the D flip-flop has been set. Upon application of the 10th input pulse, the L1 indicator switches off. As you can see from the truth tables you developed in Table III, the D flip-flop is set for two counts. The D flip-flop sets and then resets every 10 input pulses.

SPECIAL COUNTERS

Binary and BCD counters are by far the most commonly used counters in digital systems. Most counting applications can be implemented with MSI binary and BCD counters. However, there are some applications where a special counter may be required. It may be necessary to count in a peculiar sequence or to have the counter sequence through the states of some special code. In other applications it may be desirable to divide or scale an input frequency by some value other than even powers of two or by ten. Special counters can be constructed to perform all of these applications. Some typical examples are a counter that counts in the Gray code or a frequency divider that divides the input by seven.

Because of the flexibility of the JK flip-flop, such special counters are relatively easy to implement. The basic approach is to construct a standard binary counter and then by using feedback and input gating controls on the JK inputs a counter can be developed to count in any sequence with as many individual states as desired.

We refer to the number of discrete states that a counter can assume as the modulus of that counter. A modulo N counter is one that has N states. A BCD or decade counter is a modulo 10 counter since it can assume one of ten discrete binary states. The binary counters that we discussed earlier are modulo N counters where N is some power of two. A counter containing four flip-flops is a modulo 16 counter. As you have seen it is easy to construct a binary counter whose modulo is some power of two. BCD counters with a modulo of 10 are also easily assembled. However, there are other applications that require counters with modulos of other integer values.

Modulo 3 Counter. A modulo three counter is shown in Figure 7-25. Since the T inputs to both JK flip-flops are both connected to the count input, the circuit is synchronous. The feedback line from the $\overline{B}$ output back to the J input of the A flip-flop causes the circuit to count by three.

To determine the operation of the counter assume that both flip-flops are initially in the reset state. The A and B outputs are binary 0. The low output of flip-flop A holds the J input to flip-flop B low. When the trailing edge of the first input pulse occurs, flip-flop A will set. The $\overline{B}$ output is holding the J input to A flip-flop high thereby enabling it to be set when the proper T input pulse occurs. When the first input pulse occurs, A is set. The B flip-flop is not affected. With the A output high, the J input to the B flip-flop is now high thereby permitting that flip-flop to toggle upon application of the next input pulse. The $\overline{B}$ output of the B flip-flop is still high thereby continuing to enable the J input of the A flip-flop.

When the trailing edge of the second input pulse occurs, the A flip-flop will again toggle. This time it will reset. At the same time the B flip-flop will set. The J input to the B flip-flop is now low while the J input to the A flip-flop is also low. When the next clock pulse occurs, the B flip-flop will reset. The A flip-flop having been previously reset simply remains reset. On the application of the trailing edge of the third input pulse, the counter state cycles back to its original reset condition. If further input pulses are applied, the counter will simply repeat the cycle just described. The count sequence for the modulo three counter is shown in Figure 7-25. The waveforms showing the operation of the modulo three counter are illustrated in Figure 7-26. Trace through the operation of the circuit using these waveforms as a guide to be sure you fully understand how the circuit operates. This circuit can be used for simple counting applications requiring a three state counter. The circuit can also be used as a divide by 3 scaler. The output of either the A or B flip-flops has a frequency that is one third of the count input frequency. A single output pulse occurs for every three input pulses as indicated by the waveforms in Figure 7-26.

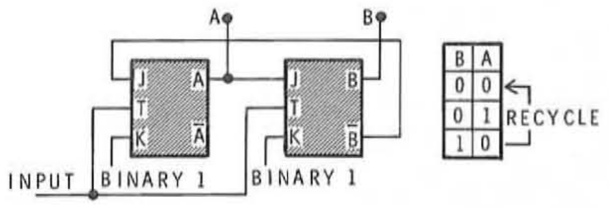


Figure 7-25 A modulo 3 counter and its count sequence.

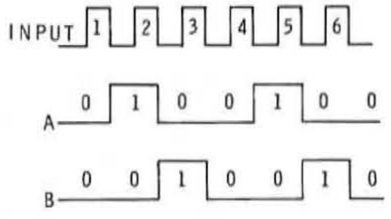


Figure 7-26 Waveforms for the modulo 3 counter.

By cascading a modulo three counter with additional JK flip-flops, modulo six and modulo twelve counters are easily formed. This is shown in Figure 7-27. By connecting a JK flip-flop to the output of the modulo three counter, a modulo six counter is formed. The modulo three will cycle through its three states twice, once with the JK flip-flop C reset and again with the JK flip-flop C set. This produces a total of six discrete states as indicated in the table shown in Figure 7-28A. If you will look closely at the sequence of the states you will find that this does not correspond to the standard binary code. Since the code produced is not the pure binary code or the BCD code, it is referred to as an unweighted code.

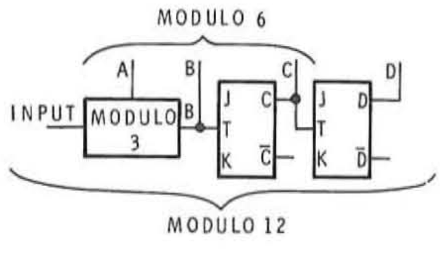


Figure 7-27
Forming modulo 6 and modulo 12 counters using a modulo 3 circuit as a base.

(A)

C	B	A
0	0	0
0	0	1
0	1	0
1	0	0
1	0	1
1	1	0

RECYCLE

COUNT SEQUENCE MODULO 6 COUNTER

(B)

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

RECYCLE

COUNT SEQUENCE FOR MODULO 12 COUNTER

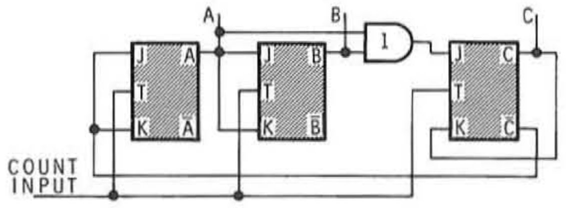
Figure 7-28
Count sequence for a (A) modulo 6 counter and a (B) modulo 12 counter.

By adding another JK flip-flop (flip-flop D) as shown in Figure 7-27, a modulo 12 counter is formed. Here the modulo 6 counter cycles through its six states two times, once while the D flip-flop is reset and again while it is set. This produces the 12 discrete states shown in Figure 7-28B. The counter shown in Figure 7-27 can be used to perform frequency division by three, six or twelve.

Counters with a modulo 6 and a modulo 12 can also be formed by putting the JK flip-flops ahead of the modulo three counter instead of after it. The counter thus formed still has six or twelve states respectively. However, the binary code produced by this arrangement is different from that obtained by the connection shown in Figure 7-27. When using this arrangement in frequency divider applications, it generally doesn't matter which connection is used as the output frequency will always be either 1/6th or 1/12th of the input frequency. Where the specific code sequence is critical however, these two arrangements should be carefully considered. It should be pointed out that the counter shown in Figure 7-27 produces a 50 percent duty cycle when the C and D outputs are used. By putting the modulo 3 counter after the cascaded JK flip-flops, a duty cycle other than 50 percent will be produced.

duty cycle

Modulo 5 Counter. A modulo five counter is shown in Figure 7-29. The counter produces five distinct three bit states. Synchronous operation is obtained by applying the count input to the T input of all three JK flip-flops. A combination of feedback and external logic gates are used to cause the counter to count by five.



C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0

RECYCLE

COUNT SEQUENCE

Figure 7-29 Modulo 5 counter and its count sequence.

The count sequence for this circuit is shown in Figure 7-29. Note that the count sequence simply recycles for the application of each five input pulses.

The input and output waveforms of the modulo five counter are illustrated in Figure 7-30. Compare the output states of these waveforms for each of the flip-flops to the count sequence table in Figure 7-29.

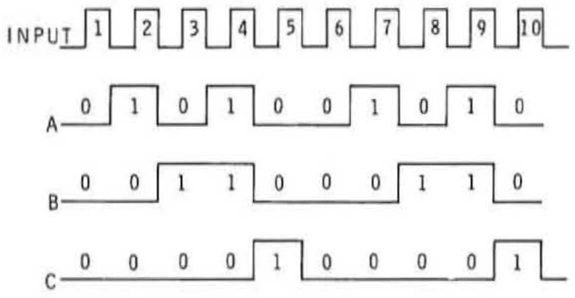


Figure 7-30
Waveforms for the modulo 5 counter.

The detailed operation of this counter is not included here. It will be excellent practice for you to reason out the count sequence of this circuit yourself. Remembering the operation of the JK flip-flop and using the count sequence table and the waveforms in Figure 7-30, trace the operation of the counter for all five states until it recycles. As a starting point, assume that all three flip-flops are initially reset.

If you will refer back to Figure 7-22 showing the diagram of the type 7490A decade counter you will see that flip-flops B, C and D in this circuit form a modulo 5 counter by themselves. As indicated earlier a separate input is used for this circuit. Normally, it is tied to the output of the A flip-flop to produce BCD counting. However, the modulo 5 section of this counter can be used independently by simply applying the count input to the B terminal. The A flip-flop is not used.

Modulo N Counters with MSI. While JK flip-flops can be interconnected by the use of feedback and external logic gates to form a counter with any desired modulo, the availability of MSI integrated circuit counters greatly simplifies the construction of modulo N counters. The type 74193 TTL MSI synchronous up/down counter discussed earlier is an excellent choice for implementing modulo N counters.

Figure 7-31 shows the type 74193 TTL MSI counter connected as a modulo N counter. The counter is connected to count down. For frequency divider applications, it does not matter whether the counter

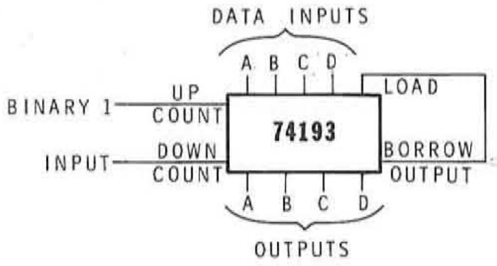


Figure 7-31
A type 74193 TTL MSI
Counter used as a modulo N counter.

counts up or down. The up count input is held to a binary 1 while the input pulses are applied to the down count input. The borrow output line which essentially detects the 0000 state is connected back to the load input line of the counter. To the four parallel data input lines is connected a binary word that will determine the modulus of the counter. The modulus of the counter is equal to the binary equivalent of the decimal number applied to the data inputs. This is indicated by the table in Figure 7-32. For example to obtain a modulo 7 counter, the binary number for the decimal number seven (0111) is connected to the data inputs.

MODULO	DATA INPUTS			
	D	C	B	A
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Figure 7-32 Parallel data input code and related decimal modulos.

In operation, the counter is preset to the binary number applied to the data inputs. The counter is then decremented by the input pulses. It down counts in binary until the zero state is reached. At this time the borrow output line goes low and again presets the counter to the number applied to the data inputs. This sequence repeats as long as pulses are applied to the count input.

As soon as the borrow output line goes low, the binary number applied to the data inputs will be immediately (asynchronously) loaded into the counter. Of course as soon as the new number is loaded into the counter, the borrow output will disappear since the counter state will no longer be 0000. What this means is that the duration of the borrow output pulse must be long enough in order to ensure that the data input is loaded before the zero output state disappears. In order to ensure that this condition happens, the input duration of the clock pulse must be greater than the total propagation delay of the gates in the counter associated with presetting the number on the data inputs. Recall from the previous discussion of the operation of the 74193 counter that the borrow output is also derived from the down count input signal.

Even though the borrow output pulse disappears as soon as the counter is preset to the data input states, the internal propagation delays of the circuit are such that all flip-flops become preset before the borrow output disappears. The load input signal must propagate through both the flip-flops and the gates in the circuit. Since the propagation delay through the various circuits in the counters vary from one device to the next, it is possible that erratic operation can occur if the propagation delays are too short. The reliability of the counter can therefore be improved by adding some propagation delay between the borrow output and the load input pins. This can be accomplished by cascading a number of inverters between these two pins on the circuit. Be sure to use an even number of inverters so the proper polarity binary signal will be applied.

Self Test Review

25. Determine the output frequency of the circuit shown in Figure 7-33. What is the overall circuit modulo? __________.
26. What is the modulo of the circuit in Figure 7-34? __________ Sketch the input and output waveforms and make a table of the count sequence.

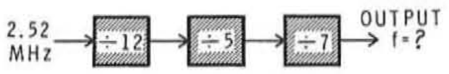


Figure 7-33 Circuit for Self Test Review question 25.

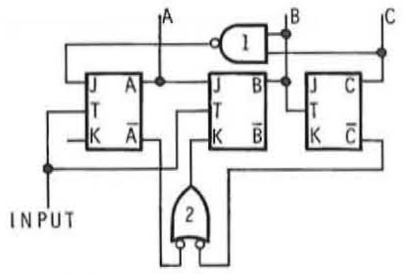
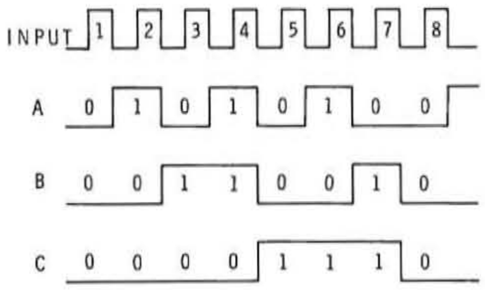


Figure 7-34 Circuit for Self Test Review question 26.

Answers

25. 6 kHz modulo 420 (12 × 5 × 7)
26. modulo 7 See Figure 7-35.



C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0

Figure 7-35 Waveforms and count sequence for modulo 7 counter in Figure 7-34.

NOTE: Using 2-7476 JK flip-flop ICs and a 7400 IC, you can breadboard the modulo 7 counter in Figure 7-34 on the Experimenter and verify its operation.

EXPERIMENT 14

COUNTER APPLICATIONS

Objective: To demonstrate several practical applications for binary and BCD counters.

Materials Required

Heathkit Digital Design Experimenter ET-3200

1 — 7400 IC (443-1)

2 — 7476 IC (443-16)

1 — 7490A IC (443-7)

1 — 74193 IC (443-612)

1 — 1 kΩ resistor

Procedure

1. Refer to the experimental circuit in Figure 7-36. This is a scaler circuit using counters. Study the circuit and determine the ratio by which it divides the input frequency. In the spaces provided below, record the input frequency and the frequencies at points X, Y, and Z in the circuit. Refer back to the appropriate sections in the unit or previous experiments if necessary to determine how the circuit operates.

Frequency at	INPUT	60	Hz
	X	6	Hz
	Y	2	Hz
	Z	1	Hz

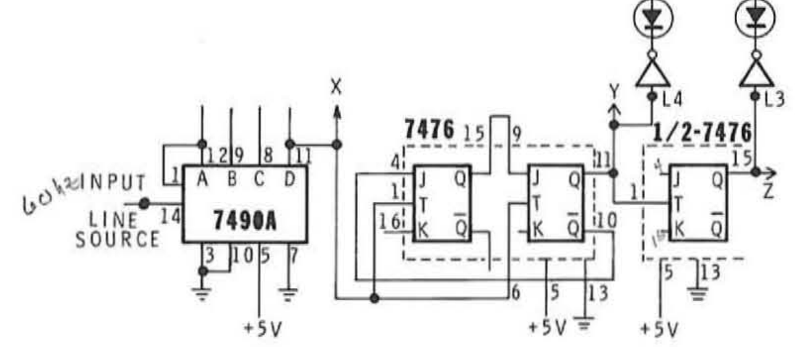


Figure 7-36
Scaler circuit for Steps 1, 2, and 3

Preset pin 2 & 7 must be tied high (5V)

2. Construct the circuit shown in Figure 7-36. As before take your time to be sure that the circuit is wired correctly. As the experiments become more sophisticated, the number of integrated circuits to be interconnected increases. This also increases the chances of your making a wiring mistake. If the circuit does not perform properly, the first thing to check is your circuit wiring. Be sure that you have connected +5 volts and ground to each of the integrated circuits in the experimental circuit.
3. To verify the operation of the circuit, apply power and observe the outputs Y and Z on LED indicators L4 and L3 respectively. The frequency of the output pulses at Z will be slow enough for you to count them. Count the number of pulses at Z that occur in one half minute using the sweep-second hand on your watch as a timing indicator. Record the number of pulses occurring in a half minute in the space provided. ____30____.

   What is the overall frequency division ratio of this circuit? ____60____. What is the basic function of this circuit? ____divide____.
4. Refer to the circuit shown in Figure 7-37A. Study this circuit carefully noting the function of each logic element in the circuit. Analyze the operation of the circuit. To do this, assume that the 74193 counter is initially reset. Also assume that the latch made up of gates 1 and 2 is also initially reset so that the output at pin 6 of gate 2 is binary 0. Assume that the operation of the circuit is started by actuating the B logic switch a single time. Assume that the data switches are set so that SW4 = 1, SW3 = 0, SW2 = 1, and SW1 = 0. Sketch the output of gate 4.

Count down n pulses and stop. load parallel & count down from parallel data

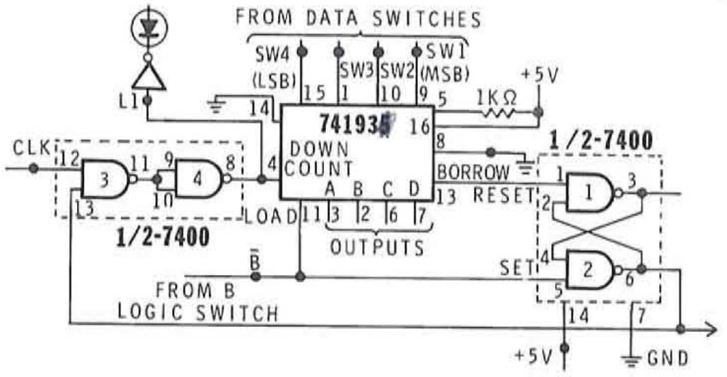


Figure 7-37A Experimental circuit for Steps 4 and 5.

5. Construct the circuit shown in Figure 7-37. The circuit will be driven by the clock output. Set the clock frequency to 1 Hz. You will observe the output of the circuit on LED indicator L1. Set the data switches as indicated in Step 4.
6. Apply power to the circuit. L1 should be off at this time, however, if L1 begins to blink off and on, let the circuit run for several seconds until L1 goes off and remains off. The circuit is now initialized. To start the operation of the circuit, depress the B logic switch once. Then observe the L1 output. Count the number of output pulses that occur before L1 turns off and remains off.

   Set the date switches so that SW1 = SW2 = 1 and SW3 = SW4 = 0. Record below the binary and decimal numbers represented by this word. Momentarily depress the B logic switch to initiate the operation of the circuit. Again count the number of output pulses you observe at L1 before the circuit stops.
7. Using the information that you collected in Step 6 above, compare the number of output pulses occurring on L1 with the decimal value of the binary number loaded into the 74193 counter. How are they related? What is the basic function of this circuit?

Discussion

The circuit shown in Figure 7-36 is a divide by 60 scaler. It accepts the 60 Hz waveform from the line source output on the Experimenter and divides it by 60. The Z output is 1/60 of the input frequency or 1 Hz.

Close analysis of this circuit will show that the output at point X is 1/10 of the input frequency or 6 Hz. The 7490A decade counter is used as a divide by 10 circuit.

The first 7476 IC in the circuit is connected as a divide by 3 counter. The output at point Y then is 1/3 of the frequency at point X or 6 ÷ 3 = 2 Hz. This signal is applied to one of the JK flip-flops in the other 7476 IC. It divides the frequency of point Y by 2 to produce an output frequency of 1 Hz. The signal at Z is a 1 Hz square wave with a 50 percent duty cycle.

Since the line source in the Experimenter is derived from the 60 Hz power line, the input frequency is very accurate. The frequency of the power line voltage has an error typically less than 0.1 percent. For that reason the output frequency at Z is a very accurate 1 second source. With a 50 percent duty cycle at the output, the LED indicator L3 remains on for one half second and then off another half second.

When you counted the number of output pulses on L3 occurring in one half minute, you should have recorded a value of 30. One half minute, of course, is equal to 30 seconds and a 1 Hz signal will cause 30 pulses to occur during that period of time.

The circuit in Figure 7-37A is designed to generate a fixed number of output pulses upon command and then stop automatically. The number of output pulses to be generated by the circuit is determined by the binary number input set into the data switches. For example, you set in the binary number 0101 which is the binary equivalent of the decimal number 5. When the B logic switch is actuated then the circuit will generate 5 output pulses and then stop. These output pulses occur at the clock rate and you observed them on LED indicator L1. Here's how the circuit operates.

Assume that the 74193 binary counter is initially reset. Note that it is wired as a down counter and as the count pulses are applied to the down count input at pin 4. Another clue to the use of this device as the down counter is the use of the borrow output. The borrow output is effective only in the down counting mode. Assume also that the latch circuit made up of gates 1 and 2 is also reset. This means that the output of gate 2 is low. This low output inhibits the AND gate made up of gates 3 and 4 in the 7400 IC. The external clock pulse is applied to this gate. The clock pulse will step the counter when gate 3 is enabled.

The signal from the B logic switch is used to initiate the circuit operation. When the B logic switch is depressed, the B output goes low. This signal causes two things to happen. First, it forces the load input on the 74193 low thereby presetting the counter to the binary number set on the data switches, in this case 0101. At the same time that this signal sets the latch made up of gates 1 and 2. With the latch set, the output of gate 2 is high. Gate 3, therefore is enabled and clock pulses will pass through gate 3 and gate 4 (which is connected as an inverter) and will cause the counter to step. The counter will down count starting at its preset number 5. The counter will continue to step until the 0 condition is reached. At this time the borrow output line will go low during the clock interval. This will cause the latch to become reset. As the latch resets, gate 3 again becomes inhibited and the clock pulses no longer reach the counter.

During the time that the counter is decrementing, LED indicator L1 monitors the clock pulses occurring at the output of inverter 4. Since it takes five clock pulses to decrement the counter to 0, this LED indicator will switch off and on five times. This indicates that five output pulses will occur. This corresponds to the binary number preset into the counter.

You again demonstrated the operation of the circuit by programming the counter with the number 1100 or 12. When the B logic switch is actuated, the circuit considered generated 12 output pulses before it stops. The waveforms produced by this circuit are illustrated in Figure 7-37B. These waveforms assume that the circuit is programmed to generate five output pulses.

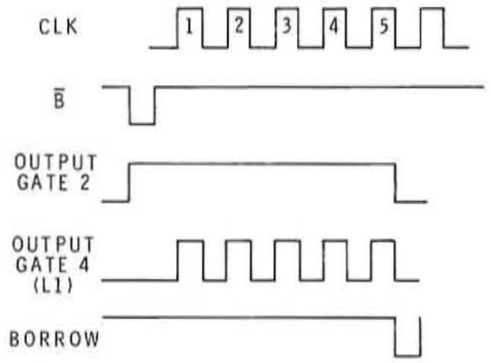


Figure 7-37B Waveforms for the circuit in Figure 7-37A.

The number of output pulses generated by this circuit is limited by the count capability of the 74193 counter. Its maximum count is 1111 or 15. In order to extend this to larger values, additional 74193 counters can be cascaded to accommodate the desired number of pulses.

SHIFT REGISTERS

Another widely used type of sequential logic circuit is the shift register. Like a counter, a shift register is made up of binary storage elements. While flip-flops are the most commonly used storage element in shift registers, other types of circuits are also used. The storage elements in a shift register are cascaded in such a way that the bits stored there can be moved or shifted from one element to another adjacent element. All of the storage registers are actuated simultaneously by a single input clock or shift pulse. When a shift pulse is applied, the data stored in the shift register is moved one position in one of two directions. The shift register is basically a storage medium where one or more binary words may be stored. However, because of the ability to move the data one bit at a time from one storage element to another makes the shift register valuable in performing a wide variety of logic operations.

Shift Register Operation

The illustration in Figure 7-38 shows how a shift register operates. Here the shift register consists of four binary storage elements such as flip-flops. The binary number 1011 is currently stored in the shift register. Another binary word, 0110, is generated externally and is available to the shift register serially. As shift pulses are applied, the number stored in the register will be shifted out and lost while the external number will be shifted into the register and retained.

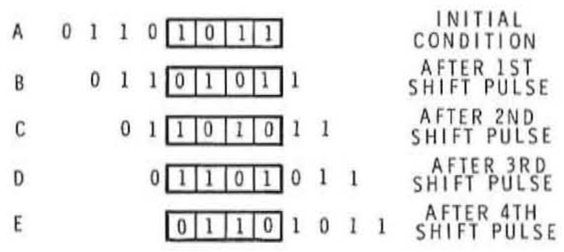

A 0 1 1 0 [1|0|1|1] INITIAL CONDITION
B 0 1 1 [0|1|0|1] 1 AFTER 1ST SHIFT PULSE
C 0 1 [1|0|1|0] 1 1 AFTER 2ND SHIFT PULSE
D 0 [1|1|0|1] 0 1 1 AFTER 3RD SHIFT PULSE
E [0|1|1|0] 1 0 1 1 AFTER 4TH SHIFT PULSE

Figure 7-38
Operation of a shift register.

The initial conditions for this shift register are illustrated in Figure 7-38A. After one clock pulse, the number stored in the register initially is shifted one bit position to the right. The right most bit is shifted out and lost. At the same time, the first bit of the externally generated serial number is shifted in to the left most position of the shift register. This is illustrated in Figure 7-38B. The remaining three illustrations in C, D, and E show the results after the application of additional shift pulses. After four shift pulses have occurred, the number originally stored in the register has been completely shifted out and lost. The serial number appearing at the input on the left has been shifted into the register and now resides there.

This figure illustrates several important points about a shift register. First, it indicates that the basic shift register operations are serial in nature. That is data is moved serially, a bit at a time, into and out of the register. Most shift register operations are serial operations but many circuits are provided with both parallel inputs and parallel outputs. Such shift registers permit data to be preset in parallel and data to be read out in parallel. The ability to combine both serial and parallel operations makes the shift register an ideal circuit for performing serial to parallel and parallel to serial data conversions.

Another important point to note is that the data is shifted one bit position for each input clock or shift pulse. Clock pulses have full control over the shift register operation. In this shift register, the data was shifted to the right. However, in other shift registers it is also possible to shift data to the left. The direction of the shift is determined by the application. Most shift registers are of the shift right type.

The shift register is one of the most versatile of all sequential logic circuits. It is basically a storage element used for storing binary data. A single shift register made up of many storage elements can be used as a memory for storing many words of binary data. Such memories are referred to as serial memories since the data stored in them is entered and removed in serial form.

Shift registers can also be used to perform arithmetic operations. Shifting the data stored in a shift register to the right or to the left a number of bit positions is equivalent to multiplying or dividing that number by a specific factor. As indicated earlier, the shift register is also widely used for serial to parallel and parallel to serial data conversions. Shift registers can also be used for generating a sequence of control pulses for a logic circuit. And in some applications shift registers can be used to perform counting and frequency dividing.

In this section you are going to study the basic operation of a shift register. One of the most commonly used types of shift registers is the bipolar shift register, which is made up of flip-flops. These can be constructed with individual JK flip-flops or are available in a variety of configurations in MSI form. Another type of shift register is the MOS shift register. These registers made with MOSFETs are available in two basic types, static and dynamic. Static shift registers are made up of MOSFET flip-flops. Dynamic shift registers are made up of storage elements that take advantage of the unique characteristics of MOSFETs, namely their high impedance and capacitive nature. Because of the small size of the MOSFET structure, many storage elements can be made on a single chip of silicon. Therefore, long shift registers capable of storing many words can be made very small and economical. Both types of shift registers are widely used in digital systems.

Bipolar Logic Shift Registers

Shift registers constructed from bipolar logic circuits such as TTL and ECL circuits are usually implemented with JK flip-flops. Type D flip-flops can also be used, but shift registers implemented with JK flip-flops are far more versatile. A typical shift register constructed with JK flip-flops is shown in Figure 7-39. The serial input data and its complement are applied to the JK inputs of the input (A) flip-flop. From there the other flip-flops are cascaded with the outputs of one connected to the JK inputs of the next. Note that the clock (T) input lines to all flip-flops are connected together. The clock or shift pulses are applied to this line. Of course, since all flip-flops are toggled simultaneously, the shift register is definitely a synchronous circuit. Note that the asynchronous clear inputs on each flip-flop have been connected together to form a reset line. Application of a low or binary 0 level to this line causes the shift register to be reset. This shift register can also be preset by using any of the techniques described earlier for presetting binary counters. Data applied to the input will be shifted to the right through the flip-flops. Each clock or shift pulse will cause the data at the input and that stored in the flip-flops to be shifted one bit position to the right.

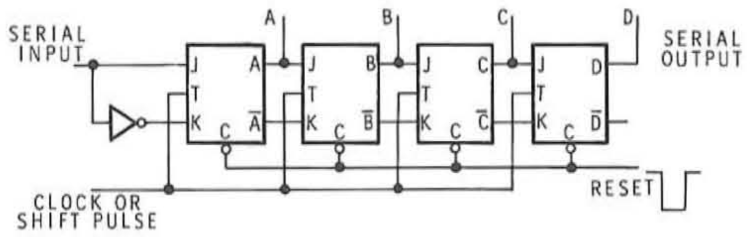


Figure 7-39 Four bit shift register made with JK flip-flops.

The waveforms in Figure 7-40 illustrate how a serial data word is loaded into the shift register of Figure 7-39. As the waveforms show, the binary number 0101 in serial form occurs in synchronization with the input clock or shift pulses. In observing the waveforms in Figure 7-40, keep in mind that time moves from left to right. This means that the clock pulses on the right occur after those on the left. In the same way, the state of the serial input shown on the left occurs prior to the states to the right. With this in mind, let's see how the circuit operates.

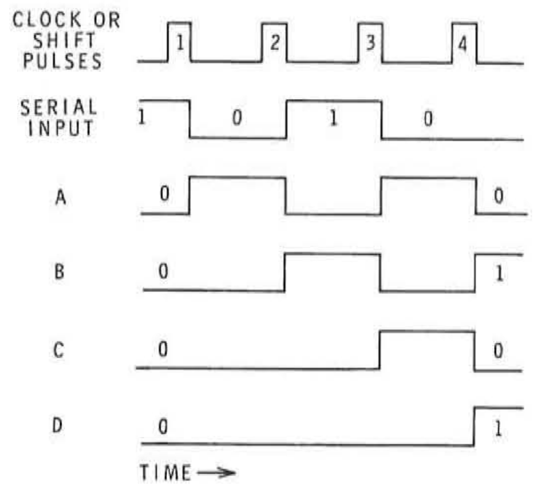


Figure 7-40
Waveforms illustrating how the serial binary number 0101 is loaded into a shift register.

Note that the shift register is originally reset. The A, B, C, and D outputs of the flip-flops therefore are binary 0 as indicated in the waveforms. Prior to the application of the number 1 shift pulse, the serial input state is binary 1. This represents the first bit of the binary word to be entered. On the trailing edge of the first clock pulse, the binary one will be loaded into the A flip-flop. The JK inputs of the A flip-flop are such that when the clock pulse occurs the flip-flop will become set. This first shift pulse is also applied to all other flip-flops. The state stored in the A flip-flop will be transferred to the B flip-flop. The states stored in the B and C flip-flops will be transferred to the C and D flip-flops respectively. Since all flip-flop states are initially zero, naturally no state changes in the B, C or D flip-flops will take place when the first clock pulse occurs.

After the first clock pulse the A flip-flop is set while the B, C, and D flip-flops are still reset. The first clock pulse also causes the serial input word to change. The clock or shift pulses are generally common to all other circuits in the system and therefore any serial data available in the system will generally be synchronized to the clock.

The input to the A flip-flop is now binary 0. When the trailing edge of the second clock pulse occurs, this binary 0 will be written into the A flip-flop. The A flip-flop which was set by the first clock pulse causes the JK inputs to the B flip-flop to be such that it will become set when the second clock pulse occurs. As you can see by the waveforms, when the second clock pulse occurs, the A flip-flop will reset while the B flip-flop will set. The 0 state previously stored in the B flip-flop will be transferred to the C flip-flop, and the C flip-flop state will be shifted to the D flip-flop. At this point the first two bits of the serial data word have been loaded into the shift register.

The serial input is now binary 1 representing the third bit of the serial input word. When the third clock pulse occurs the A flip-flop will set. The zero previously stored in the A flip-flop will be transferred to the B flip-flop. The binary 1 stored in the B flip-flop will now be shifted into the C flip-flop. The D flip-flop remains reset.

The serial input to the A flip-flop is now binary 0. When the trailing edge of the fourth clock pulse occurs, the A flip-flop will reset. The binary 1 stored there previously will be transferred to the B flip-flop. The 0 stored in the B flip-flop will be shifted into the C flip-flop. The binary 1 in the C flip-flop now moves to the D flip-flop. As you can see, after four clock pulses have occurred, the complete four bit binary word 0101 is now shifted into the register as indicated by the states shown in the waveforms. A glance at the flip-flop output waveforms will show the initial binary 1 bit moving to the right with the occurrence of each shift pulse.

While we have illustrated the operation of the shift register with only four bits, naturally as many flip-flops as needed can be cascaded to form longer shift registers. Most shift registers are made up to store a single binary word. In most modern digital systems, shift registers have a number of bits that is some multiple of four.

While shift registers are readily implemented with JK or D type flip-flops, in most applications, MSI shift registers are used. MSI shift registers are available in four and eight bit sizes. Here we are going to discuss a typical four bit MSI TTL shift register. You will see how it can perform the basic shift right operation described previously and how it can be connected to shift left or be parallel loaded.

Figure 7-41 shows the logic diagram of a type 7495 TTL shift register. It is made up of four flip-flops with the appropriate gating on the JK inputs. A mode control input line controls this input gating. The mode control also operates the clock input selection circuitry. Two clock signals can be used depending upon the state of the mode control.

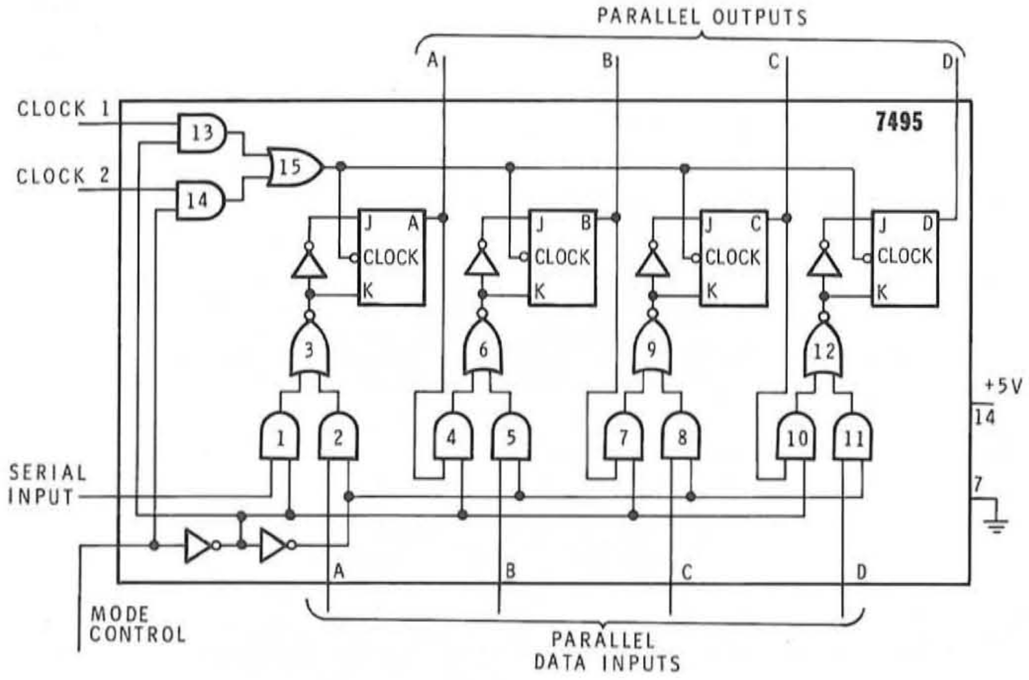


Figure 7-41 Logic diagram of 7495 TTL MSI shift register.

When the mode control input is a binary 0, gates 1, 4, 7, and 10 are enabled. This causes the shift register to be set up to perform the basic shift right operation. Serial input is applied to gate 1 and passes through gate 3 to the JK inputs of the flip-flop. The output of the A flip-flop is connected to the inputs of the B flip-flop through gates 4 and 6. In the same way, the outputs of the B and C flip-flops are connected to the inputs of the C and D flip-flops respectively. Also notice that a binary 0 on the mode control input also enables gate 13. This permits clock pulse 1 to pass through gates 13 and 15 to control the flip-flops. In this mode, the shift register performs the standard shift right operation.

When the mode control is placed in the binary 1 state, gates 2, 5, 8, and 11 are enabled. With these gates enabled and gates 1, 4, 7, and 10 inhibited, the parallel data inputs are recognized. Also note that a binary 1 on the mode control input also enables gate 14 so that clock pulse 2 can actuate the flip-flops. When a clock pulse occurs, an external 4-bit parallel word will be loaded into the flip-flops. In this mode, the shift register can be parallel loaded or preset to some desired value.

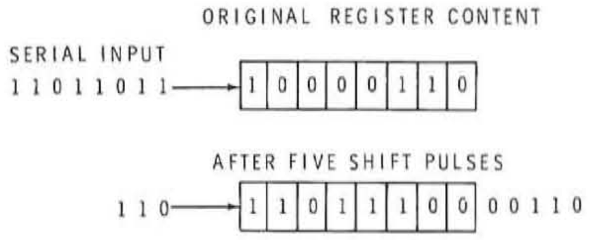
Shift left operations can be performed with the mode control in the binary 1 position if the parallel data input lines are connected to the appropriate flip-flop outputs. To perform a left shift, the D flip-flop output is connected to the C data input, the C output is connected to the B data input, and the B flip-flop output is connected to the A data input. The D data input is used as the serial input line for external data. When clock pulses are applied to gate 14, data will be shifted left from D to C, C to B, and B to A. External serial data is shifted into D. With this connection, the mode control input acts as a shift right, shift left control line.

Self Test Review

27. An 8 bit shift register contains the number 10000110. The serial number 11011011 is applied to the input. After 5 shift pulses, what is the number in the shift register? (Assume shift right operation.)
28. Shift registers can be made up of____________ or ____________ type flip-flops.
29. Most shift registers in modern digital applications are of the ____________type.
30. Which of the following operations are *not* typical of the 7495 MSI shift register?
    a. shift right
    b. shift left
    c. serial in
    d. serial out
    e. reset
    f. parallel load
31. How many shift pulses are required to serially load a 16 bit word into a 16 flip-flop shift register?________
32. How could you reset an 8 bit serial in-serial out shift register?

Answers

27.



28. JK, D
29. MSI
30. e. reset The 7495 shift register does not provide a separate clear input line.
31. 16
32. Shift in eight binary 0s.

EXPERIMENT 15

SHIFT REGISTERS

Objectives: To demonstrate the operation and characteristics of bipolar integrated circuit shift registers.

Materials Required

Heathkit Digital Design Experimenter ET-3200

1 — 7404 IC (443-18)

2 — 7476 IC (443-16)

1 — 7475 IC (443-13)

1 — 7495 IC (443-680)

Procedure

1. Construct the four bit shift register circuit shown in Figure 7-42. You will use two type 7476 dual JK flip-flops. Shift pulses for the circuit will be derived from logic switch A. Logic switch B is connected to reset the shift register via the asynchronous clear inputs on the flip-flops. The serial input to the shift register will come from data switch SW1. Note that one of the inverters in a 7404 IC is used to make the JK inputs of the first flip-flop complementary. The shift register outputs will be monitored on the LED indicators. Don't forget to connect +5 volts and ground to each IC.

Connect pin 2 & 7 of each 7476 to 5V

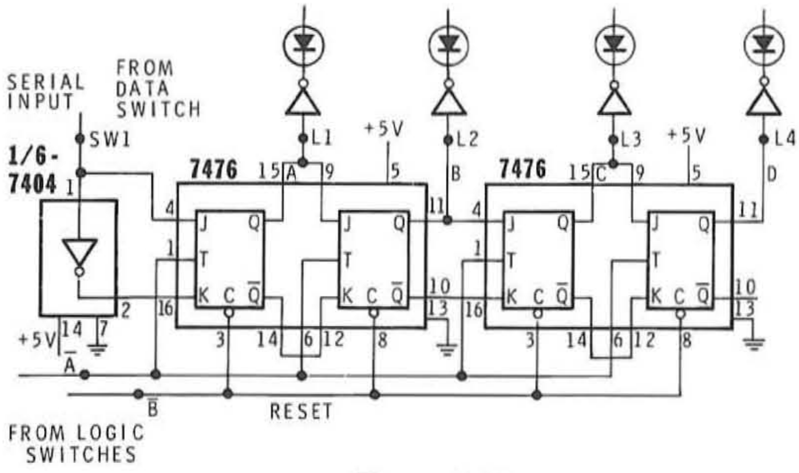


Figure 7-42
Shift register demonstration circuit.

2. Apply power to the circuit. Reset the shift register by actuating the B logic switch. Set data switch SW1 to the binary 1 position. Then using the A logic switch, apply 4 shift pulses. Observe the LED indicators as you do. Record the binary value of the register contents after four shift pulses.

   ABCD = 1111

   Next, set the SW1 switch to binary 0. Again apply four shift pulses with the A logic switch. Again record the binary contents of the register.

   ABCD = 0000

3. Using the SW1 data switch and the A logic switch, load the shift register by using the step by step procedure given below.

   SW1 = 1, depress switch A
   SW1 = 0, depress switch A
   SW1 = 1, depress switch A
   SW1 = 0, depress switch A

   After you have loaded the shift register, observe the LED indicators and in the space below write the decimal equivalent of the binary number in the shift register.

   Decimal number = 5

Discussion of Steps 1 through 3

In these steps you constructed a four bit shift register with JK flip-flops. You loaded data into the shift register serially using data switch SW1 as your data source. A single bit of information was entered into the shift register with each actuation of the A logic switch. As the bits were entered into the A flip-flop, they were shifted to the right one bit at a time for each shift pulse.

The 7404 inverter is used to convert the data from SW1 into two complementary signals which are applied to the JK input of the input (A) flip-flop. The JK inputs must always be complementary in a shift register using a JK flip-flop.

In Step 2 you applied a binary 1 to the shift register input. Then actuating the A logic switch four times, you generated four shift pulses which loaded the shift register with binary 1s. As you generated the shift pulses, you should have noted the entry of the first binary 1 bit into the A flip-flop and with the second shift pulse, the shifting of the data to the right until the register was full (1111). Next, you set the input to binary 0 and then loaded the register with binary 0s with four shift pulses (0000). As the binary 0s were loaded, the binary 1 bits in the register were shifted out.

In Step 3 you loaded the binary number 0101 into the register a bit at a time by setting SW1 to the desired state and then depressing a logic switch to generate a shift pulse. While the bit pattern in the shift register is easy to identify by simply observing the LED indicators, you cannot convert that bit pattern into its decimal number equivalent unless you know which bit is the LSB. Since this information was not given, your answer could have been either 0101 = 5 or 1010 = 10. In a binary counter the least significant bit is readily identified since it is the flip-flop to which the input or count pulses are applied. With this information you can always determine the value of the number in the counter. With the shift register, however, the input flip-flop may not necessarily be the least significant bit. We could also assign the right most or serial output flip-flop as the least significant bit. The choice of weight for the input and output flip-flops is up to the designer and will depend upon his application. For a large percentage of applications, the input flip-flop contains the most significant bit while the output flip-flop contains the least significant bit. In other words, data is entered into the shift register beginning with the least significant bit. As data is shifted out of the shift register the least significant bit is shifted out first. Unless otherwise stated, we will use that convention here. As you can see from the demonstration described here, the shift register does perform a serial to parallel conversion. Data is entered serially from the SW1 data switch and then displayed in parallel on the LED indicators. Data can also be entered serially with SW1 and read-out serially by simply observing the state of L4.

Procedure (continued)

4. Construct the shift register circuit shown in Figure 7-43. At this time you can disassemble the shift register circuits used in the previous steps. The shift register shown in Figure 7-43 is a 7495 IC. This is an MSI shift register already completely wired internally and ready to use. The flexibility of the input output leads permits this device to be used for a wide variety of functions. It eliminates the need to interconnect individual flip-flops to perform shift register operations. As before you will use the LED indicators to observe the shift register contents. Data switches SW1 through SW4 will be used as a parallel data source for the shift register. Logic switch A will be used to generate the shift pulses in the previous steps. Logic switch B will serve as a mode control for the circuit. Don't forget to connect +5 volts and ground to the 7495 IC.

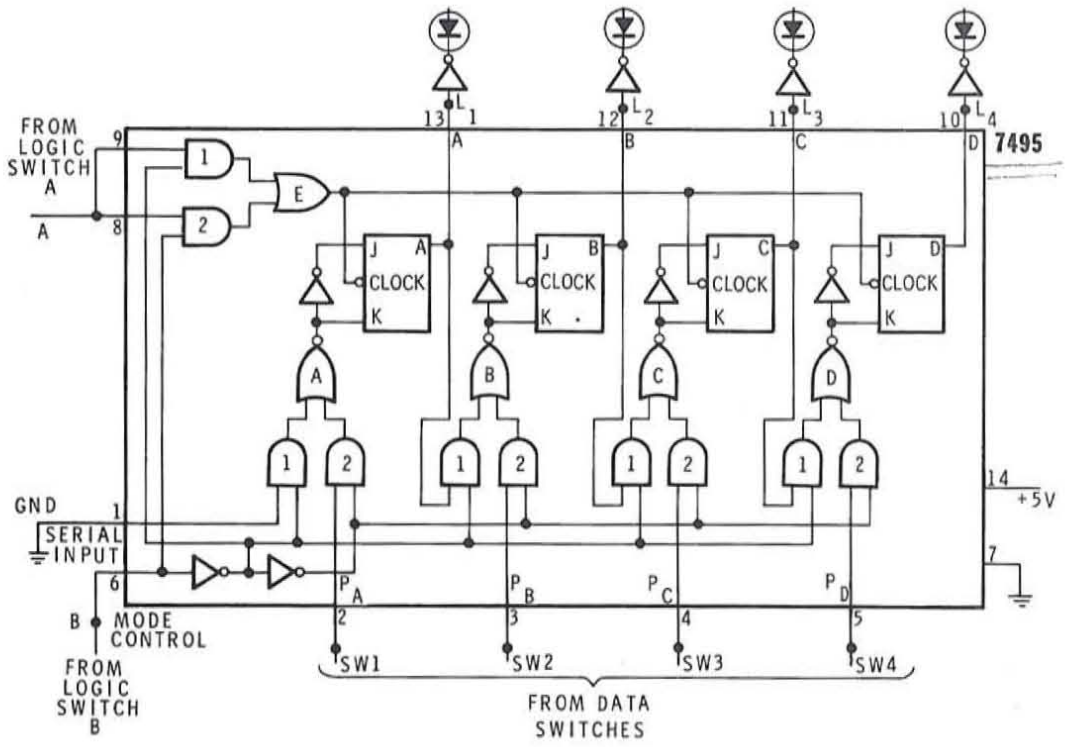


Figure 7-43 7495 MSI shift register circuit for Steps 9 and 10.

5. Apply power to the circuit. As before, the flip-flops in the shift register can come up in to any state. Disregard the LED indicators states at this time. Set all of the data switches to binary 0. Then depress the B logic switch and hold it in the down position. Then momentarily depress the A logic switch. Note the LED indicator states and record below.

   DCBA = 0000

   Next, set all of the data switches to binary 1. Again depress the B logic switch. Holding it in the low position, actuate the A logic switch momentarily. Release both switches. Observe the LED indicator states and record the binary number below.

   DCBA = 1111

   Depress the A logic switch four times and note the LED indicator states. After you have applied four shift pulses, record the LED indicator states below.

   DCBA = 0000

Discussion of Steps 4 and 5

In Step 5 you demonstrated how a shift register can be preset by parallel loading it from some data source. In this case the data source was the data switches SW1 through SW4. First, you set the data switches to binary 0. You then loaded the binary number 0000 into the register. You did this by setting the mode control high by depressing the B logic switch. Then you generated a single clock pulse by depressing the A logic switch. The A logic switch generates a clock pulse that causes the parallel input to be preset into the register.

The 7495 shift register has a mode control input line terminated at pin 6. When this mode control input line is binary 0, the shift register is set up for shift right operations. When the mode control input line is binary 1, the shift right function is disabled and the circuitry for parallel loading the shift register from an external source is enabled. What you did when you depressed the B logic switch is to set the mode control line high. Then you actuated the A logic switch to generate the clock pulse that actually loads the register.

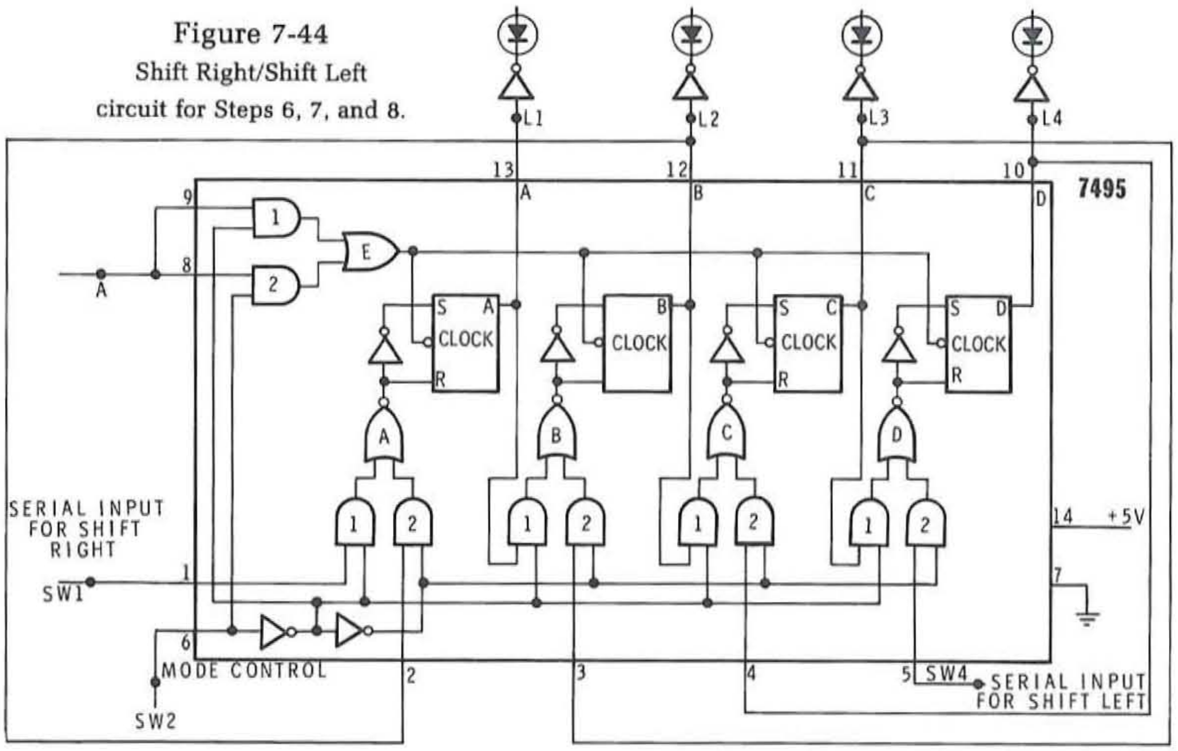
Next, you set the data switches to binary 1. Again, using the A and B logic switches you preset the register. The number in the register at this time should be 1111. Finally, you applied four shift pulses with the A logic switch. This caused the binary 1111 to be shifted out serially to the right. After the four shift pulses were applied, the register contents should have been 0000 since the serial input (pin 1) is low. In this step then you demonstrated how the shift register can be parallel loaded and how this parallel data can then be shifted out serially. This demonstrates the parallel to serial data conversion process performed by a shift register.

Procedure (continued)

6. Modify your 7495 shift register circuit to conform to the configuration shown in Figure 7-44. Here you are connecting the parallel data inputs back around to the outputs in order to permit the shift register to perform shift left operations. As before you will use the A logic switch to generate shift pulses. Data switch SW1 will be used for a serial data input for shift right operations. SW4 will be used as the data source for shift left operations. Switch SW2 will be used to control the mode of the circuit. The mode control will select either shift right or shift left operations.

   Studying the circuit in Figure 7-44, determine the binary state of the mode control input to perform shift left operations. ______________.

Figure 7-44
Shift Right/Shift Left circuit for Steps 6, 7, and 8.



7. Set data switches SW1, SW2, and SW4 to the binary 0 state. Apply power to the circuit and depress the A logic switch four times. Record the state of the LED indicators below.

ABCD = 000

Next, set switch SW1 to the binary 1 position. Depress the A logic switch four times. Note the direction of shifting as the shift pulses are → applied. After four pulses have been applied, record the state of the register below.

ABCD = 1111

8. Set SW1 to binary 0. Again depress the A logic switch four times. → Note the direction in which the data shifts. Next, set SW2 to binary 1, and the SW4 switch to binary 1. Apply four shift pulses with the A logic switch. Again note the direction of shifting, and record the ← contents of the register below.

ABCD = 1111

Set the SW4 switch to binary 0 and apply two shift pulses. Note the direction of the shifting and record the LED indicator states in the space below.

ABCD = 1100

Discussion of Steps 6, 7, and 8

In these steps you demonstrated how the 7495 IC shift register can perform both shift right and shift left operations. In Step 7 you shifted data to the right using SW1 as the serial input data source. First, you shifted in binary 0s to clear the register then shifted in binary 1s. You then shifted out the binary 1s as binary 0s were shifted in. During these shifting operations you should have noted that the data moves from left to right. The logic indicators have been wired so that they indicate the direction of shift directly. Data moves from flip-flop A to flip-flop B to flip-flop C to D or from logic indicator L1 to L2 to L3 and finally to L4. All this occurs with the mode control in the binary 0 position.

When the SW2 switch is placed in the binary 1 position, the register will be set up for shift left operations. The 7495 IC is internally wired to perform shift right operations automatically. By simply enabling the mode control with a binary 0 the shift right function is performed. However, the shift left operation must be externally wired if it is to occur. The shift left operation is implemented by connecting the appropriate flip-flop outputs back to the parallel input lines. The parallel input lines permit the 7495 to be either preset from some parallel force or connected for shift left operations. Note in Figure 7-44 that the D flip-flop is connected to the C flip-flop input. The C flip-flop output is connected to the B flip-flop input. And finally, the B flip-flop output is connected to the A flip-flop input. The D flip-flop receives its input from data switch SW4. This is the serial input line for the shift register when used for shift left operations. Now as shift pulses are applied, data will move from SW4 to the D flip-flop then to C, B, and then A. You demonstrated the shift left effect by loading all binary 1s into the register. As you did you should have seen the LEDs light from right to left indicating a left shift. You then applied two shift pulses with the SW4 switch set to binary 0. This causes binary 0s to be loaded into the C and D flip-flops. The binary 1s stored in those two flip-flops previously are shifted to the A and B flip-flops. Therefore, the binary number stored in the register after the two pulses are applied will be 1100.

This completes your work for Experiment 15. Do not disassemble your shift register circuit as it will be used in the next experiment.

SHIFT REGISTER APPLICATIONS

The shift register is basically a storage element for a binary word. Data can be conveniently shifted into and out of the register in serial form. Despite its simplicity, the shift register has many applications. In this section we are going to look at some of the more popular uses for shift registers.

Serial to Parallel Conversion. One of the most common applications of a shift register is in serial to parallel or parallel to serial data conversions. There are many occasions in digital systems where it is necessary to convert an existing parallel word into a serial pulse train. The shift register can readily perform both of these operations.

Figure 7-45 shows how a shift register is used in serial to parallel and parallel to serial data conversions. In Figure 7-45A the shift register is shown being loaded by a parallel input. The number 1101 is preset into the shift register. Then four clock pulses are applied so that the data is shifted out serially. In Figure 7-45B, the shift register is used for serial to parallel conversion. Here the serial input number 1001 is shifted into the shift register by four clock pulses. Once the data is in the register, the outputs of the individual flip-flops may be monitored simultaneously to obtain the parallel output data.

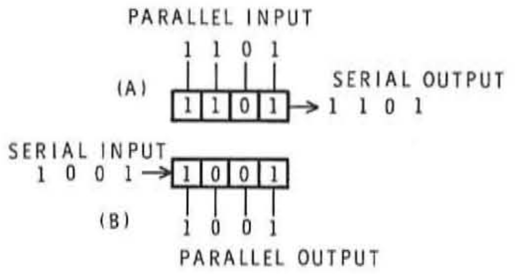


Figure 7-45 Parallel to serial and serial to parallel data conversions with a shift register.

Scaling Operations. A shift register can be used to perform arithmetic operations such as multiplication and division. Shifting the binary number stored in a shift register to the left has the effect of multiplying that number by some power of 2. Shifting the data to the right has the effect of dividing the number in the register by some power of 2. Shifting operations are a simple and inexpensive way of performing multiplication and division with binary numbers.

Figure 7-46A shows a shift register containing a binary number. Assuming the binary point is located to the far right, we can then convert the binary number in to its decimal value. In the initial condition state this number is 3. Now if we perform a shift left operation and move the binary word one position to the left you can see immediately from Figure 7-46B that a new binary number has been formed. As we shift the data to the left one bit position binary 0s are entered on the right. The new number stored in the register is 6. As you can see, shifting the data one position to the left has the effect of multiplying the original number by 2.

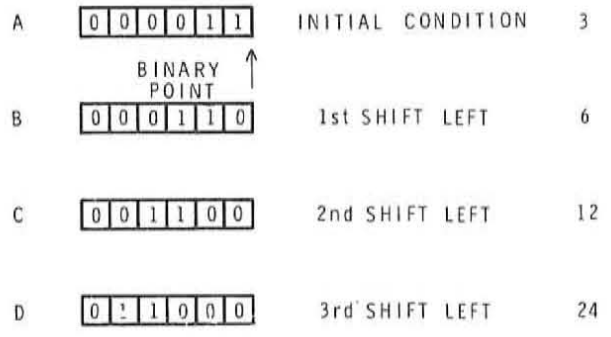
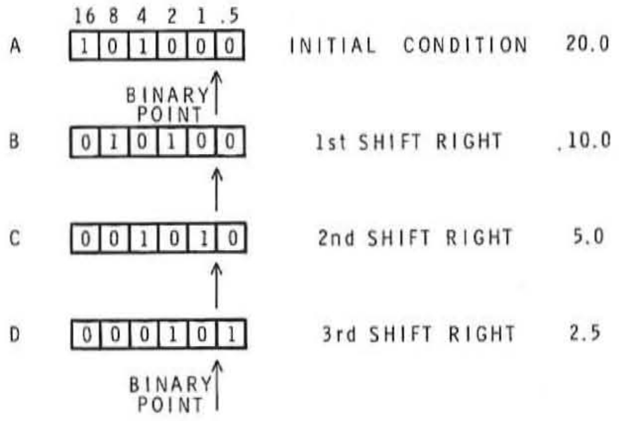


Figure 7-46 Multiplication by factors of 2 by shifting left.

If we perform another shift left operation the number in the register becomes as shown in Figure 7-46C. Converting it to decimal we see that it is 12. The additional shift left operation has again multiplied the number in the register by 2. Two shift left operations have caused the initial number to be multiplied by four.

A third shift left operation will further verify this effect. The number shown in Figure 7-46D is now 24. Shifting the word one additional position to the left has multiplied the number previously by 2. With three shift left operations the initial number in the register has been multiplied by eight. As you can see then, the factor by which the number in the register is multiplied is some power of 2. The multiplying factor is 2^N where N is the number of shift left operations that take place. With three shift left operations as in Figure 7-46, the original number is multiplied by $2^3 = 8$. The important thing to remember about this operation is the shift register must be large enough to accommodate the largest number expected by multiplication. In addition, note that binary 0s are shifted into the right most position as the data is moved to the left.

Division by some power of 2 is accomplished by shifting right. This is illustrated in Figure 7-47. Here the number initially stored in the register is 20.0. Note in this 6 bit register the binary point has been specified as being between the right most bit position and the next most significant bit.



16 8 4 2 1 .5
A 1 0 1 0 0 0 INITIAL CONDITION 20.0
BINARY POINT
B 0 1 0 1 0 0 1st SHIFT RIGHT 10.0
C 0 0 1 0 1 0 2nd SHIFT RIGHT 5.0
D 0 0 0 1 0 1 3rd SHIFT RIGHT 2.5
BINARY POINT

Figure 7-47 Division by powers of 2 by shifting right.

Applying one clock pulse causes the data in the register to be shifted one position to the right. Evaluating the new decimal value of this number we find that it is 10.0. The number initially stored in the register has been divided by 2. Applying another shift pulse causes the data to be moved one more position to the right. Evaluating this number we find that it is 5.0. Again the number has been divided by 2 while the overall division accomplished by two shift pulses is 4. Performing a third shift right operation moves the data again one bit to the right. Evaluating the new number we find it to be 2.5. Again the data has been divided by 2.

The value by which the number in the register is divided is some power of 2. The divide ratio is 2^N where N is the number of shift right operations. Here the original number 20 was divided by $2^N = 2^3 = 8$ or $20 \div 8 = 2.5$. Again an important consideration is that the register be large enough to accommodate the numbers resulting from the scaling operations by shifting. If the register is not large enough, data will be shifted out of the register and lost thereby making the arithmetic operation incorrect.

Shift Register Memory. Shift registers, because they store binary data, are often used for temporary memories in digital equipment. Such a shift register memory is usually capable of storing at least one binary word. Many such memories are made long enough to store many binary words. In such an application, there are two operations that the memory must perform. First, it must be able to accept data and then store it. In other words, we must be able to write new data into the memory. Second, we must be able to retrieve that data or read it out upon command. One of the requirements of the memory is that when we do read the data out that it will not be lost. A shift register can accomplish both of these operations by providing external logic circuitry as shown in Figure 7-48. Here an eight bit shift register is used to store a single binary word. The external control gates are used to select a read or write operation. To store or write data into the memory, the write/recirculate line is set to the binary 1 state. This causes gate 1 to be enabled. Serial data applied to the other input of gate 1 is passed on into the shift register. Once data is stored in the memory, the write/recirculate control line is set to the binary 0 condition. This inhibits gate 1 and prevents other data appearing at the input from being recognized by the shift register. Instead gate 2 is now enabled. Note that the shift register output is connected to gate 2. As shift pulses are applied, the data in the register is shifted out serially and may be used by some external circuit. As the data is being shifted out it is also being shifted back into the input of the shift register through gates 2 and 3. In other words, we are recirculating the data in the register. The read out operation is accomplished and at the same time the data is restored. When we wish to write a new word into the shift register, the write/recirculate line is again made binary 1 and 8 shift pulses are applied.

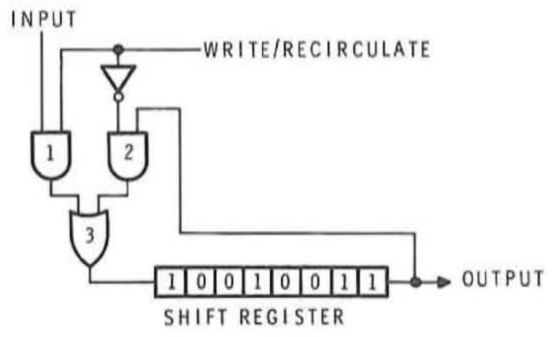


Figure 7-48
Shift register memory.

Sequencer/Ring Counter. Another popular application of the shift register is a sequencer or ring counter. Many logic circuits require a sequence of equally spaced timing pulses for initiating a series of operations. A properly connected shift register can be used to serve this purpose.

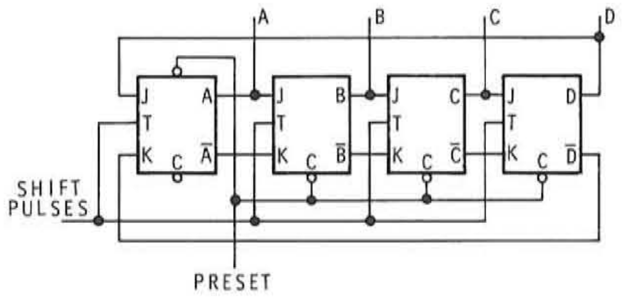
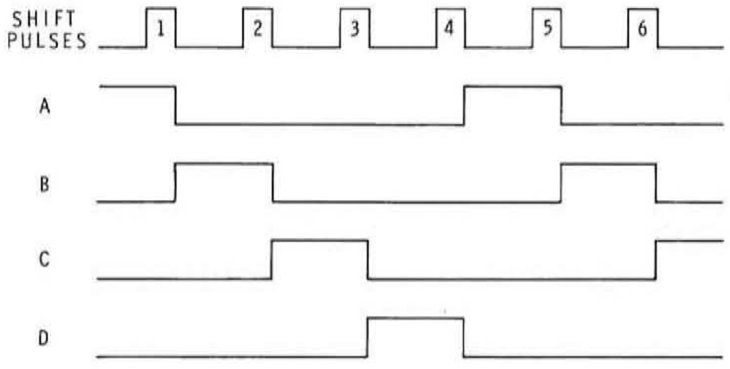


Figure 7-49
A shift register connected
as a sequencer or ring counter.

A shift register connected as a ring counter is shown in Figure 7-49. This is the standard shift register circuit we discussed earlier. However, note that the outputs of the shift register from the D flip-flop are connected back to the JK inputs of the A flip-flop. This provides feedback that causes the shift register to continue to rotate or sequence the data in the register. The asynchronous set and clear inputs of the flip-flop are used to preset a single bit in the shift register. A binary 0 level applied to the preset line causes the A flip-flop to become set and the other three flip-flops to become reset. As shift pulses are applied, the binary 1 in the A flip-flop is shifted to the B, C and D flip-flops, and then back around to the A flip-flop. This sequence repeats as long as the shift pulses are applied. Since the one bit initially programmed into the shift register is continually recirculated, the name ring counter definitely applies. The waveforms and the state table in Figure 7-50 show the operation of the four bit shift register as a ring counter.



STATE	A B C D
0	1 0 0 0
1	0 1 0 0
2	0 0 1 0
3	0 0 0 1

RECYCLE

Figure 7-50 Waveforms for a 4 bit shift register ring counter. State table for ring counter.

A close look at the waveforms for the four bit ring counter in Figure 7-50 shows that the output at any flip-flop has a frequency one fourth that of the shift pulse frequency. In other words, the shift register connected as a ring counter produces frequency division by four since four shift input pulses occur for each flip-flop output pulse. As you can see then the shift register when connected as a ring counter can be used as a frequency divider. By presetting one of the flip-flops in the shift register, frequency division by any integer value can be accomplished by simply using as many flip-flops as needed. For example, to divide by seven, seven flip-flops would be required in the ring counter.

One of the disadvantages of the ring counter circuit shown in Figure 7-49 is that it must be initially preset in order for it to function properly. When power is initially applied to this circuit, the flip-flops in the register can come up in any state. If any random state is allowed in the shift register, the operation previously described will not occur. The preset operation must take place to initially load one of the flip-flops with a binary 1 and the others with binary 0. This disadvantage can be overcome by using a self correcting circuit as shown in Figure 7-51. Here the NAND gate monitors the outputs of flip-flops A, B, and C. The output of the NAND gate and its complement are connected to the JK inputs of the A flip-flop. With this circuit arrangement, any of the sixteen possible states can occur in the shift register and the shift register will automatically correct itself to where only one of the flip-flops is set and the others are reset. Regardless of the initial states of the flip-flops, after a maximum of two shift pulses, the contents of the shift register will automatically be corrected so that only a single flip-flop is set. From that point on the shift register will simply recirculate the single one bit stored there.

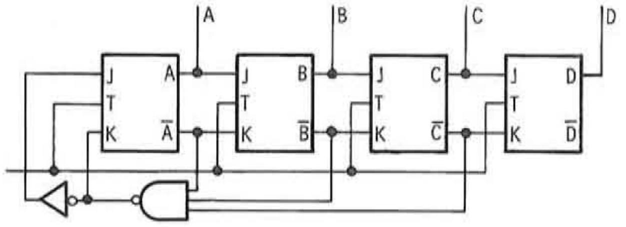


Figure 7-51 Self correcting shift register ring counter.

To use the ring counter described here as a sequencer, the output pulses from the flip-flops are simply connected to the logic circuits whose sequence is to be controlled. Since the shift pulses are normally derived from a fixed frequency clock, the timing interval of the shift registers is precise and thereby permitting very exacting control of the external logic circuits. In addition, it is not necessary to use all of the pulses derived by the shift register. Only those required by the circuit need be used. Keep in mind that as more circuits must be controlled or sequenced, additional flip-flops can be added to the shift register to produce them.

Counters. When connected as a ring counter, shift registers can also be used as a counter. In some special applications they may replace binary counters for certain operations. The four bit ring counter circuit described previously has four distinct states and these states repeat or recycle as the clock pulses are applied.

A popular type of shift register counter is the Johnson counter shown in Figure 7-52. While any number of flip-flops may be cascaded to form a Johnson counter, a five bit circuit is often used. Note that like in the ring counter, the output of the last flip-flop is connected back to the inputs of the first flip-flop in order to recirculate the data. However, note that in this case the normal output of the E flip-flop is connected to the K input and the complement output is connected to the J input. Because of this connection the Johnson counter is often referred to as a twisted ring counter or switch tail counter. With this arrangement, the counter or shift register will have 2N different states where N is the number of flip-flops in the shift register. The five bit Johnson counter shown in Figure 7-52 therefore, will have ten discrete states.

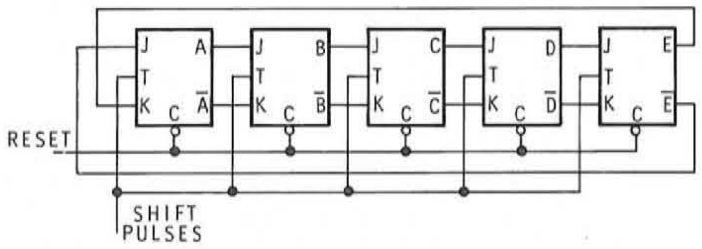


Figure 7-52 Shift register connected as a Johnson counter.

STATE	A	B	C	D	E
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
5	1	1	1	1	1
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	1	1
9	0	0	0	0	1

RECYCLE

Figure 7-53
State table for Johnson counter.

Like the ring counter shift register discussed previously, it is necessary to initialize the counter after power is applied in order to have the counter operate properly. A self correcting circuit similar to that shown in Figure 7-51 can be used to initialize the circuit. Otherwise initialization can be accomplished by simply resetting all of the flip-flops to zero. When shift pulses are applied, the binary state sequences shown in the table of Figure 7-53 are generated. Note the ten individual states. The counter recycles every tenth input pulse. Since the Johnson counter has ten individual states, it is often used as a divide by 10 frequency divider. Other counting operations can often be implemented with this type of counter. However, because of the nonweighted codes generated by the Johnson counter and the ring counter shift register, many counting applications are difficult and inconvenient to implement.

Self Test Review

33. A binary number is shifted 5 positions to the left. The number therefore has been
    a. multiplied by 5
    b. divided by 5
    c. multiplied by 32
    d. divided by 32
34. A binary number must be divided by 128. How many positions must the number be shifted (and in what direction) to achieve this?
35. How many flip-flops must be used in a ring counter shift register to perform frequency division by 12?
    a. 4
    b. 6
    c. 12
    d. 24
36. How many states does a Johnson counter with 8 flip-flops have?
    a. 8
    b. 16
    c. 32
    d. 256
37. Shift registers can operate in both serial and parallel modes. List all four possible combinations of these modes.
    a. ____________
    b. ____________
    c. ____________
    d. ____________
38. In our discussion of using a shift register to multiply and divide by shifting, we assumed that the right most bit was the LSB. What happens to our rules about multiplication and division by shifting if the left most bit is made the LSB?

Answers

33. c. multiplied by 32.
34. 7 positions to the right.
35. c. 12
36. b. 16
37. a. Serial In-Serial Out (SI-SO)
    b. Parallel In-Parallel Out (PI-PO)
    c. Serial In-Parallel Out (SI-PO)
    d. Parallel In-Serial Out (PI-SO)
38. The rules are reversed. A right shift now becomes a multiply and a left shift becomes a divide.

EXPERIMENT 16

SHIFT REGISTER APPLICATIONS

OBJECTIVES: To demonstrate several practical applications of integrated circuit shift registers.

Materials Required

Heathkit Digital Design Experimenter ET-3200

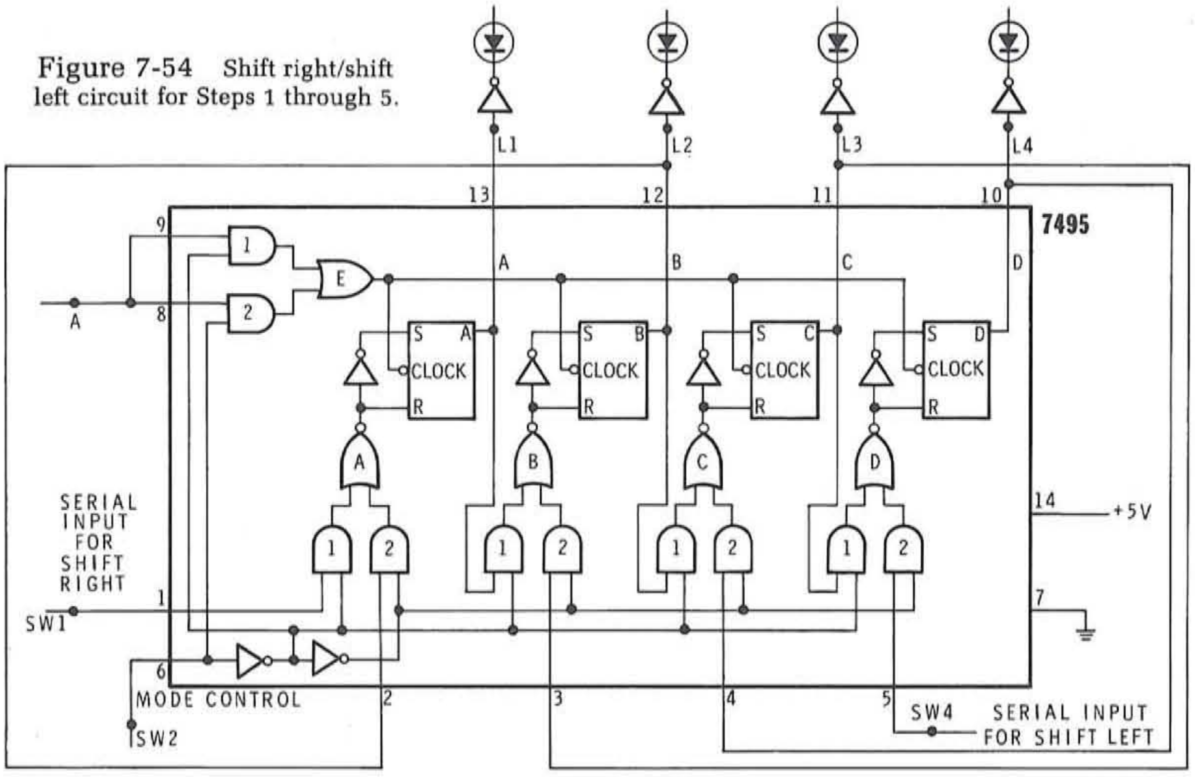
1 — 7404 IC (443-18)

1 — 7495 IC (443-680)

Procedure

1. For this experiment you will use the 7495 shift register circuit you constructed in Experiment 15. As you recall the register was wired for both shift right and shift left operations. To perform the next step you do not need to make any further modifications to the circuit. The circuit is repeated in Figure 7-54.

Figure 7-54 Shift right/shift left circuit for Steps 1 through 5.



2. Set the SW2 switch to binary 1. SW1 should be set to binary 0. Set SW4 to binary 0 and depress the A logic switch 4 times. The register is set up for shift left operations and this step should clear the register to 0.

   Next set SW4 to binary 1. Depress the A logic switch two times. Record the binary number stored in the register. Assume that the D flip-flop (indicator L4) is the LSB.

   ABCD = 0011,
   Decimal value = 3

   Now set SW4 to binary 0.

   Depress the A logic switch once and again note the contents of the register. Record it and its decimal equivalent below. ABCD = 0110, decimal value = 6. Again depress the A logic switch a single time. Record the binary and decimal equivalent of the register contents.
   ABCD = 1100, decimal value = 12.

3. Study the data you obtained in Step 2 above. Determine the relationship between the numbers obtained when the register was loaded and as it was shifted to the left. What mathematical operation was performed by the shift left operation? X 2.

4. Set SW2 and SW4 to binary 0. The SW1 switch should also be at 0 at this time. Clear the register by pressing the A logic switch 4 times. Next, load the register by following the step by step instructions below.

   SW1 = 1, Depress A logic switch
   SW1 = 0, Depress A logic switch
   SW1 = 1, Depress the A logic switch
   Set SW1 = 0

   Record the binary contents of the register and its decimal value below. Again use the D flip-flop (indicator L4) as the LSB.
   ABCD = 1010, decimal value = 10.
   Depress the A logic switch once. Note the binary value of the register contents and record it in its decimal equivalent below.
   ABCD = 0101, decimal value = 5.

5. Study the data that you obtained by the shift right operations you carried out in Step 4. What mathematical operation is performed when a shift right operation occurs? ÷ 2.

Discussion of Steps 1 through 5

In these steps you demonstrated how a shift right/shift left circuit could be used to perform multiplication and division operations on binary numbers. In Step 2 you loaded the binary number 3 into the shift register. Then you shifted it to the left one step. Evaluating the number in the register you saw that it was 6. Shifting the number one more bit position to the left produced the binary number 12. Your conclusion from this is that with each shift left operation the number stored in the register was multiplied by 2. Here, you shifted the number 3 two positions to the left producing a total multiplication factor of $2^2 = 4$.

Next, you set up the shift register for producing a shift right operation. You initially loaded the binary number 1010 into the register. Of course, this is the binary equivalent of the decimal number 10. You then applied a single shift pulse and caused the number to be shifted to the right. Evaluating the new number you found it to be 0101 or 5. Here shifting a number to the right causes that number to be divided by two for each shift right. The original number in the register is divided by a number equal to 2^N were N is the number of positions shifted to the right.

A shift right/shift left shift register is easy to use for scaling operations involving the multiplication or division of number by some power of 2.

Procedure (continued)

6. Next, you are going to demonstrate how the data in a shift register can be recirculated by feeding the serial output back to the serial input. This permits the data to be shifted out serially for use in some external source but the data will still be retained since it is recirculated.

   Modify your 7495 shift register circuit so that it appears as shown in Figure 7-55. Note that the output from the D flip-flop is fed back to the serial input of the shift register at pin 1. You will use the data switch SW1 as the mode control and SW4 will serve as an additional serial input. The logic control gating at the input to the A flip-flop permits either of two data sources to be shifted into the register, that from pin 1 and that from pin 2.

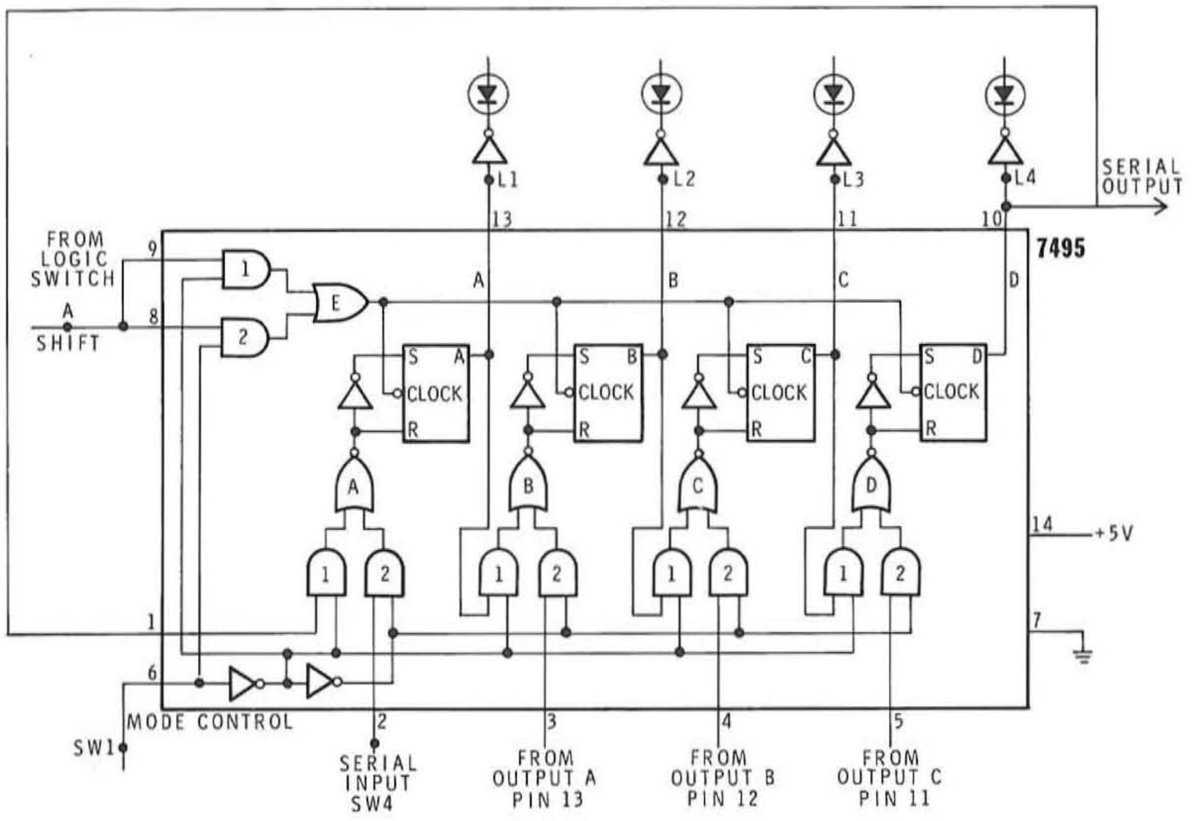


Figure 7-55
Write/recirculate shift register circuit for Steps 6 through 9.

7. Set SW1 to binary 1 and SW4 to binary 0. Depress the A logic switch four times. This should clear the register and the other indicators should be off.

Next, set SW4 to binary 1. Depress the A logic switch two times. In the space provided below, record the binary number stored in the register.

ABCD = 1100

Next, set SW1 to binary 0. Then depress the A logic switch four times. For each shift pulse, note the position of the binary 1 bits on the LED display. Record the state of the register contents after four shift pulses have been applied.

ABCD = 1100

In Table I, record the state of the shift register contents as indicated by the LED indicators. Then after each clock pulse, again record the four bit register state. Complete the table as indicated.

TABLE I

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

8. Study the information you recorded in Table I. From this information determine the operation of the circuit. Refer back to Figure 7-55 if necessary to see how the circuit operates. Is the data in the shift register lost or retained as a result of shifting the data? retained.

9. Set the SW1 mode control switch to binary 1. Set SW4 to binary 0. Again apply four shift pulses with the A logic switch. Note the contents of the register after the four shift pulses have been applied and record the register state below.

ABCD = 0000

As a result of the operation above, was data lost or retained as a result of the four shift pulses? Lost.

Discussion of Steps 6 through 9

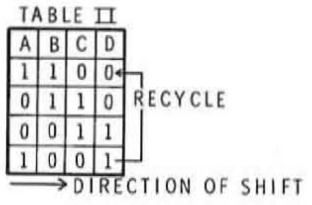
In these steps you demonstrated how data can be recirculated in the shift register in order that the contents be retained even when the data must be shifted out serially to another source. This is done by connecting the output of the shift register from the D flip-flop back around to the serial input to the A flip-flop at pin 1. With the mode control set to the binary 0 position, the shift register will perform a shift right operation. As the shift right operation is performed, the serial data appears a bit at a time at the normal output of the D flip-flop. But at the same time, this data is shifted back into the shift register. It takes four shift pulses to cause a single four bit binary word to be shifted out. After four shift pulses occur, the data is also shifted back into the register and is ready to be used again.

When the mode control switch is set to binary 1, the serial data input at pin 1 on the 7495 IC is disabled. In this case, the input at pin 2 is recognized. This permits an external serial data source to feed data to the shift register. In this case you used SW4 to provide data to the shift register. With the mode control input at binary 1, you can load a serial word appearing at pin 2 into the register as shift pulses are applied.

An important point to note is that with the mode control in the binary 1 state the parallel data inputs are enabled. The input to flip-flop A is used as the serial data source. But the other inputs must be connected to flip-flops A, B and C respectively in order for a shift right operation to be performed with the mode control input high.

In Step 7, you set the mode control to the binary 1 position and the SW4 switch to binary 0. This caused data (0000) to be loaded into the shift register from SW4. Next, you loaded two binary 1s by setting SW4 to binary 1 and depressing the A logic switch twice. This loaded the number 1100 into the register.

TABLE II


RECYCLE
DIRECTION OF SHIFT

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

Next, you set the mode control input to binary 0. This disables the serial input from SW4 and causes the data to recirculate. As you applied four shift pulses, you should have observed the data shifting right out of the D flip-flop and then back around into the A flip-flop. After four shift pulses, the data is shifted out of the register but it is also recirculated. After four shift pulses, the contents of the register is still 1100. Your data in Table I should appear as shown in Table II.

Finally, you set the mode control switch to binary 1. This again enables the serial input from SW4. You then loaded binary 0s with four shift pulses. As you loaded these 0s you noticed that the binary number 1100 was shifted out serially and lost as the new number 0000 was shifted in.

A shift register when connected in this way forms what is known as a load/recirculate register. The mode control input lines lets you put in data from an external serial source. In the recirculate mode it permits data to be shifted out and used externally but also recirculates it so that it is retained for another operation.

Procedure (continued)

10. Using the recirculate register shown in Figure 7-55, clear the register by loading all 0's. If the register is already at 0 then this operation is not necessary. To clear the register, set SW1 to binary 1 and SW4 to binary 0. Then depress the A logic switch 4 times.
11. Next, set SW4 to binary 1. Depress the A logic switch once. Set the mode control switch SW1 to binary 0. Then begin depressing the A logic switch. Note the result. Continue depressing the A logic switch a number of times until you are fully aware of what the circuit is doing.

Discussion of Steps 10 and 11

In these steps you demonstrated the operation of the shift register as a ring counter. You cleared the register and loaded a binary 1 into the A flip-flop. You then set the mode control so that the shift register would recirculate. Then by depressing the A logic switch you were able to cause the binary 1 bit to move from one flip-flop position to the next and then recirculate. When used in this manner, the shift register is known as a ring counter. This ring counter makes an excellent sequencing circuit for driving digital circuits that require a time sequence of pulses.

Procedure (continued)

12. Modify the shift register circuit so that it conforms to that shown in Figure 7-56. The parallel data inputs will not be used in this step. The mode control input line is simply connected to ground. The output from the D flip-flop is connected through one of the inverters in a 7404 IC and then fed back around to the input of the shift register.

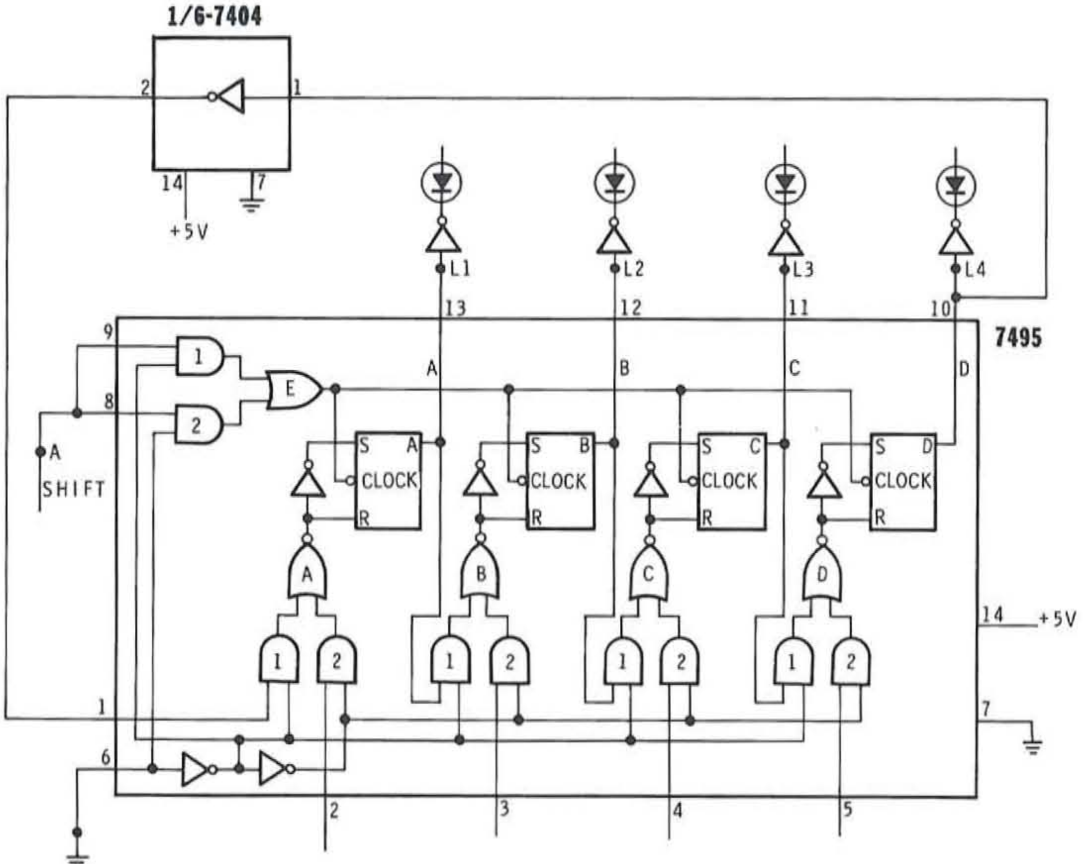


Figure 7-56 Shift register circuit for Steps 12 through 15.

13. Study the circuit in Figure 7-56 and answer the following questions.

    The circuit is set up to perform a shift (right, left) ______________ operation.

    The shift register circuit connected in this way is known as a ______________ counter.

14. Apply power to the circuit. Depress the A shift input switch until the shift register contains all 0's. Record this state in the first position of Table III. Next, depress the A logic switch and after each actuation record the shift register state in the sequential locations in Table III. Continue depressing the A logic switch one at a time and recording the register states until Table III is complete.

15. Disconnect the shift clock input lines at pins 8 and 9 on the 7495 IC from the A logic switch output and connect them to the clock (CLK) output. Set the clock frequency to 1 Hz. Apply power to the circuit and observe the shift register output states. As soon as all the LED indicators show the register content to be 0000, monitor the shift register states after each clock pulse and verify them against the data you collected and recorded in Table III. Continue to let the shift register circuit operate while observing Table III. Be sure that you let the circuit run long enough so that you understand the operation that is taking place.

TABLE III

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

← Read

Discussion of Steps 12 through 15

The circuit you constructed in Step 10 is a Johnson counter. Since the mode control input is set to binary 0 by grounding it, the circuit that will perform a shift right operation. The circuit connection feeds the normal output of the D flip-flop through an inverter and applies the complement back around to the serial input of the shift register. This is the equivalent of connecting the normal and complement output of the shift register output flip-flop back around to the K and J inputs of the input flip-flop on a shift register as indicated previously in the unit. When we do this we form a switch tail or Johnson counter. Since the normal and complement flip-flop outputs of the JK inputs are not available, the arrangement in Figure 7-56 accomplishes the same effect. Essentially we invert the output of the shift register and feed it back to the serial input. The result is a four bit Johnson counter. Such a counter has 2 N discrete states where N is the number of flip-flops. Since we are using four flip-flops this circuit should have 8 discrete states. You verified this by stepping the counter and recording the flip-flop states in Table III. You then verified this operation by letting the counter operate automatically from the 1 Hz clock pulse. You should have found the shift register states to be like those indicated in Table IV.

TABLE IV

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

RECYCLE

MOS SHIFT REGISTERS

In applications requiring shift registers with a limited bit capacity, bipolar integrated circuits such as TTL or ECL are used. CMOS shift registers are also available. Such registers are generally used for storing a single binary word. This word may be as small as four bits but could be as long as 32 bits in some applications. Where more data storage is needed, additional flip-flops or MSI shift registers can be cascaded to form memories which can be used for storing many binary words. When implemented with standard bipolar and CMOS integrated circuits, such shift register memories become very large and expensive. In these applications MOS shift registers can be of value.

MOS integrated circuit shift registers using P or N channel enhancement mode MOSFETs contain many storage elements. Because of the very high component density and very low power dissipation of the MOS structure, very large shift registers can be made on a very tiny silicon chip. MOS shift registers with thousands of storage elements are available for memory applications. Such shift registers are commonly used to store many binary words in a serial format. For example, to store 128 eight bit binary words we would need an $8 \times 128 = 1024$ bit shift register.

MOS shift registers this large are very practical and are commonly used for temporary data storage and for delay operations. Any application requiring the temporary storage of a large volume of binary data can use MOS shift registers. In addition, most MOS LSI shift registers are of the serial in/serial out type. The parallel loading and readout of data is not generally performed with MOS shift registers. These MSI and LSI circuits provide a very economical memory source.

There are two basic types of MOS shift registers: static and dynamic. A static shift register is one in which the clock may be stopped without loss of data. This is the type of shift register that we have discussed in the previous sections. Clock signals are applied to shift data into or out of the register. When the clock pulses are stopped, the data in the shift register is retained in the storage elements. Data is not lost if the clock is stopped.

In another type of MOS shift register the data will be lost if the clock is stopped. This type of shift register is known as the dynamic type. Because of the characteristics of the storage element used in a dynamic register, the clock pulses must run continuously if data is to be retained. Data must be continuously recirculated or refreshed in order to prevent its loss. Naturally, the static shift register is generally more desirable from a standpoint of operation and convenience. However, static MOS shift registers are generally more complex and consume much more power.

Dynamic shift registers can be made smaller and more simply, can operate at higher speeds and have far lower power dissipations. Such trade offs must be considered when designing with MOS shift registers. Most MOS shift registers are fully compatible with TTL and CMOS circuits. No special interfacing is required.

Dynamic MOS Shift Register. The basic storage element in a MOS shift register, whether it is dynamic or static, is the capacitance that exists between the gate and the channel of the MOSFET transistors used. While this capacitance is very small (on the order of several tenths of a picofarad), the high impedance nature of the MOSFET permits a charge voltage to be placed on this capacitance and retained for a relatively long period of time. The impedance between the gate and the source of an enhancement mode MOSFET is on the order of 10^{15} ohms or greater. Such a high impedance is virtually an open circuit and has a minimum effect on the gate capacitance. If we apply a voltage between the gate and source of a MOSFET, the gate capacitance will charge and remain there until it leaks off through the very high impedance between the source and gate. In high quality MOSFETs, this discharge time can be as long as one millisecond.

The storage element circuit used in a MOS shift register is a MOSFET inverter. The input capacitance of the inverter transistor stores the data. Figure 7-57 shows how two MOSFET inverters (I1 and I2) are combined with MOSFET transmission gates (Q_1 and Q_2) to form a one bit storage element. The input data is applied to inverter I1 through transmission

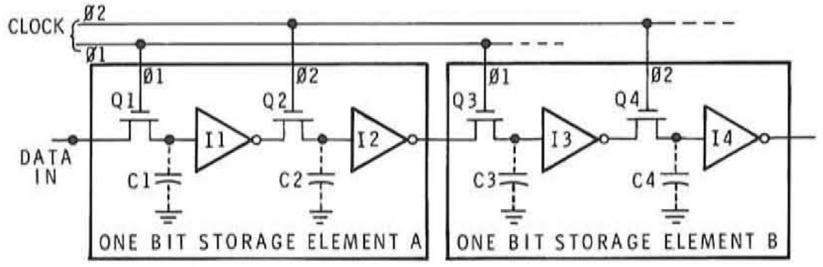


Figure 7-57
Dynamic MOS Shift Register.

gate Q1. MOSFET Q1 is simply used as an on/off switch to connect the input to capacitor C1 and disconnect it. The output of inverter I1 is connected to the input of inverter I2 through transmission gate Q2 which again is used as a simple on/off switch. The switching of the transmission gate transistors is controlled by two clock signals designated phase 1 ($\phi1$) and phase 2 ($\phi2$). These two phase clock signals are illustrated in Figure 7-58. Note that when $\phi1$ is on $\phi2$ is off and vise-versa.

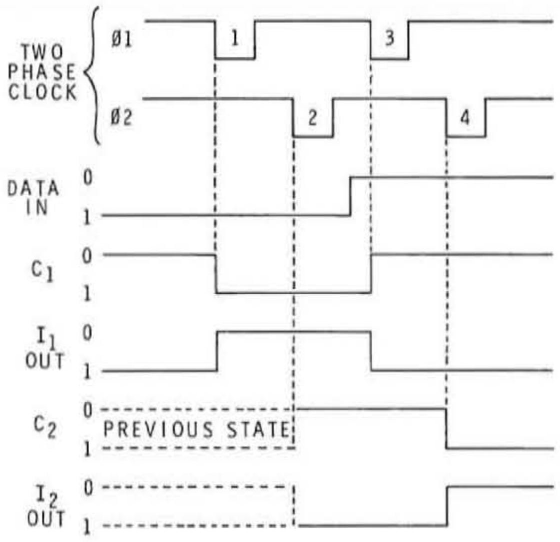


Figure 7-58
Clock and circuit waveforms
for the dynamic MOS shift register.

For our discussion here let's assume the use of P channel MOS circuits where a binary 0 is equal to zero volts or ground and a binary 1 is some negative voltage level.

The data to be stored (written) into storage element A is applied to the data input line. Assume that we apply a binary 1 input which is some negative voltage level. When clock pulse $\phi1$ occurs, transmission gate Q1 will conduct. This will cause capacitor C1 to charge to the input voltage. Applying a binary 1 input voltage to inverter I1 causes a binary 0 level to appear at its output. After the occurrence of the $\phi1$ clock pulse, capacitor C1 retains the charge and acts as the input voltage source for inverter I1.

The $\phi 2$ clock pulse occurs next. This causes transmission gate Q2 to conduct. The state of the output of I1 is, therefore, transferred to capacitor C2. This is a binary 0, so capacitor C2 has zero charge. The input to inverter I2 is a binary 0. The output of I2, therefore, is a binary 1. After one $\phi 1$ clock pulse and one $\phi 2$ clock pulse, the binary 1 that was at the input to storage element A appears at the output of storage element A. On the next cycle of $\phi 1$ and $\phi 2$ clock pulses, this binary 1 value will be transferred to the next storage element (B) of the shift register. Any new data appearing at the input of the first storage element will be shifted in at this time. The waveforms in Figure 7-58 illustrate the storage of a binary 1 in element A and its transfer to element B as a binary 0 is entered.

The inverters in Figure 7-57 can be any one of several different types of MOS logic inverters. Figure 7-59 shows the two types most commonly used. In Figure 7-59A, a static inverter is used. Here Q2 is the inverter element while Q1 is a MOSFET biased into conduction to act as a load resistance. This type of device dissipates power because Q1 is continuously conducting. In long MOS shift registers this power dissipation is additive and can produce a significant amount of heat.

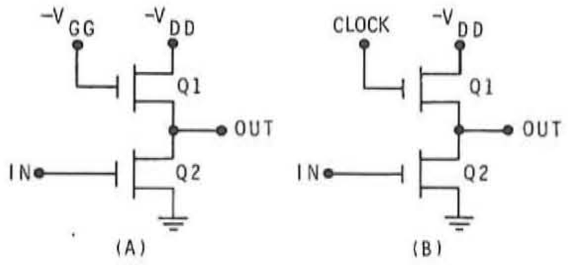


Figure 7-59 MOS inverters (A) static and (B) clocked.

Another type of inverter shown in Figure 7-59B uses a clocked load device. Here, Q2 is the inverter element while Q1 is the load element. Q1, however, does not conduct, except during the time a clock pulse is applied. When such an inverter is used in the dynamic shift register circuit at Figure 7-57, the load element is clocked during $\phi 1$ or $\phi 2$ along with the associated output transmission gate. For example, the load element in I1 would be clocked at $\phi 2$ time while the load element in I2 would be clocked at $\phi 1$ time. This arrangement greatly reduces the power dissipation of the device.

In order to keep the data from being lost during the shifting process, the clock must run continuously and the data must be continually recirculated from output to input. Write/recirculate logic at the input of the shift register is used to select the mode of operation. Should the clock pulses stop, the data stored as charges on the capacitances in the circuits will leak off and be lost. The loss of data can occur in only several hundred microseconds depending upon the circuitry used. For that reason the minimum clock rate is approximately 5 kHz for most typical dynamic MOS shift registers. Dynamic shift registers with minimum clock rates in the 100 Hz range are available.

Static MOS Shift Registers. There are some applications where it is desirable to stop the clock in a digital system. For such applications, static MOS shift registers can be used. Such shift registers employ storage elements that retain the data even after the clock has stopped. In addition, it is not necessary to continually recirculate the data in order to retain it.

A typical static storage element for a static MOS shift register is shown in Figure 7-60. It also uses the gate capacity of a MOSFET inverter for temporary data storage. The storage element uses two such inverters. One inverter is transistor Q3 with its load device Q2. The other inverter is Q7 with its load device Q6. These two inverters are cross coupled through transmission gates Q4 and Q5. When Q4 and Q5 conduct, a latch type flip-flop is formed. Naturally, a latch will retain data even when the clocking signals are removed as long as Q4 and Q5 conduct. Transistor Q1 is used as a transmission gate to load data into the storage element. To explain the operation of the circuit, assume that P channel devices and negative logic are used.

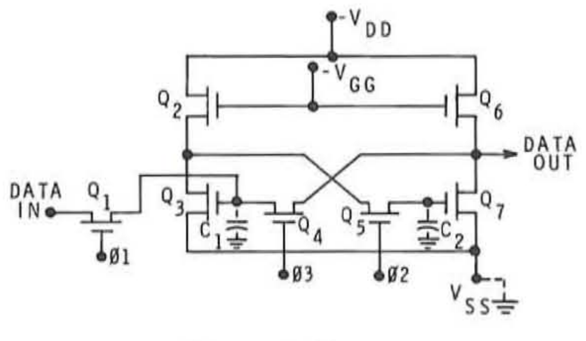


Figure 7-60
One bit MOS storage element for a static MOS shift register.

The proper operation of this shift register requires a three phase clock signal. These clock signals are shown in Figure 7-61. The $\phi 3$ clock is a delayed replica of the $\phi 2$ clock signal. In some static MOS shift registers, the $\phi 3$ and sometimes the $\phi 2$ and $\phi 3$ clock pulses are generated on the chip. Therefore, the circuit requires only a single or double phase external clock for proper operation.

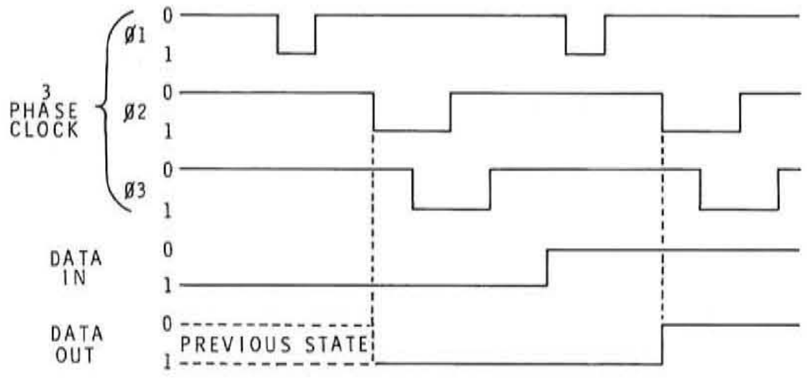


Figure 7-61 Waveforms for static MOS shift register.

To load data into the circuit, the desired bit is applied to the data input line. When the $\phi 1$ clock occurs, transmission gate Q1 conducts. This causes the gate capacitance of Q3 (C1) to charge to the proper state. If a binary 1 is applied to the data input, C1 will assume a negative charge. This negative charge is applied to inverter Q3. Q3 conducts and its drain goes low representing a binary 0.

The $\phi 2$ clock occurs next and Q5 conducts, thereby, transferring the state of Q3 to capacitor C2. In our example, this is a binary 0. The output of inverter Q7 then is as a negative level binary 1. The $\phi 3$ clock signal occurs and Q4 conducts. This applies the negative signal back to the gate of Q3 to keep it on. At this time, the data is latched. The shift register will remain in this state until the state of the input has changed and the next clocking cycle has been completed.

Self Test Review

39. MOS shift registers are which of the following type?
    a. SI-SO
    b. SI-PO
    c. PI-PO
    d. PI-SO
40. The two types of MOS shift registers are ______________ and ______________.
41. Static MOS shift registers consume
    a. more
    b. less

    power than a dynamic register.
42. MOS shift registers are used mainly
    a. for storing a single binary word.
    b. parallel to serial data conversions.
    c. for multiplying and dividing operations.
    d. as a memory for storing many binary words.
43. The main storage element in an MOS register is the ______________ ______________ of a MOSFET inverter.
44. What two requirements are necessary to prevent a data loss in a dynamic MOS shift register?
    a. ______________
    b. ______________

Answers

39. a. SI-SO
40. static, dynamic
41. a. more
42. d. as a memory for many binary words
43. gate capacitance
44. a. continuously running clock
    b. data recirculation

CLOCKS AND ONE SHOTS

Most sequential logic circuits are driven by a clock. The clock is a periodic signal that causes logic circuits to be stepped from one state to the next. The clock steps the sequential circuits through their normal operating states so that they perform the function for which they were designed.

The clock signal is generated by a circuit known as a clock oscillator. Such an oscillator generates rectangular output pulses with a specific frequency, duty cycle and amplitude. The most commonly used clock oscillator is some form of astable multivibrator. Such circuits can be constructed with discrete components or with logic gates.

Another circuit widely used to implement sequential logic operations is the one shot. The one shot or monostable multivibrator produces a fixed duration output pulse each time it receives an input trigger pulse. The duration of the pulse is usually controlled by external components. By cascading one shot circuits, a wide variety of sequential circuits can be implemented.

Clock Oscillator Circuits

Practically all digital clock oscillator circuits use some form of astable multivibrator circuit for generating a periodic pulse waveform. Such a circuit has two unstable states, and the circuit switches repeatedly between these two states. Both discrete component and integrated circuit clocks are used in digital equipment.

Discrete Component Circuits. The most commonly used clock oscillator is the astable multivibrator circuit shown in Figure 7-62. It consists of two transistor inverters Q1 and Q2 with the output of one connected to the input of the other. Resistors R2 and R3 are used to bias the transistors into saturation. Capacitor C1 and C2 couple the output of one inverter to the input of the other. In normal operation, one transistor is conducting while the other is cut off. The frequency of oscillation is determined by the values of R2, R3, C1 and C2.

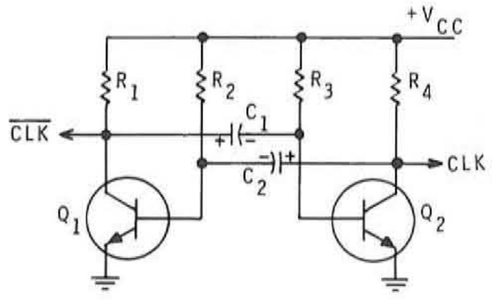


Figure 7-62 Astable multivibrator clock oscillator.

Assume that Q2 is conducting and Q1 is cut off. Capacitor C1 then charges through the emitter-base junction of Q2 and R1 to the supply voltage $+V_{CC}$. Capacitor C2 which was previously charged to the supply voltage with the polarity shown keeps Q1 cut off as it discharges through resistor R2. As soon as it discharges to zero it begins to charge in the opposite direction. When the charge on C2 reaches about 0.7 volt, Q1 conducts. As soon as Q1 conducts, it effectively connects the positive side of C1 to ground. This puts a negative voltage between the base and the emitter of Q2 causing it to switch off quickly. C1 then discharges through R3. At this time C2 is recharged through the emitter-base junction of Q1 and R4. As soon as the charge on C1 has reached zero, it will begin to charge to the supply voltage. However, as soon as the voltage is high enough, Q2 again conducts and the state of the circuit reverses. This cycle continues at a rate determined by the discharge time of C1 and C2 which in turn depends upon the values of R2 and R3. R2 and R3 are generally selected in order to ensure saturation of Q1 and Q2. Capacitors C1 and C2 are then chosen to produce the desired operating frequency with the given base resistors. The frequency of oscillation (f) is approximately equal to:

$$f = \frac{1}{1.4\ RC}$$

This formula assumes that R = R2 = R3 and C = C1 = C2. With this arrangement the circuit will produce a 50 percent duty cycle output square wave. Unequal values of capacitors can be used to produce a duty cycle more or less than 50 percent.

Figure 7-63 shows the circuit outputs. The outputs are taken from the collectors of the two transistors and they are complementary. The outputs switch between the supply voltage $+V_{CC}$ and the V_{CE} (sat) of each transistor. This clock oscillator circuit can drive most standard logic families such as TTL and CMOS directly. For other types of logic, interface circuitry may be required between the clock oscillator and the logic circuitry.

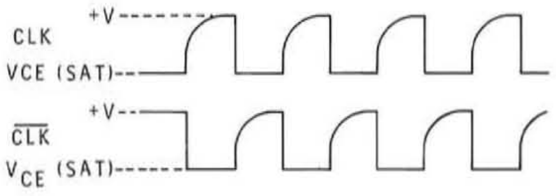


Figure 7-63 Waveforms for the astable multivibrator.

Another form of clock circuit is the relaxation oscillator shown in Figure 7-64A. This circuit uses a programmable unijunction transistor (PUT). The PUT is a four layer semiconductor device that is used as a threshold sensitive switching device. It has three terminals designated the cathode (K), the anode (A), and the gate (G).

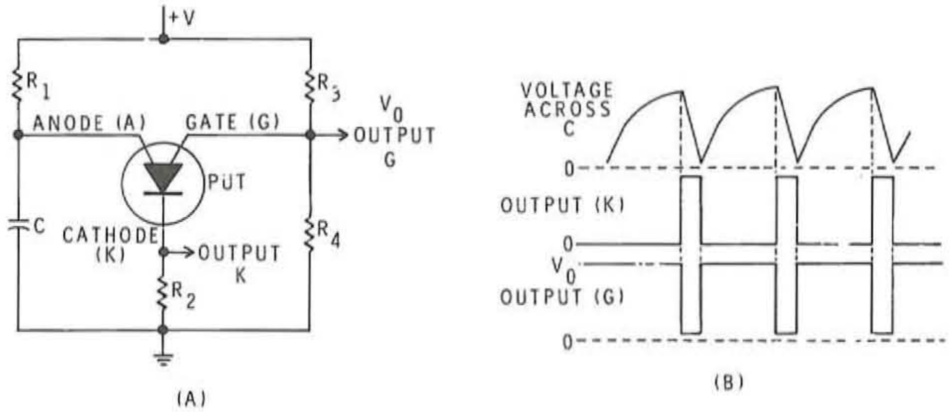


Figure 7-64 (A) Clock oscillator using a PUT. (B) Waveforms for the PUT clock oscillator.

The PUT will conduct current between its cathode and gate when the anode is properly biased. Normally, the device will not conduct if the anode potential is equal to or less than the voltage at the gate. The voltage at the gate is set by external resistors R3 and R4. In most circuits R3 is made equal to R4 making the voltage at the gate approximately one half the supply voltage +V. In operation, capacitor C charges through R1 toward the supply voltage. As soon as the charge on capacitor C is equal to 0.7 volt more positive than the gate voltage, the PUT conducts and current flows between the cathode and the gate. The PUT becomes a very low resistance and capacitor C discharges through it and R2. R2 is a very low value of resistance therefore, the capacitor discharges quickly. As the capacitor discharges it produces a voltage across R2 which is used as the output. The waveforms for the PUT relaxation oscillator are shown in Figure 7-64B. The upper waveform shows the charging voltage across the capacitor C. The remaining waveform shows the output signals from the cathode and from the gate. The duration of the output pulse produced by this circuit is very narrow because of the very rapid discharge time of C when the PUT conducts.

The frequency of oscillation in this circuit is a function of the RC time constant (R_1 C) and the voltage at the gate of the PUT.

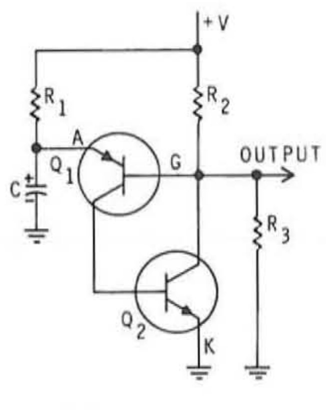


Figure 7-65
NPN-PNP simulation of a PUT in an astable clock oscillator.

The PUT can be simulated by using complementary bipolar transistors as shown in Figure 7-65. Transistors Q1 and Q2 are connected to form a switch that is controlled by the charging capacitor C and the voltage divider R2/R3. The operation of the circuit is identical to the PUT circuit discussed earlier. The emitter of the PNP transistor Q1 is equivalent to the anode of the PUT. The base of the PNP transistor Q1 tied to the collector of Q2 is equivalent to the gate of the PUT. The emitter of the NPN transistor Q2 is equivalent to the cathode of the PUT. The voltage divider made up of R2 and R3 sets the bias voltage on the base of transistor Q1. As in the PUT circuit, the voltage divider resistors are generally made equal so that the voltage at the base of Q1 is approximately one half the supply voltage. Capacitor C then charges through R1 to the supply voltage. When the charge on C exceeds the emitter-base voltage threshold of Q1 (about 0.7 volts), Q1 will conduct. It causes base current to flow in Q2. As a result, Q2 saturates. Capacitor C discharges quickly through Q2 and Q1. As with the PUT circuit, the duration of the output pulse is very short due to the very short discharge time of the capacitor. The output can be taken from the collector of Q2. Alternately, a low value resistance can be connected in the emitter lead of Q2 to develop a positive going output pulse if required. The waveforms for this circuit are similar to those for the PUT circuit discussed earlier.

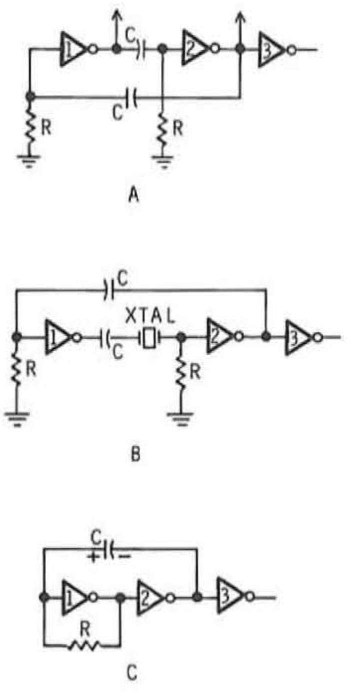


Figure 7-66 Astable multivibrators made with TTL inverters.
(A) Conventional astable,
(B) crystal controlled astable,
(C) simplified astable.

IC Clock Circuits. The astable multivibrator of Figure 7-62 can also be implemented by using integrated circuit gates or inverters. Figure 7-66A shows the astable circuit implemented with TTL inverter circuits. The operation of this circuit is practically identical to the circuit in Figure 7-62. The frequency of oscillation is a function of the values of resistance and capacitance in the circuit. The resistors provide a charge path for the capacitors and are also used to provide bias to the inverter circuits. The frequency of oscillation of this circuit is approximately equal to:

$$f = \frac{1}{2\,RC}$$

Where f is in kHz, R is in k ohms and C is in microfarads. The output of the circuit will be a 50 percent duty cycle square wave. Inverter 3 is used to buffer the circuit output and to isolate the load from the frequency determining components.

The basic IC astable multivibrator can be modified as shown in Figure 7-66B to include a frequency determining crystal. The values of R and C are selected to perform oscillation near the desired frequency. The circuit will oscillate at the crystal frequency. This circuit is desirable when the clock frequency must be very accurate and remain stable. Again inverter 3 is used to buffer the output and isolate the load from the frequency determining components.

Figure 7-66C shows another version of the basic astable multivibrator. This circuit uses only a single RC network. Resistor R is used to bias inverter 1 close to the linear region. In this circuit, the value of R is very critical and should be somewhere in the 150 to 220 ohm range when standard TTL inverters or OR gates are used.

The operation of this astable multivibrator is somewhat different from the discrete component circuit and its integrated circuit counterparts discussed earlier. This is how it operates.

Refer to Figure 7-66C. Assume the output of inverter 2 goes low. This low will be coupled through capacitor C to the input of inverter 1, therefore, the output of inverter 1 will go high. The high input to inverter 2 ensures that its output remains low. At this time capacitor C charges through R to the output voltage of inverter 1. When the voltage to the input of inverter 1 reaches approximately 1.5 volts, the output of inverter 1 will go low forcing the output of inverter 2 high. The high output from inverter 2 plus the charge on capacitor C ensures a high level to the input of inverter 1 keeping its output low. Capacitor C now begins to discharge through R. As soon as the charge on C becomes low enough, the output of inverter 1 will switch high causing the output of inverter 2 to go low. The cycle will then repeat itself. The period (p) of oscillation of this circuit is approximately 3 RC. As in the other circuits, inverter number 3 is used to isolate the load from the frequency determining components and to ensure a clean square wave output.

$$\boxed{\begin{aligned} p &= 3RC \\ f &= \frac{1}{3\,RC} \end{aligned}}$$

Two-Phase Clocks. The circuits we have discussed generate a single-phase clock. For most MOS integrated circuits, a two-phase clock is required. Bipolar logic circuits and CMOS usually operate from a single-phase clock. There are several different methods of generating two-phase clock signals, but one of the most commonly used methods is shown in Figure 7-67. Two TTL JK flip-flops are connected to form a

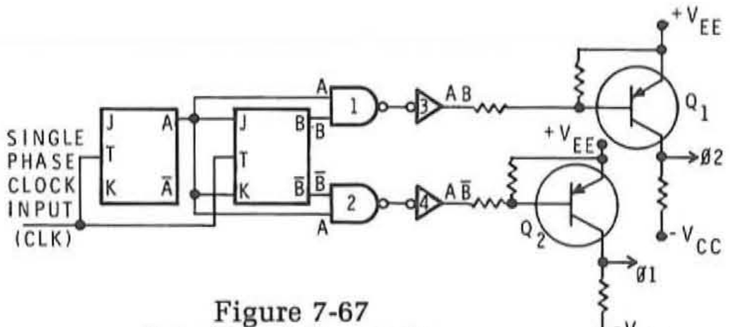


Figure 7-67
Two-phase clock generator.

synchronous 2-bit binary counter. The count sequence is 00, 01, 10, and 11. Gates 1 and 2 are used to detect the AB and $A\overline{B}$ states of the counter. These gates are used to drive transistors Q1 and Q2 which form the interface circuitry for developing the proper logic levels for use with PMOS integrated circuits. The phase 1 (ϕ1) and phase 2 (ϕ2) clock signals switch between $+V_{EE}$ and $-V_{CC}$. These levels are typically +5 and −5 volts. The waveforms for the two-phase clock circuitry are shown in Figure 7-68.

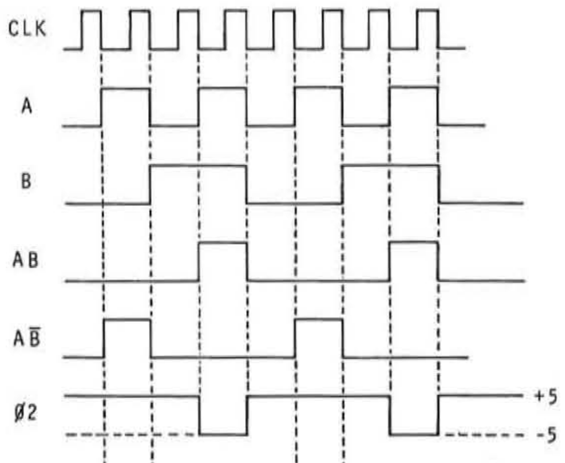


Figure 7-68 Waveforms for two-phase clock generators.

When both flip-flops are set, the output of inverter 3 goes high. Q1 cuts off and the $\phi 2$ output becomes $-V_{CC}$. When the output of inverter 3 goes low, Q1 conducts and the output becomes $+V_{EE}$. The operation of Q2 is similar. When flip-flop A is set and B is reset, the output of inverter 4 goes high. Q2 cuts off and the $\phi 1$ output goes to $-V_{CC}$. When the output of inverter 4 is low, Q2 conducts and the $\phi 1$ output is $+V_{EE}$. The waveforms show the sequence.

One Shot Multivibrators

The one shot or monostable multivibrator is a circuit that generates a rectangular output pulse of a specific time duration each time it receives an input trigger pulse. The output pulse duration is usually adjustable by varying the value of external circuit components. By cascading these circuits, a variety of sequential logic operations can be implemented.

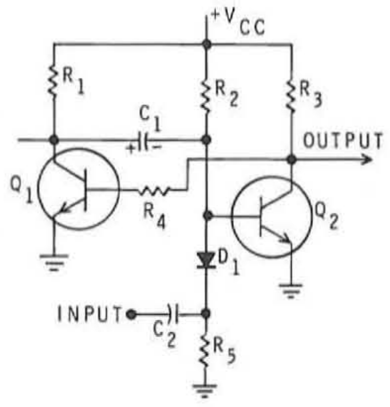


Figure 7-69
One shot multivibrator.

Discrete Component One Shot Circuit. Figure 7-69 shows the schematic diagram of a one shot multivibrator. This circuit has two states: a stable state where Q2 conducts and Q1 is cut off and an unstable state where Q1 conducts and Q2 is cut off. The circuit normally rests in its stable state when it is not being triggered. The unstable state is initiated when the circuit receives an input trigger pulse. The one shot then goes into its unstable state and for a period of time depending upon the values of C1 and R2 it generates an output pulse. The circuit then returns to its stable state.

In the stable state, resistor R2 forward biases the emitter-base junction of Q2. Q2 saturates and its output is near zero volts or ground. Therefore, the voltage applied to R4 is insufficient to cause Q1 to conduct. Q1 therefore, is cut off and its collector is at $+V_{CC}$. Capacitor C1 charges through the emitter-base junction of Q2 and resistor R1 to a voltage approximately equal to supply voltage $+V_{CC}$.

The circuit will remain in this stable state until it receives an input trigger pulse. To trigger the circuit, an input pulse is applied. The network consisting of C2 and R5 differentiates the input pulse. The sharp positive and negative pulses occurring at the leading and trailing edges of the input waveform are then applied to diode D1. D1 permits only the negative going pulse to be coupled to the base of Q2. The negative going pulse reverse biases the emitter-base junction of Q2. Q2 switches off and its output voltage rises to $+V_{CC}$. This causes Q1 to become forward biased. It receives base current through R3 and R4 from $+V_{CC}$. With Q1 saturated its collector is near zero volts. C1 begins to discharge through resistor R2. The negative voltage from this capacitor at the base of Q2 keeps Q2 cut off. As C1 discharges through R2, however, its voltage drop becomes smaller. Soon C1 will be completely discharged and will begin to charge to the opposite polarity. When the voltage across it is high enough, it forward biases Q2. As soon as Q2 switches on, the output pulse is terminated. C1 then recharges through the emitter-base junction of Q2 and R1. Figure 7-70 shows the input and output waveforms generated by the one shot circuit.

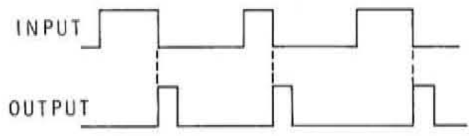


Figure 7-70 Waveforms for the one shot multivibrator.

There are several important facts about the one shot that should be considered in its application. First, the output pulse duration is a function of the values of C1 and R2. The value of R2 is rather critical since it must be low enough in value to ensure the complete saturation of Q2 during normal operation. The value of C1 can be almost any value. The time duration of the output pulse (t) is approximately $t = 0.69R2\ C1$. In most practical monostable multivibrators, the output pulse can be adjusted from nanoseconds to seconds.

The duty cycle is generally limited to a maximum of approximately 90 percent. A duty cycle greater than approximately 90 percent will generally cause the circuit to operate unreliably. The reason for this is that sufficient time must be provided for the circuit to recover between input trigger pulses. This is the time required for capacitor C1 to completely recharge through the emitter-base junction of Q2 and R1 after a pulse has been generated. This finite charge time for C1 can be reduced by making R1 smaller. However, there is a limitation because of practical circuit considerations. As for minimum duty cycle, there is no practical lower limit. Duty cycles of only a few percent can be achieved with such a multivibrator.

DUTY CYCLE

Duty cycle is the ratio of the output pulse duration to the total period of the trigger pulse input expressed as a percentage.

$$\text{Duty cycle} = \frac{t}{p} \times 100 \text{ percent}$$

Here t is the pulse duration and p is the period.

As an example, assume the pulse duration is 5 milliseconds and the input frequency is 50 Hz. See Figure 7-71. The input period is $1/50$ = .02 seconds or 20 milliseconds. The duty cycle then is:

$$\text{Duty cycle} = \frac{5}{20} \times 100 = 25 \text{ percent}$$

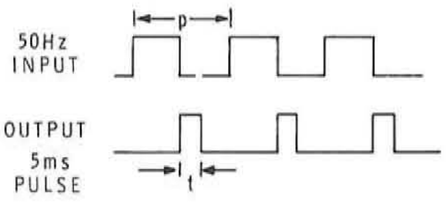


Figure 7-71

Integrated One Shots. Most one shot circuits in use today are in integrated circuit form. Their operation is virtually identical to the discrete component circuit just discussed.

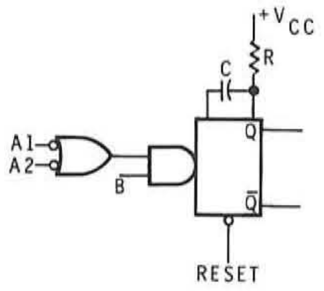


Figure 7-72
Integrated circuit one shot.

Figure 7-72 shows the logic symbol used to represent these one shots. This circuit has three inputs by which the one shot may be triggered. Inputs A1 and A2 can trigger the one shot if the B input is held high. Inputs A1 or A2 will trigger the one shot on the trailing edge. When A1 or A2 switches from high to low, the one shot will generate an output pulse. Complementary output pulses appear at the Q and $\overline{Q}$ outputs. The duration of the output pulse is a function of the external components C and R. The manufacturer provides guidelines for selecting these values and charts for computing the pulse switch for given values of C and R. Generally, the external value of R is limited to approximately 50 k ohms while practically any value of capacitance from 10pf to 100μf can be used. Duty cycles as high as 90 percent are possible.

Input B can also be used to trigger the one shot if inputs A1 and A2 are not used (held low). The one shot will be triggered when input B switches from low to high. In other words, input B triggers the one shot on the leading edge of the input. This input is used primarily for inhibiting or enabling of inputs A1 and A2. Note also that the one shot has a reset input. This is similar to the asynchronous direct clear input of a JK flip-flop. Bringing this input low automatically terminates the output pulse during a timing period. When the one shot is not triggered, the Q output is binary 0 while $\overline{Q}$ is binary 1. When a trigger pulse is received, the one shot goes into its unstable state where Q is binary 1 and $\overline{Q}$ is binary 0. A reset pulse applied during the timing period will cause the normal output to switch to binary 0, immediately terminating the timing sequence. The waveforms of Figure 7-73 illustrate these operations of the IC one shot. Input pulses 1 and 2 trigger the circuit into operation on the trailing edge. The output is a pulse whose duration t is defined by the values of R and C. Note that the timing interval terminates prior to the application of each new input pulse. On the third input pulse, the one shot is triggered but the timing interval is cut short because of the occurrence of a reset pulse.

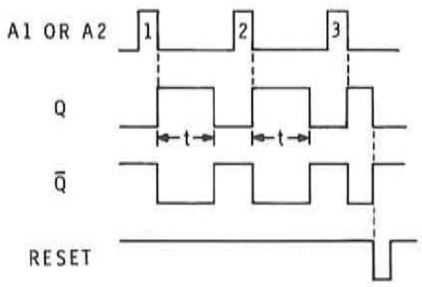


Figure 7-73
Waveforms of an IC one shot.

Another type of IC one shot circuit available to the digital designer is the retriggerable monostable or negative recovery monostable. Most one shots require a finite period of time in order to recover from a trigger pulse. Once a one shot has been triggered and times out, it will take a short period of time for the capacitor to become recharged through the circuitry resistances. It is this recovery time that limits the upper duty cycle limit of most one shots to approximately 90 percent. The retriggerable monostable eliminates this problem. Its recovery time is practically instantaneous thereby making 100 percent duty cycle outputs a possibility. A 100 percent duty cycle represents a constant binary 1 output at Q.

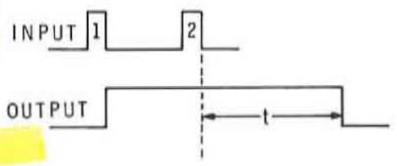


Figure 7-74
Input and output waveforms of a retriggerable monostable.

One of the benefits of the retriggerable monostable is its ability to generate very long duration output pulses. By adjusting the external resistor and capacitor values of the one shot to provide an output pulse duration that is longer than the interval between the input trigger pulses, the retriggerable one shot will remain in the triggered state for a substantial period of time. The waveforms in Figure 7-74 illustrate this effect. Initially, the one shot is in its normal stable state. When the trailing edge of input pulse 1 occurs, the monostable is triggered. However, before it can complete its output pulse whose duration is a function of the external component values, input pulse 2 occurs. When its trailing edge occurs the first timing interval is automatically terminated and a new timing interval initiated. This happens quickly so that the output remains high. Note that another input pulse does not occur after input pulse 2 and therefore the one shot is then allowed to time out and generate its normal output pulse width t.

In addition to generating very long output pulses, the retriggerable one shot can also be used as a missing pulse detector. By making the pulse width of the multivibrator longer than the period of input trigger pulses, the one shot will remain triggered during the sequence of input pulses. If one of the input pulses should disappear or be lost due to a malfunction or noise interference, the one shot will time out. Its output will go low and will therefore indicate the missing pulse.

One Shot Applications

Because of the flexibility of an integrated circuit one shot, many sequential operations can be quickly and easily implemented. The ability to adjust the output pulse width with external components to a desired value plus the retriggerable and reset features makes the one shot a versatile component. As a result, digital designers find many applications for it. Because of the nature of the one shot, these applications involve pulse generation, timing and sequencing. To generate a pulse of specific width, all that the designer needs to do is to add a one shot with the appropriate size external resistor and capacitor.

Figure 7-75 shows how one shots can be used for generating a sequence of timing pulses. Here one shots labeled A, B, and C trigger one another. Assume that the one shots trigger on the trailing edge of the input. The waveforms for this circuit are shown in Figure 7-76. When an input pulse occurs, it triggers one shot A. This one shot generates a pulse width t_1, that is a function of its external component values. At the termination of its output pulse, it triggers one shot B. One shot B generates another output pulse of a specific duration t_2. Upon its termination this pulse triggers one shot C which produces another output pulse, t_3. Such a chain of one shots provides a simple method of sequencing and timing digital operations.

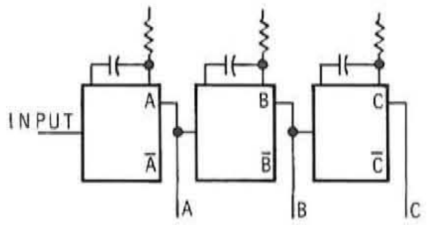


Figure 7-75
One shot pulse sequence generator.

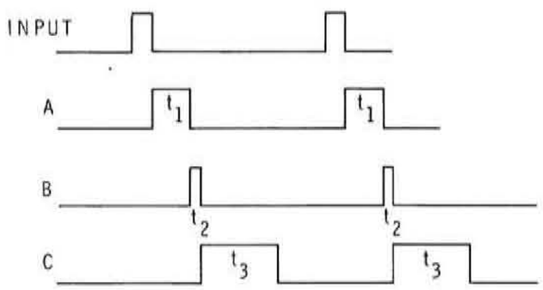


Figure 7-76 Waveforms of the one shot pulse sequencer.

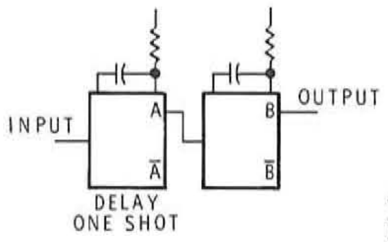


Figure 7-77
Pulse delay using one shots.

Another common application for the one shot is in implementing a delay. In some circuits it is necessary to delay the operation of a particular portion of a circuit. This is essentially a timing operation. A one shot can provide this delay. The input signal to be delayed is applied to the A one shot as shown in Figure 7-77. The A one shot generates the desired delay time. At the end of its delay interval it triggers one shot B which then produces the output pulse that initiates the desired operation. The waveforms in Figure 7-78 illustrate this delay function. The pulse width of one shot B can be adjusted to equal that of the input pulse if desired.

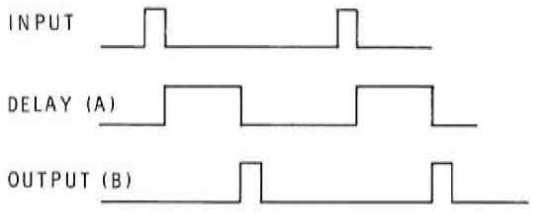


Figure 7-78 Using a one shot to delay the occurrence of a pulse.

While the one shot appears to be a very flexible and versatile component, it has a poor reputation among digital designers. Before the availability of the high quality integrated circuit one shots, the monostable functions were implemented with discrete component circuits like the one discussed earlier. Such circuits were generally unstable and unreliable. In order to provide a very stable fixed output pulse width, high quality timing resistors and capacitors had to be used. In addition, one shots of this type were very susceptible to false triggering by noise on the power supply line, at the normal trigger input or on the circuit ground. Any stray noise or "glitch" can effectively trigger the one shot and cause timing operations to occur at times when it is not wanted. Because of these problems most digital designers attempt to design without one shots. In most cases, timing functions can be implemented with other types of logic circuits such as counters and shift registers combined with logic gates. In synchronous logic circuits under the control of a master timing clock signal, sequencing pulses with the proper time intervals and durations can be readily generated without the use of one shots. Normally this method is preferred.

The modern integrated circuit one shot has overcome most of the problems associated with the early unreliable circuits. However, the timing pulse stability is still largely a function of the quality of the external resistor and capacitor used to set the output pulse duration. The noise problems have essentially been taken care of by providing high threshold noise immunity at the input. By the use of proper grounding and power supply decoupling networks, false triggering can be kept to a minimum. A good general rule of thumb is to design sequential logic circuits using counters, registers, and gates and developing the timing pulses based on synchronous clock signals. However, you will find some applications where one shots are necessary and desirable.

Self Test Review

45. The two basic types of clock oscillators are the __________ __________ and __________ __________.
46. What determines the frequency of oscillation of most clock circuits?
    a. Crystal
    b. Power supply voltage
    c. RC time constant
47. Two phase clocks are used mainly with which type of logic circuits?
    a. CMOS
    b. ECL
    c. TTL
    d. MOS
48. A crystal controlled clock is used when the clock frequency must be __________ and __________.
49. A discrete component one shot has a timing resistor of 33 k and a capacitor of .01 μf. What is the duration of the pulse it generates?
50. The upper duty cycle limit on most one shots is __________ percent.

Answers

45. astable multivibrator, relaxation oscillator
46. c. RC time constant
47. d. MOS
48. accurate, stable
49. $t = 0.69\ (33000)\ (.01 \times 10^{-6})$
    $t = .2277 \times 10^{-3}$
    t = .2277 millisecond or 227.7 microsecond
50. 90 percent

EXPERIMENT 17

CLOCKS AND ONE SHOTS

OBJECTIVES: To demonstrate the operation of several clock oscillator circuits and a retriggerable one shot multivibrator.

Materials Required

Heathkit Digital Design Experimenter ET-3200

2 — MPS A20 transistors (417-801)

1 — 150 ohm 1/2w resistor

2 — 1 K ohm 1/2w resistors

2 — 4.7 K 1/2w resistors

2 — 1000 μf electrolytic capacitors

1 — type 7404 TTL IC (443-18)

1 — type 74123 TTL IC (443-90)

2 — IN4149 diodes (56-56)

1 — 47k 1/2 watt resistor

Procedure

1. Construct the astable multivibrator circuit shown in Figure 7-79. Take your time in wiring the circuit to be sure that you do not make any wiring mistakes. You will observe the operation of this circuit on LED indicators L1 and L2.

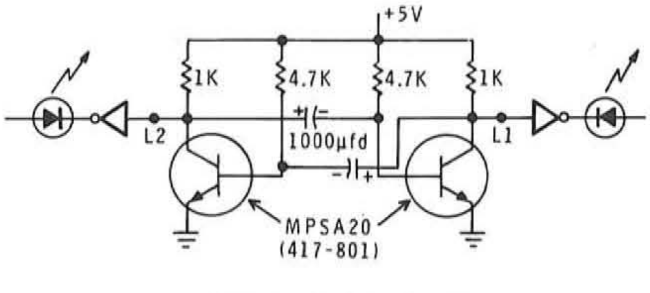


Figure 7-79 Astable multivibrator experimental circuit.

2. Using the component values indicated in Figure 7-79, compute the frequency of oscillation of this astable multivibrator circuit. Record your answer below.

   F = ______________ Hz

   Next, compute the period of oscillation for this circuit.

   Period = ______________ seconds

   The duty cycle of this circuit will be ______________.
3. Apply power to the circuit and observe LED indicators L1 and L2. Use the sweep-second hand on your watch to measure the period of oscillation. Record your measured value below and compare to the computed value you determined earlier.

   Period = ______________ seconds

   Does the actual operation of the circuit correspond to the answers you gave in Step 2 above? Account for any discrepancy that you might observe.
4. Construct the clock oscillator circuit shown in Figure 7-80. You will use a type 7404 TTL hex inverter. Be sure to connect +5 volts and ground to the integrated circuit. You will observe the circuit output on LED indicator L4.
5. Apply power to the circuit. Measure the period of oscillation using the sweep-second hand on your watch and record below.

   Period = ______________ seconds

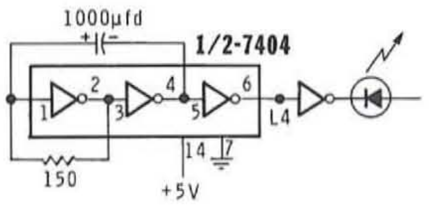


Figure 7-80
IC clock oscillator circuit.

Discussion of Steps 1 through 5

In these steps you demonstrated two types of clock oscillators. The discrete component clock in Figure 7-79 is a standard astable multivibrator. The circuit should switch repeatedly between its two states as indicated by LED indicators L1 and L2. When L1 is on L2 will be off and vice-versa.

Using the formula given earlier in the unit, the frequency of oscillation should be about .154 Hz. This means that the circuit should have a period (time for one cycle) of approximately 1 ÷ .154 or about 6.5 seconds. This means that the circuit should change state every 6.5 seconds. Each LED indicator will remain on for approximately 3.25 seconds. This very low frequency of oscillation is caused primarily by the very high value of capacitance used in the circuit. Higher frequencies can be obtained by using smaller capacitor values.

Your measured period may be somewhat different from your calculated value. The formula given for computing the frequency is an approximation to begin with, but the most likely cause for the difference between your computed and measured values is the tolerances of the timing resistors and capacitors. Since both capacitors are the same and the base resistors are equal, the duty cycle of the circuit should be 50 percent.

In Steps 4 and 5 you demonstrated a clock oscillator circuit made from TTL inverters. As indicated earlier in the unit, the period of oscillation of the circuit is approximately 3 RC. Using the component values shown in Figure 7-80, the period of oscillation should be approximately .45 second. This represents a frequency of 2.22 Hz. In other words, the LED indicator L4 should flash on and off once every .45 seconds. Because of the asymmetrical nature of this circuit, the duty cycle will be less than 50 percent.

Procedure (Continued)

6. Disassemble the clock oscillator circuits you constructed in the previous steps. On the breadboarding socket of your Experimenter, construct the circuit shown in Figure 7-81. This circuit uses the 74123 dual retriggerable one shot. The circuit is wired so that the first one shot will be triggered from the A logic switch. This one shot will in turn trigger the second one shot in the IC. External resistors and capacitors are used to set the duration of the pulses produced by each one shot. You will monitor the one shot outputs on LED indicators L1 and L4. Note that the B logic switch is connected to the reset (C) input of the first one shot. The diodes associated with the RC timing components are used to prevent a reversed voltage from being applied to the 1000 μfd electrolytic capacitors. Don't forget to connect +5 volts to pin 16 and ground to pin 8 of the IC.

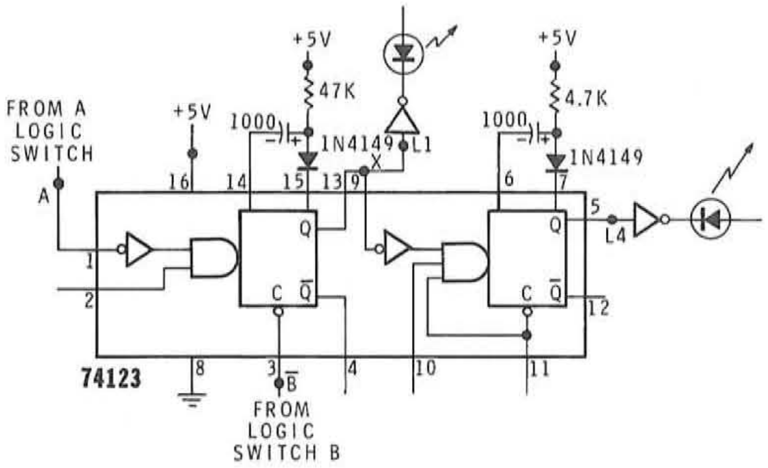


Figure 7-81
One shot experimental circuit.

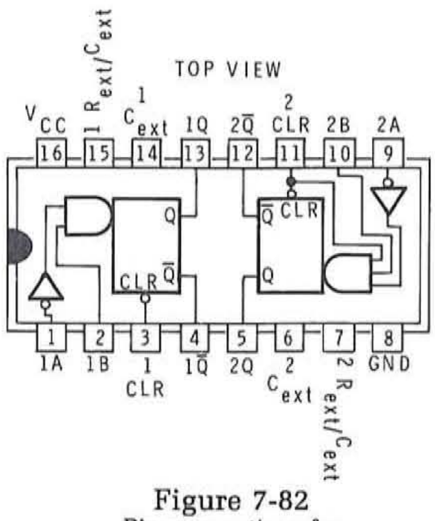


Figure 7-82
Pin connections for
74123 dual retriggerable one shot.

The pin connections for the 74123 IC are shown in Figure 7-82. The pulse duration produced by this one shot is a function of the external component values R and C and can be computed with the formula below.

$$t = .25\ RC\ (1 + \frac{0.7}{R})$$

In this formula the resistance value R is in k ohms and the capacitance value C is in μf. The output pulse duration t will be in milliseconds.

7. Using the formula given above, compute the pulse duration of each one shot in Figure 7-81. Record your pulse durations below. The output pulse width of the first one shot is t_1 and the output of the second one shot is t_2.

   t_1 = ______________ ms
   t_2 = ______________ ms

8. Study the circuit shown in Figure 7-81. Determine the operation of the circuit. Assume that the circuit operation is initiated by actuating the A logic switch. Sketch the input and output waveforms for the circuit.

   Will the circuit be triggered when the A logic switch is depressed or released? ______________.

9. Depress the A logic switch and release it. Note the LED indicators to see what operation occurs. Use the sweep-second hand of your watch to time the one shot output durations by observing L1 and L4. Repeat the sequence as often as necessary to verify the operation of the circuit.

   Does the actual operation of the circuit correspond to your result in Step 8? ______________.

10. Momentarily depress and release the A logic switch. Note LED indicator L1. After a second or two, depress the B logic switch while noting L1 and L4. What happens? ______________.

11. Remove the input from pin 1 of the IC to the A logic switch and connect pin 1 to the clock output CLK. Set the clock frequency to 1 Hz and observe the L1 indicator.

    What is the state of L1? ______________.
    What does this state indicate? ______________.

Discussion of Steps 6 through 12

In these steps you demonstrated the operation of a retriggerable one shot. The retriggerable function is not always used, and when it isn't, this circuit performs like any other monostable multivibrator. The 74123 is triggered into operation on the trailing edge of the input pulse. The circuit you constructed receives a trigger pulse from the A logic switch. The A output normally rests in the low position. When the switch is depressed, A goes high and when it is released goes low again. It is on the high to low transition that the input one shot is triggered. When it is triggered, LED indicator L1 will turn on. This output will remain on until the pulse duration specified by the external resistor and capacitor is completed. According to the calculations using the formula given earlier, the pulse output duration for this one shot (t_1) should be 11.924 or 12 seconds.

When the input one shot times out, its output will switch from high to low. This will trigger the second one shot in the circuit. Its time constant is set to produce an output pulse (t_2) of 1.35 seconds. Therefore, as soon as L1 turns off, L4 should turn on for approximately 1.35 seconds and then go out. This sequence can be repeated by depressing the A logic switch again.

The B logic switch is wired to the clear input of the first one shot. If you trigger the circuit into operation with the A logic switch, the first one shot will remain on for over 12 seconds. However, this timing interval can be terminated or cut short by applying a reset pulse with logic switch B. The moment the B logic switch is depressed, the one shot output will switch off. LED L1 will go out. This will immediately trigger the second one shot and cause L4 to light for approximately 1.35 seconds. Both operations could be terminated by connecting the $\overline{B}$ logic switch output to the clear input of the second one shot as well.

The one shot circuit that you demonstrated here shows how a delay function is implemented. The first one shot produces a delay of over 12 seconds and the second one shot generates a single output pulse 1.2 seconds long. The A logic switch initiates the circuit operation, but it is the output of the second one shot that is generally used to actuate an external circuit. See Figure 7-83.

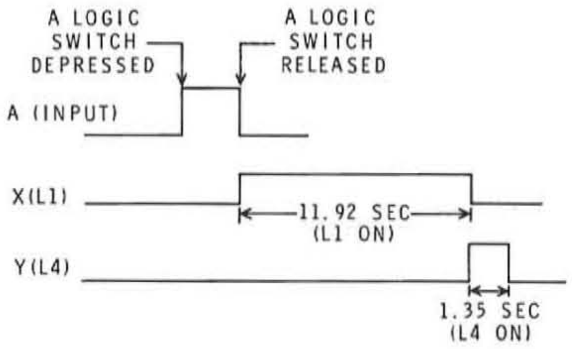


Figure 7-83 Waveforms illustrating the operation of the experimental one shot circuit.

Finally, you connected the input to the circuit to the 1 Hz output of your clock. Since the clock interval is approximately one second, the input one shot will be repeatedly triggered. This will cause the output of the one shot to turn on. Before the circuit can time out, the input will be triggered again. Therefore, the output of the first one shot will remain on as long as the clock signal is applied. When the pulse duration of the one shot is greater than the period of the input triggering signal, the retriggerable feature comes into operation and will keep LED indicator L1 turned on. The L4 indicator will remain off during this time since the second one shot will not be triggered.

EXAMINATION

UNIT 7

SEQUENTIAL LOGIC CIRCUITS

The purpose of this exam is to help you review the key facts in this unit. The problems are designed to test your retention and understanding by making you apply what you have learned. This exam is not so much a test as it is another learning method. Be fair to yourself and work every problem first before checking the answers.

1. Which of the following is **not** a basic function of sequential logic circuits?
   A. Make decisions
   B. Generate timing pulses
   C. Count
   D. Produce automatic sequencing
2. The type of sequential logic circuit that operates from a clock is called a
   A. counter
   B. one shot
   C. synchronous circuit
   D. combinational circuit
   E. asynchronous circuit
3. A binary up counter with flip-flops E D C B A (A = LSB) contains the number 00110. How many input pulses must be applied to obtain the contents 11000?
   A. 7
   B. 16
   C. 18
   D. 24
4. Another name for a decade counter is
   A. binary counter
   B. BCD counter
   C. frequency divider
   D. shift register

5. What is the maximum number that can be counted by a binary counter with 12 flip-flops?
   A. 12
   B. 24
   C. 2047
   D. 4095
6. How many BCD counters does it take to count to the number 816?
   A. 3
   B. 4
   C. 10
   D. 12
7. What is the output frequency of the circuit in Figure 7-84?
   A. 3.9 kHz
   B. 6.25 kHz
   C. 20 kHz
   D. 50 kHz

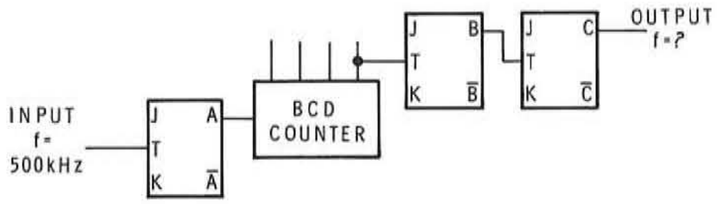


Figure 7-84
Circuit for Exam question 7.

8. What circuit could be used to add and subtract input pulses?
   A. Shift registers
   B. One shot
   C. Frequency divider
   D. Up/down counter

9. Sketch the A and B output waveforms of the special counter circuit in Figure 7-85. Analyze the operation of the circuit by assuming that both flip-flops are initially reset. Then four clock pulses are applied. From the waveforms, determine the count sequence. Select the proper code sequence below.

A.

B	A
0	0
1	0
1	1
0	1

B.

B	A
0	0
0	1
1	0
1	1

C.

B	A
0	0
0	1
1	1
1	0

D.

B	A
0	0
1	1
0	1
1	0

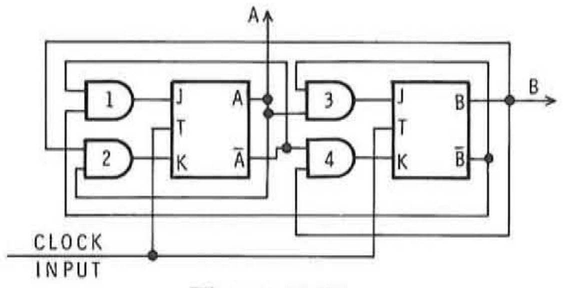


Figure 7-85
Circuit for Exam question 9.

10. The circuit in Figure 7-85 is asynchronous.
    - A. True
    - B. False

11. Which of the following does *not* affect the upper count frequency of a counter?
    - A. Rise and fall times of the input pulses.
    - B. Number of flip-flops in the counter.
    - C. Type of logic circuit flip-flops.
    - D. Propagation delay of flip-flop.

12. A synchronous counter
    - A. is slower than a ripple counter.
    - B. cannot count non-periodic inputs.
    - C. changes state in a time equal to the propagation delay of one flip-flop.
    - D. is one in which one flip-flop toggles the next in sequence.

13. A binary up/down counter with flip-flop EDCBA (A = LSB) contains the number 10001. Five pulses are applied to the up count input. Twenty-four pulses are applied to the down count input. What is the new counter contents?
    A. 01110
    B. 11000
    C. 11101
    D. 11110
14. How many clock pulses are required to load a 16 bit word into a shift register?
    A. 4
    B. 8
    C. 16
    D. 32
15. List four applications of shift registers.
    A. ______________
    B. ______________
    C. ______________
    D. ______________
16. Shift registers can be constructed with D type flip-flops.
    A. True
    B. False
17. Refer to Figure 7-86. How many 8 bit words can be stored serially in the shift register?
    A. 8
    B. 16
    C. 64
    D. 512

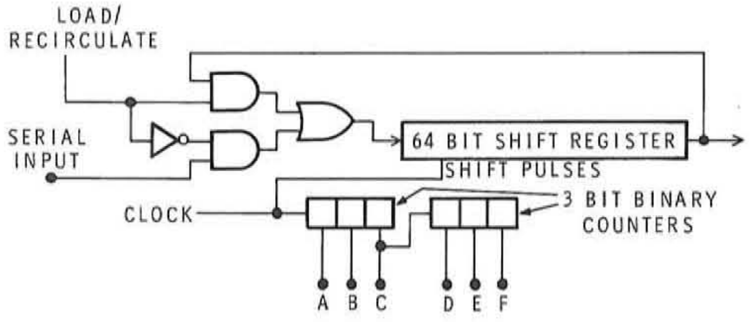


Figure 7-86
Circuit for Exam questions 17 and 18.

18. Refer to Figure 7-86. What is the purpose of the second (DEF) three bit binary counter?
    A. Causes the shift register to be shifted every 8 clock pulses.
    B. Gives a binary output code that designates the location of one of the words in the shift register.
    C. Counts the number of clock pulses required to recirculate the register contents.
    D. Controls the write/recirculate circuitry.
19. Which of the following are **not** features of an MOS shift register?
    A. low cost
    B. parallel outputs
    C. small size, high density
    D. low power consumption
    E. high speed
    F. Two phase clock
20. Which two circuits below can be used to generate a sequence of three timing pulses?
    A. Shift register
    B. BCD counter
    C. One shots
    D. Frequency divider
21. To increase the frequency of an astable multivibrator or a relaxation oscillator the value of the circuit capacitance must be
    A. increased.
    B. decreased.

ANSWERS

UNIT 7

SEQUENTIAL LOGIC CIRCUITS

1. A — Make decisions. Sequential circuits generate timing pulses, count and provide for automatic sequencing of logic operations. Combinational logic circuits make decisions.
2. C — Synchronous circuits operate from a clock signal.
3. C — 18. The binary contents is initially 00110 or 6. To achieve a contents of 11000 or 24, 24 − 6 = 18 input pulses must be applied.
4. B — BCD counter. A BCD counter is sometimes called a decade counter.
5. D — 4095. $2^{12} - 1 = 4096 - 1 = 4095$
6. A — 3. Three BCD counters are required to count to 816, one counter for each decimal digit.
7. B — 6.25 Hz. The overall circuit frequency division is 2 × 10 × 2 × 2 = 80. 500 KHz ÷ 80 = 6.25 Hz.
8. D — Up/down counter. An up/down counter can be used to add and subtract. To add 7 and 9 you first clear the counter and increment it 7 times. You then increment it 9 times. The binary content will be 10000 or the sum 16. You decrement to subtract. To subtract 5 from 16, you decrement 5 times. The content will be 01011 or 11.

9. C — The proper BA count sequence is 00, 01, 11, 10. Refer to Figure 7-85. With both flip-flops reset, $\overline{A}$ and $\overline{B}$ will be high. The output of gate 1 will be high, enabling the J input to the A flip-flop. When the first clock pulse occurs, A sets. A and $\overline{B}$ will then be high, enabling gate 3. With the J input to the B flip-flop high, B will set on the second clock pulse. A will remain set. With A and B set, gate 2 will be enabled. This makes the K input to flip-flop A high. As a result, A resets on the third clock pulse. $\overline{A}$ and B will then be high, enabling gate 4. This makes K input to the B flip-flop high. On the fourth clock pulse the B flip-flop resets. The cycle then repeats. This special counter has four states, so it produces frequency division by four.
10. B — False. The circuit in Figure 7-85 is synchronous since both flip-flops are toggled simultaneously.
11. A — Rise and fall times of the input pulse. This does not affect the upper count frequency.
12. C — Changes state in a time equal to the propagation delay of a single flip-flop. This is the maximum delay of a synchronous counter and this factor sets the upper count frequency.
13. D — 11110. The counter content is 30. The original content is 10001 or 17. Five pulses applied to the up-count input increment the contents to 22 or 100110. Twenty-four pulses are applied to the down-count input. The first 22 of these decrement the counter to 00000. The twenty-third pulse re-cycles the counter to 11111 or 31. The twenty-fourth pulse decrements the counter to 11110 or 30.
14. C — 16. One clock pulse is required to shift each bit one position.

15. A — Multiply and divide by powers of 2 by shifting left or right.
    B — Counting.
    C — Memory, binary word storage.
    D — Serial to parallel and parallel to serial data conversion.
    Also:
    E — Pulse sequence generation.
    F — Frequency division.
16. A — True. A shift register can be made from D type flip-flops as well as JK flip-flops.
17. A — 8. The 64 bit shift register can hold 64 ÷ 8 = 8 eight bit words.
18. B — The second three bit counter (DEF) gives a binary output code that designates the location of one of the 8 bit words in the shift register. It gives the address of the binary word that is about to be shifted out of the shift register. The first 3 bit binary counter (ABC) counts the shift pulses. It recycles every 8 input pulses and increments the second counter. Assume both counters are reset. Eight shift pulses causes one 8 bit word to be shifted out and recirculated. This word has the address or location 000. After the 8 shift pulses, the second counter is incremented to 001. This is the address of the next memory word in sequence. When the next 8 shift pulses are applied, this next word is shifted out. The second counter is incremented to 010, the location code or address for the third word in memory. The counters are a method of keeping track of the location of the eight 8 bit words stored in the shift register.
19. B — Parallel outputs E — High speed. Most MOS shift registers do not have parallel outputs or provide high speed (over 10 MHz) operation.
20. A — Shift register C — One shots. Either of these circuits can generate a sequence of timing pulses.
21. B — Decrease. Decreasing the circuit capacitance decreases the charge (and discharge) times thereby allowing the circuit to switch states at a faster rate.

HEATHKIT CONTINUING EDUCATION

Individual Learning Program In DIGITAL TECHNIQUES

8 COMBINATIONAL LOGIC CIRCUITS

703-8

Heathkit
Zenith Educational Systems

UNIT 8

COMBINATIONAL LOGIC CIRCUITS

CONTENTS

UNIT 8

COMBINATIONAL LOGIC CIRCUITS

INTRODUCTION

Combinational logic circuits are digital circuits that are made up of gates and inverters. The output of a combinational logic circuit is a function of the states of its inputs, the types of gates used, and how they are interconnected. As you saw in a previous unit on Boolean algebra, there are many different ways to interconnect logic gates to form combinational circuits. Any unique binary function can be implemented.

An analysis of a wide variety of different types of digital equipment reveals that there are certain combinational logic circuits that regularly reoccur. Despite the large possible number of combinational circuits, most digital equipment can be implemented with just a few basic types. These circuits are called functional logic circuits. The most common functional logic circuits are decoders, encoders, multiplexers, comparators, and code converters.

In this unit, you are going to study the most common types of functional combinational logic circuits. You will learn how they operate and how they are used. You will see that even though you can construct these common functional circuits from gates and inverters, in most cases these functional logic circuits are already available as a completely wired and ready to use MSI integrated circuit. The availability of these functional circuits in MSI form, usually eliminates the need to design them. In designing digital equipment, you will find that the job is largely one of identifying the functional circuits, selecting appropriate MSI devices and interconnecting them properly.

The Unit Objectives outline specifically what you will learn in this unit. Review these now, then go on to the Unit Activity Guide for your specific instructions in completing this unit.

UNIT OBJECTIVES

When you complete this unit you will be able to:

1. Name at least seven different types of standard combinational or functional logic circuits.
2. Write the output states of a decoder, encoder, multiplexer, demultiplexer, given the input states.
3. Implement a decoder for any states with NAND or NOR gates.
4. Name three applications for a multiplexer circuit.
5. Write the output states of an exclusive OR and an exclusive NOR circuit given the input states.
6. List three applications for the exclusive OR gate.
7. List four commonly used code conversions.
8. Explain the operation of a read only memory.
9. Give three applications for a ROM.
10. Define a programmable logic array.

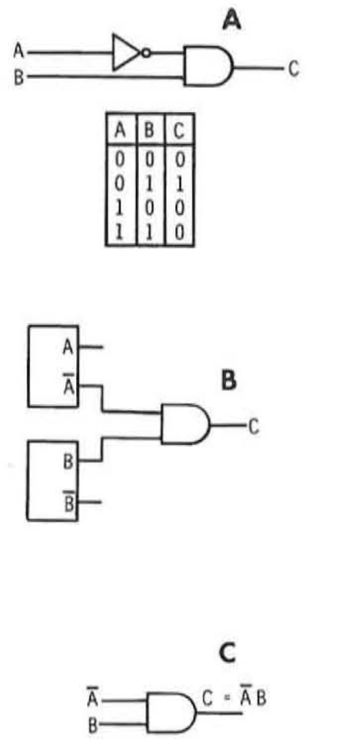
UNIT ACTIVITY GUIDE

	Completion Time
☐ Play audio record 6, side 2 Combinational Logic Circuits	________
☐ Read section Decoders	________
☐ Answer Self Test Review questions 1-6	________
☐ Perform Experiment 18	________
☐ Read section Encoders	________
☐ Answer Self Test Review questions 7-10	________
☐ Perform Experiment 19	________
☐ Read section Multiplexers	________
☐ Answer Self Test Review questions 11-15	________
☐ Perform Experiment 20	________
☐ Read section Demultiplexers	________
☐ Answer Self Test Review questions 16-19	________
☐ Read section Exclusive OR	________
☐ Answer Self Test Review questions 20-27	________
☐ Perform Experiment 21	________
☐ Read section Code Converters	________
☐ Answer Self Test Review questions 28-31	________
☐ Perform Experiment 22	________
☐ Read section Read Only Memories	________
☐ Answer Self Test Review questions 32-45	________
☐ Read section Programmable Logic Arrays	________
☐ Answer Self Test Review questions 46-50	________
☐ Complete Unit Examination	________

DECODERS

One of the most frequently used combinational logic circuits is the decoder. A decoder is a logic circuit that will detect the presence of a specific binary number or word. The input to the decoder is a parallel binary number and the output is a binary signal that indicates the presence or absence of that specific number.

The basic decoding circuit is an AND gate. The output of an AND gate is a binary 1 only if all inputs are a binary 1. By properly connecting the inputs on an AND gate to the source of the data, the presence of any binary number will be detected when the output is binary 1.



A	B	C
0	0	0
0	1	1
1	0	0
1	1	0

Figure 8-1. Two input AND gate decoders used for detecting the number 01.

Figure 8-1A shows a two input AND gate used to detect the presence of the two bit binary number 01. The number to be detected consists of two bits, A and B, with B the least significant bit (LSB). When A is 0 and B is 1, both inputs to the AND gate will be high and the output C will be a binary 1 indicating the presence of the desired number. The inverter on the A input causes the upper input to the AND gate to be binary 1 when the A input is binary 0. For any other combination of input bits the decoder output will be binary 0.

The truth table accompanying the circuit in Figure 8-1A illustrates the performance of the circuit. Note that when the input number is 01 the output C is binary 1. For all other two input combinations the output is binary 0. The circuit does indeed detect the presence of the number 01.

Figure 8-1B shows the AND gate decoder for detecting the number 01 where the binary number input source is a flip-flop register. Since the complement outputs of the flip-flops are available, the inverter is not needed. When the A flip-flop is reset and the B flip-flop is set, the number stored in the register is 01. At this time the $\overline{A}$ and B outputs are high. The decode gate output will be high at this time.

To simplify the drawing of a decoder circuit, the AND gate input source is often omitted. See Figure 8-1C. Only the input states are shown at the gate inputs. Note the output equation which can be written from the circuit or the truth table.

An AND gate can be used to detect the presence of any binary number regardless of size. The number of inputs to the gate will be equal to the number of bits in the binary word. Figure 8-2 shows how a four input gate can be used to detect the binary number ABCD = 0101. Note that the decode gate receives its inputs from a 4 bit register. When the number 0101 is present in the register the output of the decode gate will be a binary 1. For any other 4 bit number in the register, the decoder output will be a binary 0.

While there are some situations where the presence of a single binary word must be recognized, most applications require the detection of all possible states that can be represented by the input word. For example, with a two bit input word there are a total of $2^2 = 4$ different input combinations that exist. A practical decoder will recognize the existence of each of these states.

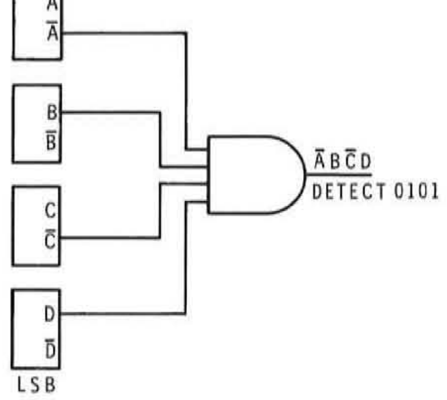
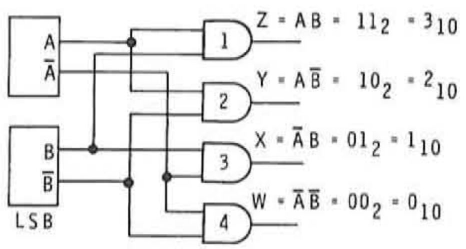


Figure 8-2. Four input AND gates used to decode 0101.


A
$\overline{A}$
B
$\overline{B}$
LSB
1
2
3
4
$Z = AB = 11_2 = 3_{10}$
$Y = A\overline{B} = 10_2 = 2_{10}$
$X = \overline{A}B = 01_2 = 1_{10}$
$W = \overline{A}\overline{B} = 00_2 = 0_{10}$

INPUTS		OUTPUTS			
A	B	W	X	Y	Z
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Figure 8-3. 1 of 4 decoder.

Figure 8-3 shows such a decoder. A two bit binary word with bits A and B (B is the LSB) is stored in a flip-flop register. Four AND gates are used to decode the four possible combinations. For example, gate 4 detects the 00 input state. If the binary number stored in the flip-flops is 00, the $\overline{A}$ and $\overline{B}$ outputs will be high. Gate 4 will produce a binary 1 output. Gates 1, 2, and 3 will have a binary 0 on at least one of their inputs thereby keeping their outputs low. The truth table in Figure 8-3 shows the four possible input states and the outputs of each of the decoder gates. Such a decoder circuit is called a one of four decoder since only one of the four possible outputs will be a binary 1 at any given time.

Another way of looking at the decoder circuit in Figure 8-3 is as a binary to decimal converter. It converts a binary number into an output signal representing one of the four decimal numbers 0, 1, 2, or 3. If flip-flops A and B are both set, the register is storing the binary number 11_2. Gate 1 will be enabled at this time and its output will indicate the presence of that particular number in the register. The output of this gate could then be used to turn on an indicator light marked with the decimal number 3.

BCD to Decimal Decoder

One of the most common applications for decoder circuits is binary to decimal conversion. A widely used type of decoder is the BCD to decimal decoder. The input to the decoder is a parallel four bit number representing the BCD digits 0000 through 1001. Ten AND gates are used to look at or observe the inputs and decode the ten possible output states 0 through 9. When a BCD number is applied to this decoder, one of the ten output lines will go high indicating the presence of that particular BCD number. The output of such a decoder is generally used to operate a lighted decimal number read-out or display.

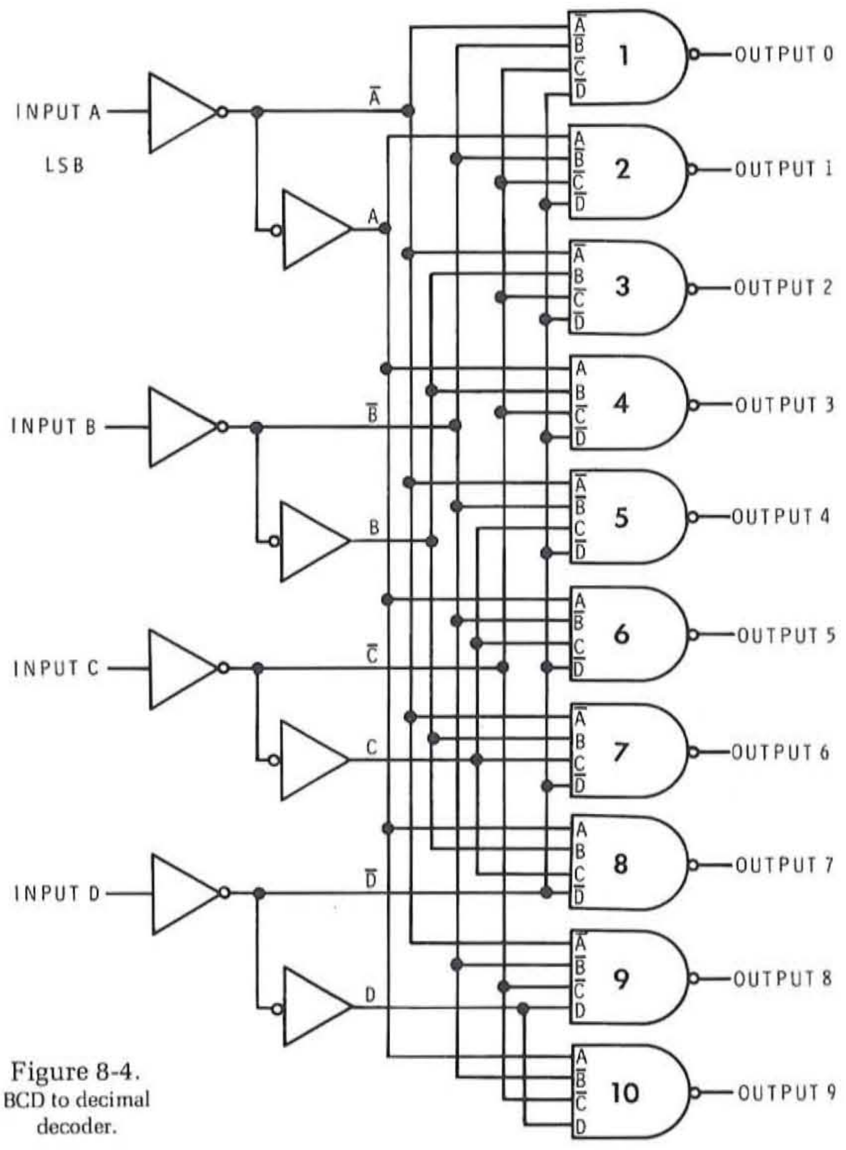


Figure 8-4.
BCD to decimal decoder.

A typical BCD to decimal decoder is shown in Figure 8-4. The four bit BCD number with bits designated A, B, C, and D are applied to the inverters which generate the normal and complement versions of the inputs to be applied to the decode gates. Bit A is the LSB. Note also that NAND gates are used instead of AND gates for the decoding process. When all of the inputs to a NAND gate are binary 1, its output will be binary 0. For all other input combinations the output will be binary 1. For that reason, all of the outputs from the gates in this decoder are high except the one decoding a specific input state.

Figure 8-5 shows the truth table for the BCD to decimal decoder. When one of the ten 8421 BCD codes is applied to the input, the appropriate output will go low. For example, when the input 0110 is applied to the decoder, all inputs to gate 7 will be binary 1. The output of gate 7 will go low indicating the presence of the four bit BCD number representing a decimal 6 is at the input. All other gate outputs will be high at this time. Notice in the truth table that if any one of the six invalid four bit BCD code numbers is applied to the input of the decoder, all outputs will remain high. This BCD decoder simply does not recognize the four bit words that are not included in the standard 8421 BCD code.

NO.	BCD INPUT				DECIMAL OUTPUT									
	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	H	H	H	H	H	H	H	L
INVALID	H	L	H	L	H	H	H	H	H	H	H	H	H	H
	H	L	H	H	H	H	H	H	H	H	H	H	H	H
	H	H	L	L	H	H	H	H	H	H	H	H	H	H
	H	H	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	H	L	H	H	H	H	H	H	H	H	H	H
	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = BINARY 1
L = BINARY 0

Figure 8-5. Truth table for BCD to decimal decoder.

The decoder circuit in Figure 8-4 can be readily constructed with individual logic gates. A typical SSI logic gate provides two four input NAND gates in a single dual in line package. To decode the 10 output states, five of these integrated circuits would be required. The inverters could be implemented with a hex inverter IC. A typical unit contains six inverter circuits in a single DIP. Since 8 inverters are required, two of these hex inverter ICs would be required. This makes a total of seven integrated circuits required to implement the BCD to decimal decoder. Some form of printed circuit board or other interconnecting medium would be required to wire the circuit as indicated.

Fortunately, modern integrated circuit technology has eliminated the necessity for constructing such a circuit with SSI logic circuits. The entire BCD to decimal decoder circuit shown in Figure 8-4 is available in a single 16 pin dual in line package. Because of the complexity of this circuit it is considered to be a medium scale integrated circuit. This is a classical example of a functional MSI logic circuit.

Octal and Hex Decoders

Two other widely used decoder circuits are the one of eight (octal) decoder and the one of sixteen (hexadecimal) decoder. The one of eight decoder accepts a parallel three bit input word and decodes all eight output states representing the numbers 0 through 7. The BCD to decimal decoder circuit in Figure 8-4 can be used as a one of eight or octal decoder by simply using the A, B and C inputs only. The D input is simply wired to a binary 0 condition and the 8 and 9 outputs from gates 9 and 10 are ignored. A hex decoder is a one of sixteen decoder. All 16 states represented by four input bits are recognized by this circuit.

Instead of drawing the complex logic circuitry for a decoder, a simplified block diagram like the one shown in Figure 8-6 is often used. The decimal weights of the inputs and the decimal equivalent of the decoded outputs are indicated within the block to identify its function.

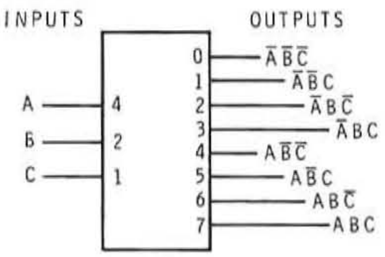


Figure 8-6.
1 of 8 or octal decoder.

BCD to 7 Segment Decoder

A special form of decoder circuit is the popular BCD to 7 segment decoder-driver. This is a combinational logic circuit that accepts the standard 8421 BCD input code and generates a special 7 bit output code that is used to operate the widely used 7 segment decimal readout display. This functional circuit is available as a single package MSI device.

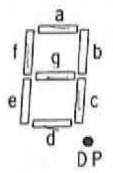


Figure 8-7.
7 segment display format.

A 7 segment readout is an electronic component used to display the decimal numbers 0 through 9 and occasionally special letter combinations by illuminating two or more segments in a specially arranged 7 segment pattern. The standard 7 segment display configuration is shown in Figure 8-7. The segments themselves can be constructed with a variety of light emitting elements such as an incandescent filament, a light emitting diode, a gas discharge glow diode or a liquid crystal segment. By illuminating the appropriate segments, the numbers 0 through 9 and many letters can be displayed. Figure 8-8 shows the typical 7 segment presentation of these numbers.

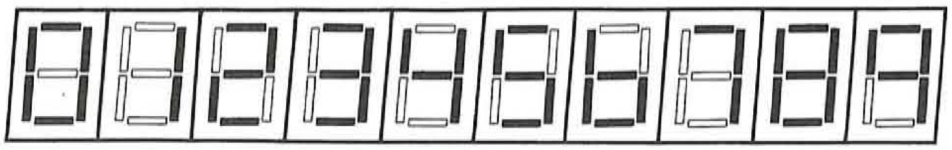
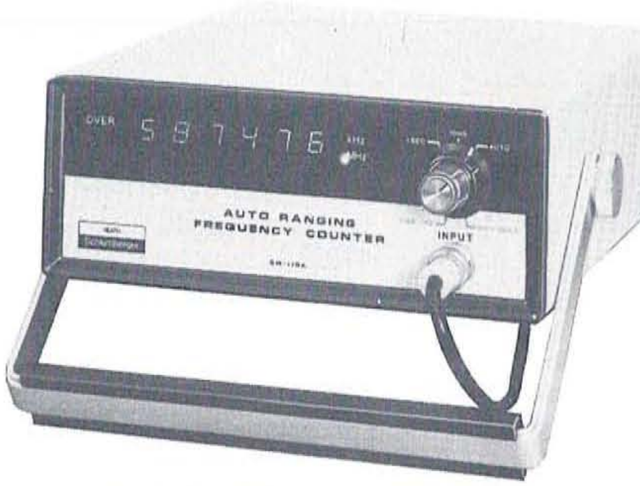
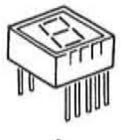


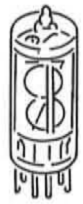
Figure 8-8. 7 segment format for numbers 0 through 9.

Seven segment displays are widely used in electronic equipment such as test instruments, electronic calculators and digital clocks. Several examples are shown in Figure 8-9. The digital counter shown in Figure 8-9A uses light emitting diode displays. A light emitting diode is a special semiconductor junction diode that emits light when it is forward biased. Most LED displays emit a brilliant red light. Yellow and green LED displays are also available.

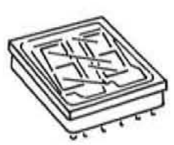




A



B



C

Figure 8-9. Types of seven segment displays used in digital equipment (A) LED, (B) incandescent, (C) gas discharge.

The digital depth sounder in Figure 8-9B uses an incandescent-filament, 7-segment display tube. Each segment is a thin tungsten wire that emits a brilliant white light when current is passed through it. The seven segments are mounted in a single plane within a glass tube. A filter or window in front of the display can make the light any color desired.

The digital clock in Figure 8-9C uses a 7-segment, gas-glow discharge display. Each element of the 7-segment readout is a cathode of a special gas-discharge diode. When a high voltage is applied to the element, the gas surrounding the element is ionized and emits a red-orange glow.

These are only a few of the many different types of 7-segment display devices available. A BCD to 7-segment decoder-driver circuit is used to operate these display devices. This is an MSI logic circuit that decodes the decimal states 0 through 9 and develops the seven output signals that operate the display device.

Figure 8-10 shows the truth table for this decoder circuit. The BCD inputs (ABCD) are in the standard 8421 code form. The outputs are designated a, b, c, d, e, f, g, and correspond to the elements shown in Figure 8-7. A binary 0 in the segment output columns indicates that the corresponding segment is illuminated. You can check this code against the segment letters illustrated in Figure 8-7.

INPUTS					SEGMENT OUTPUTS						
DECIMAL	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	1	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	1	1	0	0

Figure 8-10. Truth table for BCD to seven segment decoder driver.

A logic diagram for one particular type of BCD to 7 segment decoder-driver is shown in Figure 8-11. In addition to the four BCD inputs, this circuit also has a blanking input, a ripple blanking input and a lamp test input. When the lamp test input is binary 0 all seven segments of the display are turned on in order to see that none have failed. When the blanking input is low, all segments are turned off. This feature is used where a number of displays are grouped to readout a multi-digit number. This feature blanks or suppresses all leading zeros automatically. For example, in an eight digit display without leading zero suppression, the number 1259 would be displayed as 00001259. With leading zero suppression, only the desired digits 1259 will show. The other four displays will be automatically blanked. By applying a variable duty cycle pulse signal to the ripple blanking input, the intensity of the display can be varied without resorting to supply voltage or current controls.

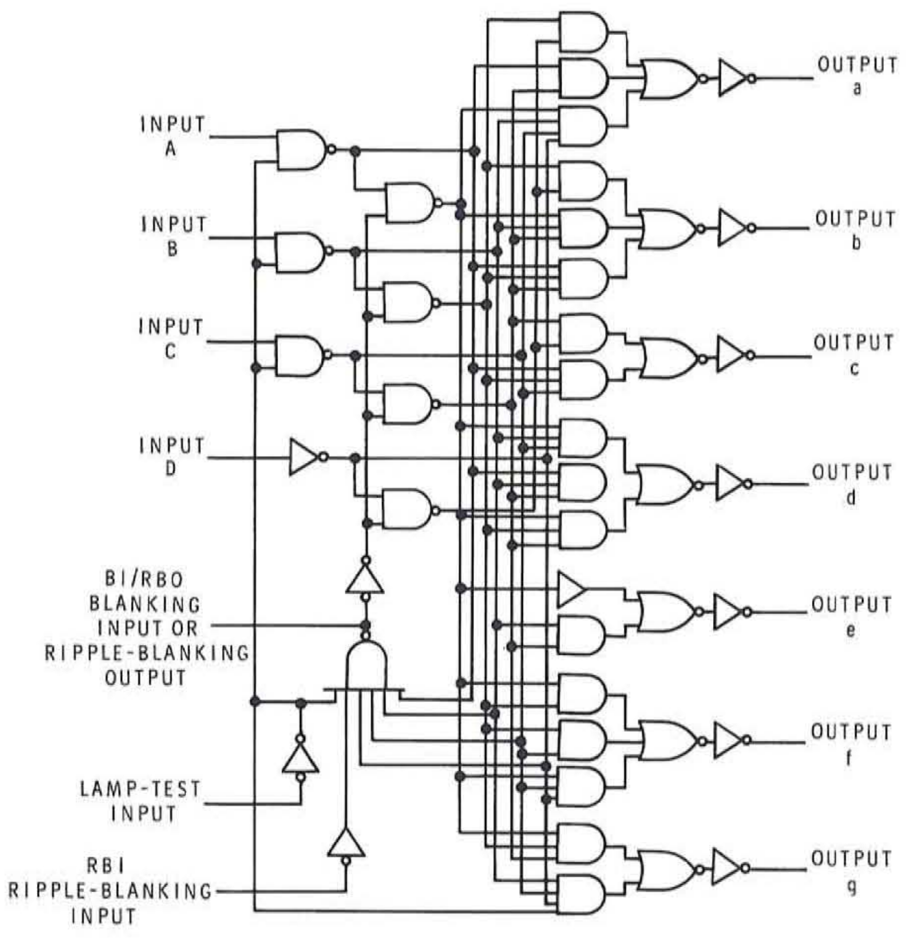


Figure 8-11. BCD-to-seven segment decoder-driver IC.

Figure 8-12 shows the typical output circuit for one segment of the decoder-driver. Besides decoding the BCD input states this circuit also drives or operates the light producing segments. The output is usually a saturated transistor switch with an open collector. Figure 8-12 shows one LED segment connected to the output. When the transistor conducts, the collector output goes low and current flows through the LED turning it on. A series dropping resistor sets the LED current and hence the intensity of the light it produces.

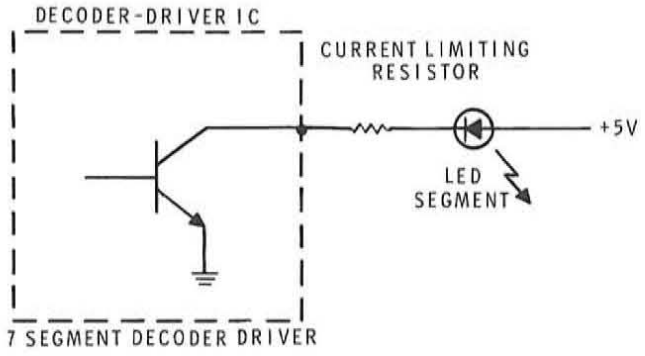


Figure 8-12. Typical 7 segment decoder-driver output circuit and external connections.

Self Test Review

1. The basic decoder circuit is a (n) ______________________________.

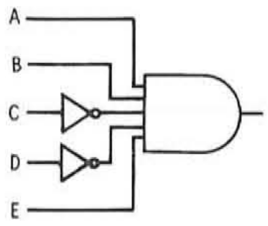


Figure 8-13. Circuit for Self Test Review question 2.

2. What is the decimal equivalent of the state being decoded by the circuit in Figure 8-13? (A is the LSB.) A B C D E
1 1 0 0 1

3. In the 7442 BCD to decimal decoder (Figure 8-4), the output of gate ______________ goes to binary ______________ when the input DCBA = 0101.

4. The maximum number of outputs from a decoder with a five bit input word is 2^5 32.

5. Draw a 1 of 4 decoder using 2 input positive NOR/negative NAND gates. Assume that both normal and complement signals are available from two flip-flops A and B (LSB).

6. Only one output of a 7442 decoder is low while all others are high.
   a. True
   b. False

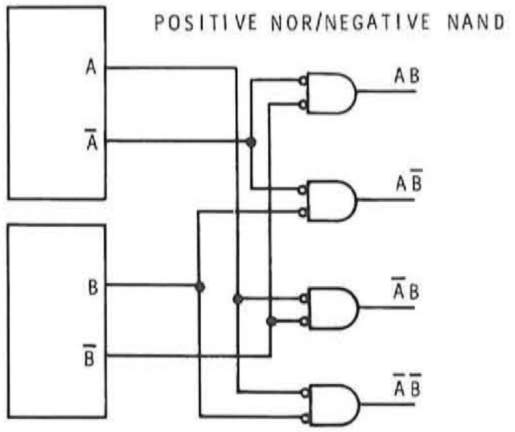


Figure 8-14. Answer to Self Test Review question 5.

Answers

1. AND gate
2. EDCBA = $10011_2 = 19_{10}$
3. gate 6, binary 0
4. $2^5 = 32$
5. See Figure 8-14
6. a. True

EXPERIMENT 18

DECODERS

OBJECTIVE

To demonstrate the operation of a decoder.

MATERIALS REQUIRED

Heathkit Digital Design Experimenter ET-3200

1 – 7400 IC (443-1)
1 – 7404 IC (433-18)
1 – 7420 IC (443-2)
1 – 7442 IC (443-53)

PROCEDURE

1. Construct the decoder circuit shown in Figure 8-15. The data switches SW1-SW4 will provide the input. LED indicator L4 is the output. Be sure to connect +5 volts and ground to pin 14 and pin 7 of the two ICs.

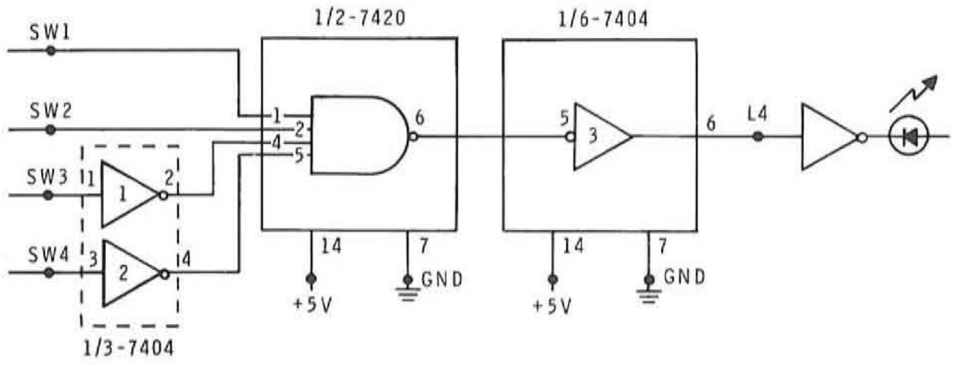


Figure 8-15. Decoder circuit for Steps 1 and 2.

2. Apply the 16 states 0000 through 1111 to the circuit and observe the output. Record the binary input state where L4 lights. 1100
Assume SW4 is the LSB. The decimal equivalent is: 12.

3. Construct the circuit shown in Figure 8-16. Take your time in constructing this circuit to avoid wiring errors. The circuit input will come from SW3 and SW4 (LSB). You will observe the outputs on LED indicators L1 through L4.

   What type of circuit is this? 1 of 4.

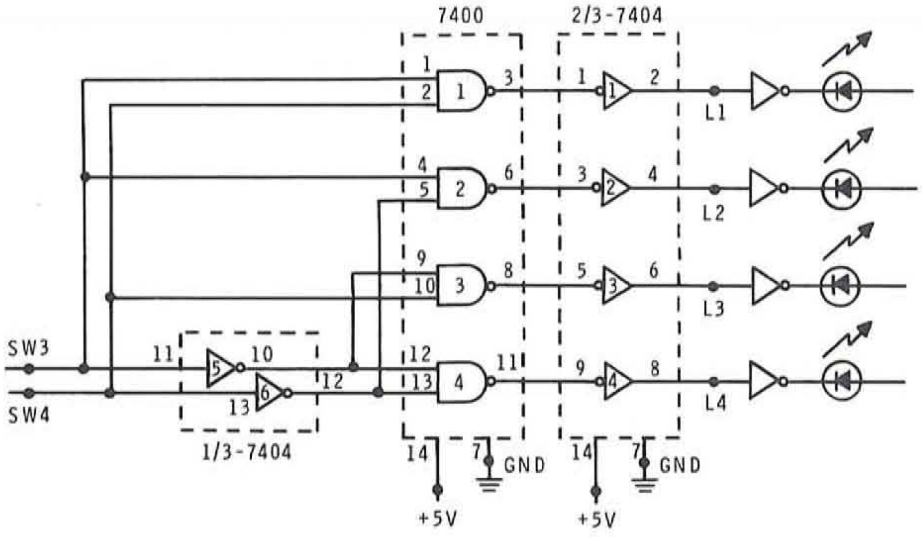


Figure 8-16.

4. With SW3 and SW4, apply the inputs indicated in Table I. Record the corresponding output states of L1, L2, L3 and L4 for each of input states.

Table I.

INPUTS		OUTPUTS (STEP 4)				OUTPUTS (STEP 6)			
SW3	SW4	L1	L2	L3	L4	L1	L2	L3	L4
0	0	0	0	0	1	1	1	1	0
0	1	0	0	1	0	1	1	0	1
1	0	0	1	0	0	1	0	1	1
1	1	1	0	0	0	0	1	1	1

Which of the following conditions did you observe for each set of inputs?

a. All outputs low.
b. All outputs high.
c. One output low.
d. Two outputs high.
e. Two outputs low.
[✓ f. One output high.]

5. Remove the connections between the outputs of inverters 1, 2, 3 and 4 and L1, L2, L3 and L4. Connect L1, L2, L3 and L4 to the outputs of the 7400 IC gates, pins 3, 6, 8 and 11 respectively.
6. Repeat Step 4. Apply the inputs in Table I and record the output states in the appropriate places.

   Which of the following output conditions did you observe for each set of inputs?

   a. All outputs low.
   b. All outputs high.
   c. One output high.
   [✓ d.] One output low.
   e. Two outputs low.
7. Compare your results from Steps 4 and 6 by observing the data in Table I. Then remove the circuit from the breadboarding socket.
8. Mount a type 7442 IC on the breadboarding socket. Connect pin 16 to +5 volts and pin 8 to ground. Connect the inputs to switches SW1 – SW4. Refer to Figure 8-17 and 8-18 for IC pin connections. You will monitor the outputs, one at a time with LED indicator L4.

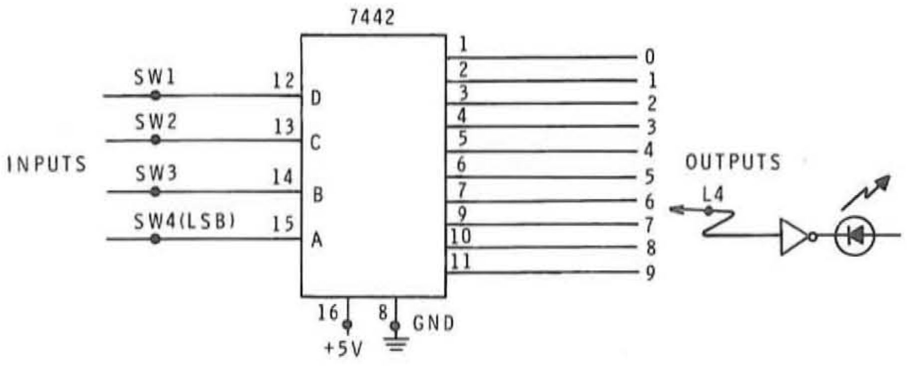


Figure 8-17. Experimental circuit for steps 8 and 9.

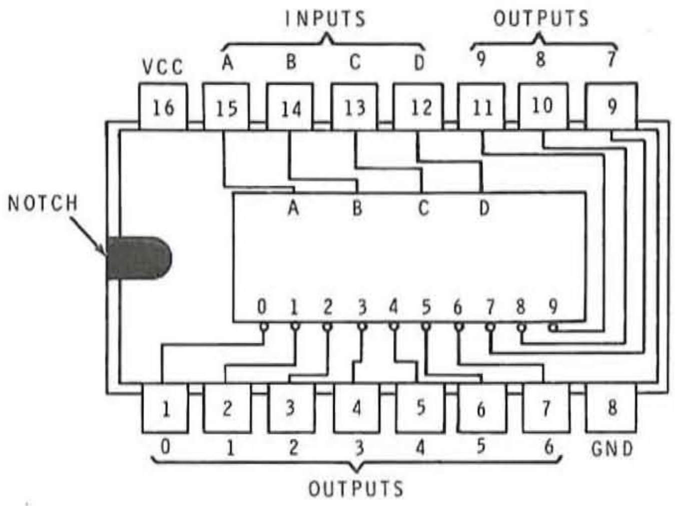


Figure 8-18. Pin connections for 7442 TTL IC BCD to decimal decoder.

9. Apply the inputs given in Table II. The LSB (A) is SW4. Observe the 10 outputs, one at a time, with L4 by connecting it sequentially to pins 1, 2, 3, 4, 5, 6, 7, 9, 10 and 11. Record your outputs in Table II.

TABLE
II

INPUTS				OUTPUTS									
D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

a. The 7442 is a one of ten decoder. What does this mean in terms of the outputs you observed?

b. The 7442 does not recognize the six states 1010 through 1111.

[✓ a. True]
b. False

DISCUSSION

In steps 1 and 2, you constructed and tested a simple decoder for a 4-bit input word. The 7420 four input NAND gate is converted into an AND gate by inverter 3 at its output. The inputs are connected so that the state 1100 (decimal 12) is decoded. When SW1 = 1, SW2 = 1, SW3 = 0, and SW4 = 0, the output will go high. For all other input codes the output will be low.

In step 3, you constructed a one-of-four decoder circuit. The 2-bit input code comes from SW3 and SW4. You observed the four possible outputs on the LED indicators.

In Step 4 you should have found that for any set of input states, only one output is high. All others are low. This proves the one of four theory. Your data in Table I should indicate the following:

0 0, L 4 on
0 1, L 3 on
1 0, L 2 on
1 1, L 1 on

Next you removed the inverters from the outputs of the NAND gates. Indicators L1-L4 monitor the NAND outputs directly with this modification. You should have found that one output was low and the other three high for any input states. The decoder is still a one of four circuit, but the selected output is low instead of high. The data in Table I should indicate:

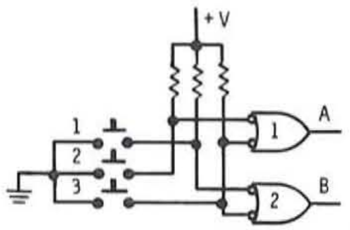
0 0, L 4 off
0 1, L 3 off
1 0, L 2 off
1 1, L 1 off

In comparing the one of four decoder with and without the output inverters, the output data of one should be the complement of the other as you would suspect. Both types of decoders are used depending upon the application. When the one of four outputs is high the decoder is said to have an active high output. An active low output indicates that the selected (decoded) one of four is low.

In Steps 8 and 9 you evaluated the operation of the 7442 BCD to decimal decoder. As you should have discovered, this circuit has active low outputs since NAND gates are used for the decoding. See Figure 8-4. Your data in Table II should correspond to the truth table in Figure 8-5. The selected output goes low. All others remain high. This circuit ignores the 1010 through 1111 states since it is an 8421 BCD (1 of 10) decoder. These six states are illegal. This is indicated by the fact when one of the six states is applied to the inputs none of the outputs go low.

ENCODERS

An encoder is a combinational logic circuit that accepts one or more inputs and generates a multi-bit binary output code. In a sense, encoders are exactly the opposite of decoders. Decoders detect or identify specific codes while encoders generate specific codes.



INPUTS	OUTPUT A	OUTPUT B
1	0	1
2	1	0
3	1	1

Figure 8-19.
Simple encoder circuit.

Figure 8-19 shows a simple encoder circuit. The inputs are three pushbuttons labeled 1, 2 and 3. The encoder circuit consists of two positive NAND/negative NOR gates. The outputs AB form a two bit binary code. When pushbutton 1 is depressed, the output of gate 2 goes high. At this time both inputs to gate 1 are high therefore its output is low. By depressing button 1, the output code 01 is generated.

Depressing the number 2 pushbutton forces the output A of gate 1 high. The B output of gate 2 is low therefore the output code is 10. Depressing button 3 forces the outputs of both gates high generating the code 11. As you can see, the binary code corresponding to the decimal number given to each input switch is generated when that switch is closed. The truth table in Figure 8-19 summarizes the operation of the circuit. When all of the switches are open (not depressed) the output code is 00.

A typical application for an encoder circuit is in translating a decimal keyboard input signal into a binary or BCD output code. Figure 8-20 shows a decimal to BCD encoder circuit. When any one of the input lines is brought low, the corresponding 4 bit BCD output code is generated. For example, bringing the number 5 input line low with a pushbutton forces the outputs of gates 1 and 3 high. Gates 2 and 4 have low outputs at this time. The output code on lines DCBA then is 0101 or the binary equivalent of the decimal number 5. This circuit, like all encoders, generates a unique output code for each individual input.

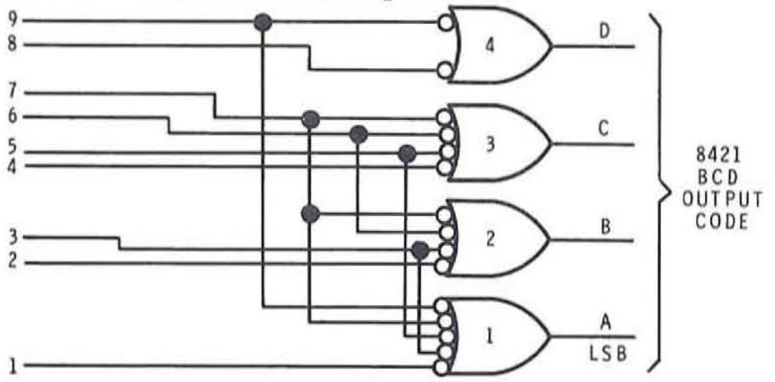


Figure 8-20.
Decimal to BCD encoder.

A typical example of a modern integrated circuit binary encoder circuit is shown in Figure 8-21. This is a TTL MSI 8 input priority encoder. The encoder accepts data from 8 input lines and generates the binary code corresponding to the number assigned to the input. The input must be brought low in order to generate the corresponding output code. We say that the inputs are active low. Unlike the two previously discussed encoder circuits, the outputs of the circuit in Figure 8-21 (labeled $\overline{A0}$, $\overline{A1}$, and $\overline{A2}$) are also active low. For this circuit what this means is that a low output represents a binary 1. This circuit generates a negative logic output code.

A unique feature of this particular circuit is that a priority is assigned to each input so that when two or more inputs are low simultaneously, the input with the highest priority is represented at the output. In this case the inputs with the higher numerical value have the highest priority. This means that if the 3 and 6 inputs are low simultaneously, the binary code representing 6 will be generated at the output.

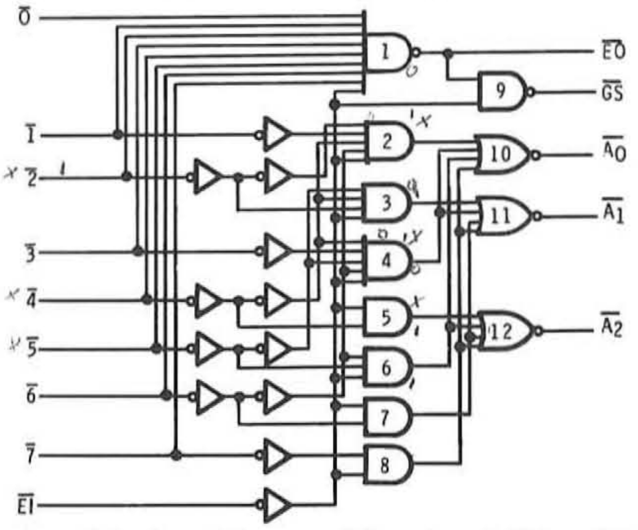


Figure 8-21.
Eight input priority encoder circuit.

The $\overline{EI}$ input on this circuit is an enabling input. When this input is high, the output of the inverter connected to it is low. This inhibits gates 1 through 8 and forces the three binary output lines high. When the $\overline{EI}$ input line goes low, the output of the inverter goes high thereby enabling all of the circuitry.

The 8 input NAND gate number 1 monitors all 8 input lines. If any one of them should go low, the $\overline{EO}$ output goes high indicating that one or more of the input lines has been activated. The $\overline{GS}$ output also goes low. If all inputs are high or open (not activated), the $\overline{EO}$ output line is low indicating this state. By using the $\overline{EI}$ input and $\overline{EO}$ and $\overline{GS}$ outputs, several of these devices may be combined to encode N different input states. A truth table for this circuit is shown in Figure 8-22.

TRUTH TABLE

INPUTS									OUTPUTS				
$\overline{EI}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$	$\overline{GS}$	$\overline{A_0}$	$\overline{A_1}$	$\overline{A_2}$	$\overline{EO}$
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	H	L	L	H
L	X	X	X	X	X	L	H	H	L	L	H	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	L	L	L	H	H
L	X	X	L	H	H	H	H	H	L	H	L	H	H
L	X	L	H	H	H	H	H	H	L	L	H	H	H
L	L	H	H	H	H	H	H	H	L	H	H	H	H

H=HIGH VOLTAGE LEVEL
L=LOW VOLTAGE LEVEL
X=DON'T CARE (EITHER 1 OR 0)

Figure 8-22. Truth table for eight input priority encoder.

Self Test Review

7. If inputs 2 and 4 on the encoder in Figure 8-20 are brought low at the same time, the output code will be
   a. 0010
   b. 0100
   c. 0110
   d. 1001

8. Answer question 7 above for the circuit in Figure 8-21.

9. Draw the logic diagram of an encoder to generate the 3 bit binary Gray code given in the table below. Use positive NAND/negative NOR circuits.

INPUT	OUTPUTS		
	A	B	C
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

10. If the 2, 4, and 5 input lines of the priority encoder of Figure 8-21 are brought low simultaneously the outputs will be
    a. $A_0 = H, A_1 = L, A_2 = H$
    b. $A_0 = L, A_1 = H, A_2 = H$
    c. $A_0 = L, A_1 = H, A_2 = L$
    d. $A_0 = L, A_1 = L, A_2 = L$

Answers

7. c. 0110 The outputs of gates 2 and 3 will go high when inputs 2 and 4 go low. In this circuit when two or more inputs are activated at the same time, the output code will be the codes produced by each input alone ORed together.
8. b. 0100 The higher numbered input has priority over the lower numbered input. Only the proper higher numbered output code is generated. This is the reason for the name priority encoder.
9. See Figure 8-23.

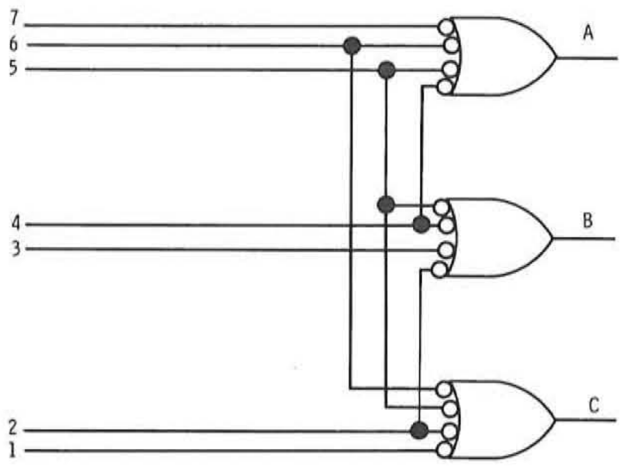


Figure 8-23. Gray code encoder answer to Self Test Review question 9.

10. c. $A_0 = L$, $A_1 = H$, $A_2 = L$
Since H = 0 and L = 1, the output code will be 101 or 5 which has the highest priority of the three inputs.

EXPERIMENT 19

7 SEGMENT DECODER-DRIVER AND DISPLAY

OBJECTIVES:

To demonstrate the operation of an integrated circuit 7 segment decoder-driver and a 7 segment LED decimal display.

MATERIALS REQUIRED

Heathkit Digital Design Experimenter (ET-3200)

1 – 7490A IC (443-7)
1 – 74193 IC (443-612)
1 – 9368 IC (443-694)
1 – 7 segment LED display (411-819)
1 – 1 kΩ resistor

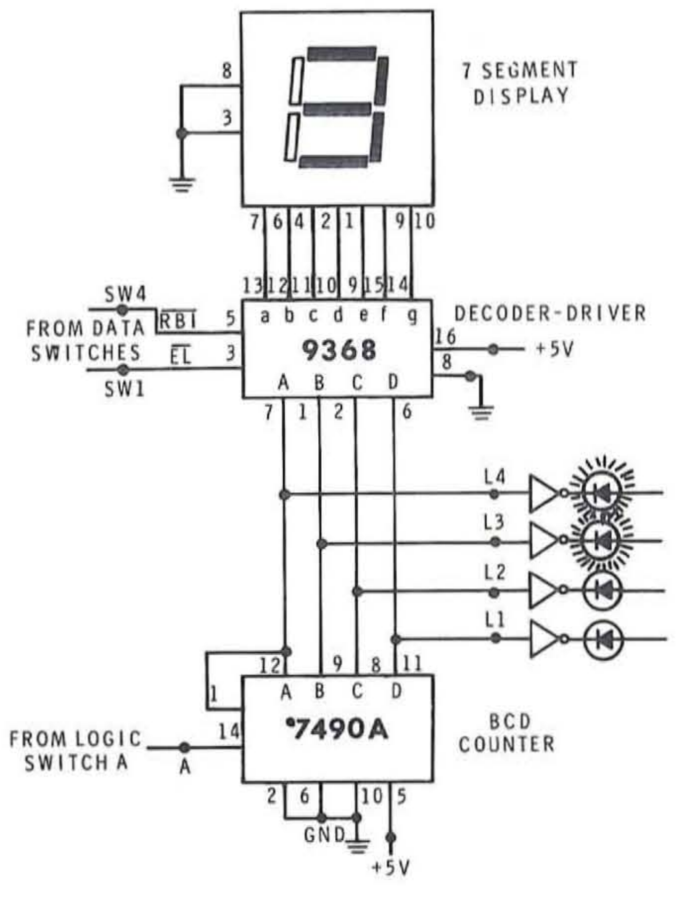
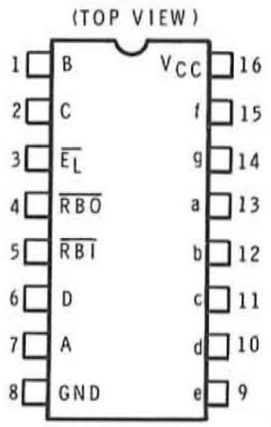


Figure 8-24. Experimental circuit for Steps 1 through 5.

PROCEDURE:

1. Construct the circuit shown in Figure 8-24. This circuit consists of a 7490A BCD counter, a 9368 BCD to 7 segment decoder-driver and a 7 segment LED display. The four BCD outputs of the 7490A counter drive the LED indicators L1 through L4 and the decoder-driver. The output of the decoder-driver is used to drive the 7 segment display. The pin connections to the 9368 decoder-driver and the 7 segment LED display are given in Figure 8-25. As before use care in wiring the circuit to prevent wiring errors. Don't forget to connect +5 volts and ground to each IC. The 9368 IC is a 7 segment decoder-driver. It accepts the four bit BCD number from the 7490A BCD counter. The 9368 IC contains a four bit latch register which can be used to store the four bit input. The output of this latch is fed to the decoder circuit that converts the BCD input into the 7 segment output code described earlier. Driver transistors in the IC provide the current necessary to operate the 7 segments of the display. The four bit latch is loaded or enabled by the EL input pin. This IC also contains a ripple blanking input line (RBI) to permit blanking of leading zeros.

A



B

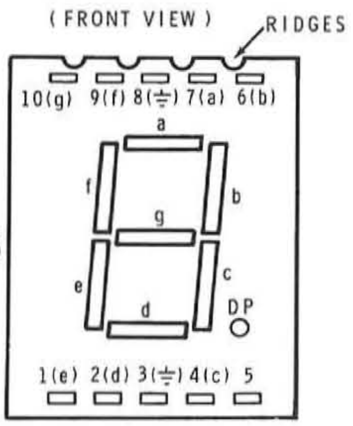


Figure 8-25. (A) Pin connections for 9368 decoder-driver IC. (B) Pin connections for type FND500 7 segment LED display.

2. Apply power to the circuit. Set data switch SW1 to binary 0 and SW4 to binary 1. Step the BCD counter with the A logic switch. As you do, note the binary LED displays L1 through L4 and the 7 segment display. Check to see that the binary number shown is equivalent to the decimal number indicated. Step the counter through its ten states several times to see that the circuit is performing properly.
3. Remove the lead connecting pin 14 of the 7490A counter to the A logic switch and connect it to the CLK output. Set the clock frequency to 1 Hz. The clock will now automatically step the counter and permit you to observe both the BCD and decimal outputs of the circuit automatically.

4. When the decimal display reads 7, quickly set SW1 to the binary 1 position. Continue to observe LED indicators L1 through L4 and the 7 segment display and note your result.

5. Put SW1 back in the binary 0 position. Observe the displays in circuit. Set SW4 to the binary 0 position. Continue to observe the displays and note any differences from the previous operation of the circuit. Specifically, observe the state of the 7 segment LED display when the counter reaches the 0000 state.

6. Modify your experimental circuit so that it appears as shown in Figure 8-26. Remove the 7490A IC and in its place install the 74193 binary counter. You will continue to use the 1 Hz clock to step the counter. Data switch SW2 will be used to reset the counter.

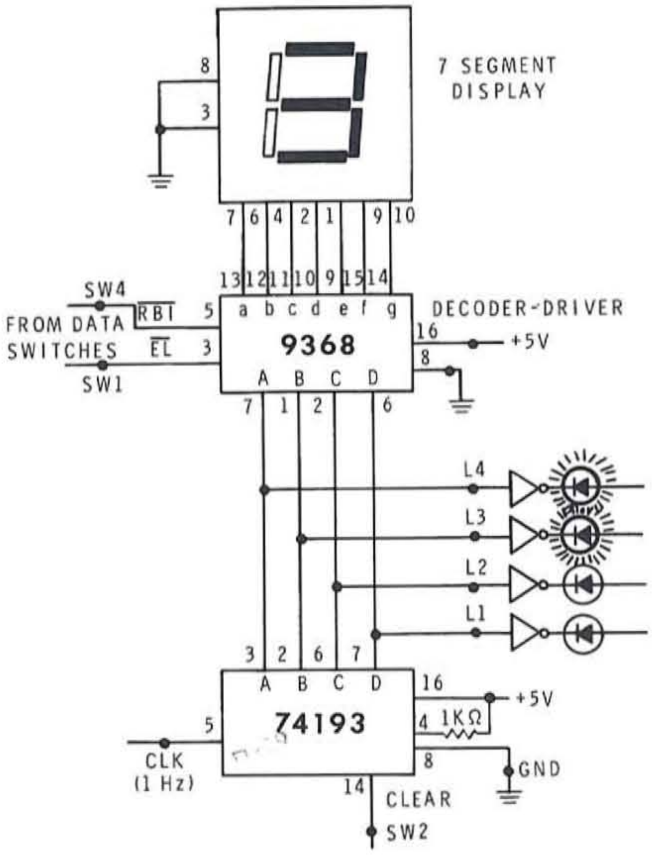


Figure 8-26.
Experimental circuit for Steps 6 and 7.

7. Set data switch SW2 to the binary 0 position. Check to see that SW1 is in the binary 0 position and SW4 is in the binary 1 position. As before the counter should change states at a 1 Hz rate as indicated by LED indicators L1 through L4 and the 7 segment display. While you are observing the LED displays, note the status of the 7 segment readout during the 6 invalid codes for BCD operation.

   Does the 9368 decoder-driver recognize the six 4 bit binary codes normally considered to be invalid in the BCD coding system? Yes.

   If your answer to the question above is yes, record the characters displayed by the 7 segment readout during these six invalid states.

1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

DISCUSSION

In this experiment you demonstrated the operation of a decoder-driver circuit that accepts a binary or BCD input code and generates the 7 segment display signals to produce the numbers 0 through 9 and other characters. In Steps 1 through 6 you used a 7490A BCD counter to drive the decoder-driver and display. This circuit counts in the standard 8421 BCD code. As you stepped the counter with the A logic switch, you should have generated the four bit BCD codes as displayed by LED indicators L1 through L4. At the same time, the corresponding decimal digit should have been displayed on the 7 segment readout. Using the 1 Hz clock signal to run the circuit permitted you to observe the outputs while the circuit stepped automatically.

In Step 4 you used data switch SW1 to set the $\overline{EL}$ input to the binary 1 state when the decimal output was 7. You should have found that the 7 segment display continued to indicate 7 while the clock continued to step the counter and display the sequential BCD states on LED indicators L1 through L4. What you did when you set SW1 to the binary 1 position was to store the number 0111 in the latch storage register of the 9368 decoder-driver. The 7 segment readout displays only the binary or BCD number stored in that internal register. By setting the $\overline{EL}$ line high you effectively inhibited the BCD inputs from the binary counter from further effecting the decoder-driver. Of course the BCD counter continued to sequence through its normal states as indicated by the changing conditions on L1 through L4. During the previous part of the experiment, you set SW1 to binary 0 condition. This enabled the latches or D flip-flops in the storage register and permitted the 7 segment outputs to follow the BCD input.

Doul

Figure 8-27 shows a simplified block diagram of the 9368 7-segment decoder/driver/latch. The 4-bit inputs are applied to the data inputs of D flip-flops that are enabled by the $\overline{\text{EL}}$ line. The outputs of these four flip-flops are fed through a one-of-sixteen decoder. The decoder outputs then drive an encoder circuit made up of OR gates that generate the 7-segment code necessary to display the digits 0 through 9 and letters A through F. The output devices are current driver transistors that supply the proper current to the segments in the driver. The ripple blanking circuitry is logic gates that are used to control a decoder and permit leading 0 suppression. The ripple blanking circuit effectively disables the gates in the decoder when the 0000 state is detected. This ensures that a 0 will not be displayed on a 7-segment readout connected to the device.

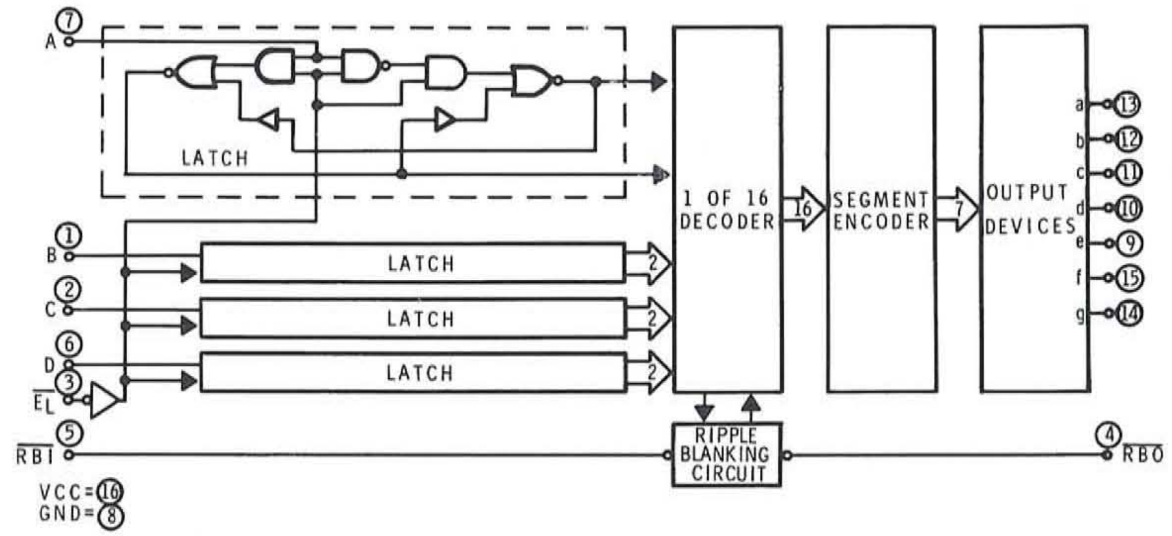


Figure 8-27. Block diagram of 9368 7-segment decoder-driver IC.

In step 5, you demonstrated the operation of the ripple blanking input. When you set SW4 to the binary 0 state, you effectively produced 0 suppression. When the counter stepped to the 0000 state, the 7-segment display should have been blank. It will not display a 0 when the $\overline{\text{RBI}}$ input is low.

Finally, you replaced the BCD counter with a standard 4-bit binary counter. In step 7, you should have found that the 9368 decoder-driver does recognize the six states normally considered to be invalid in the BCD code. In these six states, the decoder-driver causes the letters A, b, C, d, E and F to be displayed on the 7-segment display.

MULTIPLEXERS

A multiplexer is an electronic circuit that is used to select and route any one of a number of input signals to a single output. The simplest form of a multiplexer is a single pole multi-position switch. Figure 8-28 shows a rotary selector switch used as a multiplexer. Any one of six input signals can be connected to the output line by simply adjusting the position of this mechanical selector switch. Mechanical selector switches are widely used for a variety of multiplexing operations in electronic circuits. However, many applications require the multiplexer to operate at high speeds and be automatically selectable. Multiplexers of this type can be readily constructed with electronic components.

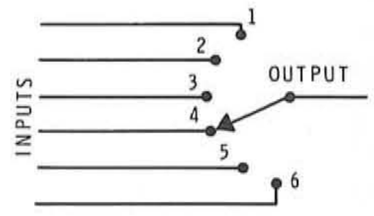


Figure 8-28. A rotary selector switch used as a multiplexer.

There are two basic types of electronic multiplexer circuits: analog and digital. The simple selector switch multiplexer in Figure 8-28 will work with either analog or digital signals. However when electronic multiplexers are constructed, they are primarily designed for either analog or digital applications. For analog applications relays and bipolar or MOSFET switches are widely used. For digital applications involving binary signals, a multiplexer can be simply constructed with standard logic gates. Our primary concern here of course is the digital multiplexer or binary data selector.

The circuit in Figure 8-29A is the simplest form of digital multiplexer. It has two input data sources and a single output. Either one of the input sources may be selected and fed to the output. The selection process takes place in AND gates 1 and 2. The flip-flop controls these two gates to determine which input is allowed to pass through OR gate 3 to the output. When this flip-flop is set, the Q output will be high enabling gate 1. The $\overline{Q}$ output will be low inhibiting gate 2. Data source 1 will therefore be allowed to pass through gate 1 and through the OR gate 3 to the output. Data source 2 will have no effect on the output state. Resetting the flip-flop reverses this condition. Gate 1 will be inhibited by Q thereby preventing data source 1 from affecting the output. However, data source 2 will be allowed to pass through gates 2 and 3 to the output. This circuit is equivalent to a single pole double throw switch as indicated in Figure 8-29B.

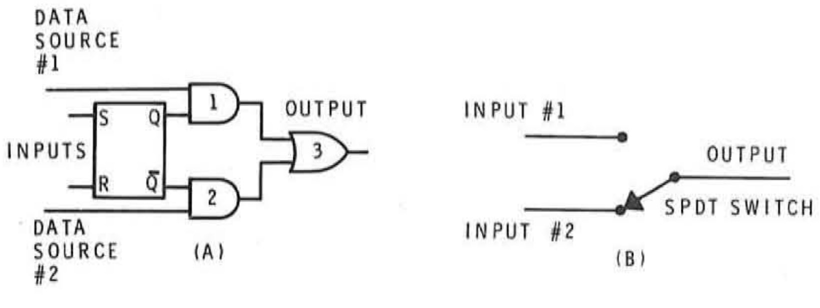


Figure 8-29. Two input digital multiplexer (A) and its mechanical equivalent (B). A SPDT switch.

A MSI functional circuit using this basic two input multiplexer is shown in Figure 8-30. Four 2 input multiplexers are combined to form a multiplexer for two four bit words. Word 1 has bits A1, B1, C1 and D1. Word 2 has bits A2, B2, C2 and D2. The enable (E) input controls the circuit. If E is high, the output of inverter 15 is low thereby inhibiting all of the AND gates and thus preventing either input word from appearing at the outputs. With E low, the circuit is enabled.

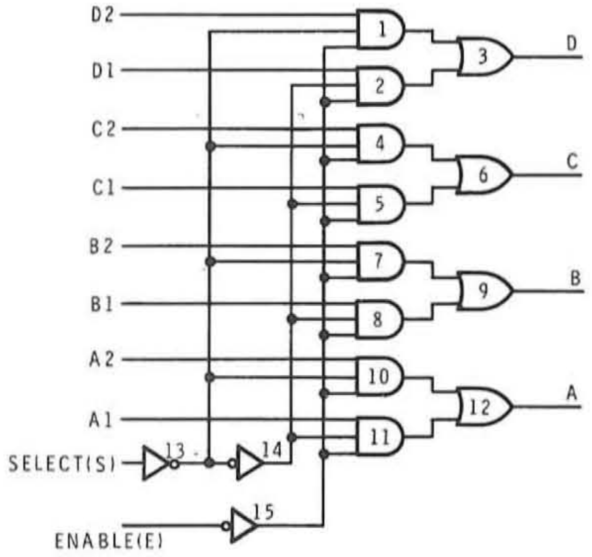


Figure 8-30.
Quad two input multiplexer.

The select (S) input specifies which four bit input word appears at the output. When the select input is high, gates 2, 5, 8 and 11 will be enabled letting input word 1 appear at the output. If the S input is low, gates 1, 4, 7 and 10 will be enabled. This permits word 2 to be passed through to the output.

A four input multiplexer circuit is shown in Figure 8-31. Each input is applied to a NAND gate that is enabled or inhibited by a 1 of 4 decoder. The outputs of the NAND gates are ORed together in gate 5. As in other multiplexers, only one of the four inputs will be enabled and allowed to pass through to the output. The selection of the input is made by the decoder circuit. A two bit binary word AB is applied to the decoder. The decoder recognizes one of the four possible input codes and enables the appropriate gate. For example, when the two bit input word is 00 the $\overline{A}\overline{B}$ output line is high. This enables gate 1 and input 1 is allowed to pass through to the output. Input code 01 enables gate 2, input code 10 enables gate 3, and input code 11 enables gate 4.

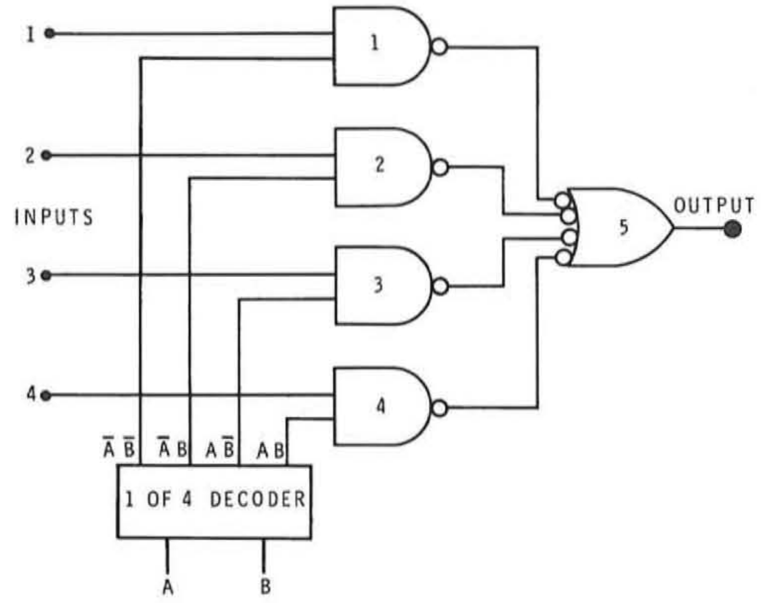


Figure 8-31.
4 input multiplexer.

The simplest way to implement the 4 bit multiplexer is to combine both the decode and enable functions in the same gate. Such a circuit is shown in Figure 8-32. The arrangement is virtually identical to the 4 input multiplexer just discussed. However, additional inputs have been added to the input gates so that they also perform the decoding functions. The normal and complement outputs from the two bit binary input word AB are applied to the enable gates in the same way they would be applied to the decoder gates. If the binary input code 00 is applied, the $\overline{A}$ and $\overline{B}$ lines will be high. Gate 1 will be enabled and input number 1 will pass through gate 1 and gate 5 to the output. Gates 2, 3 and 4 will be inhibited at this time.

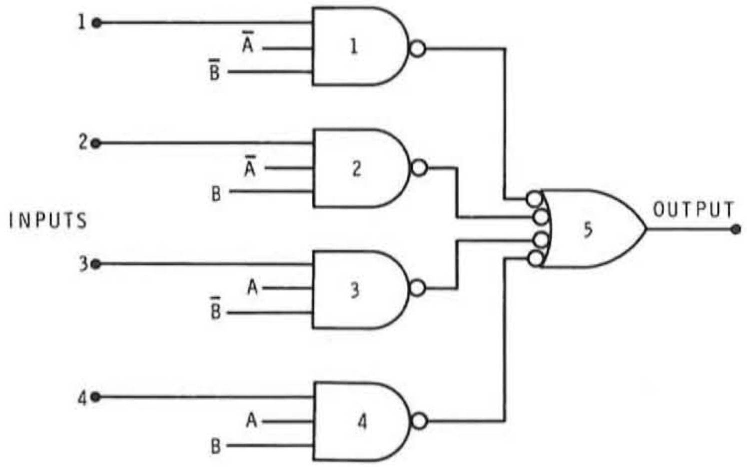


Figure 8-32. A 4 input multiplexer combining the decode and enable functions.

An eight input TTL binary multiplexer using this same technique is shown in Figure 8-33. Gates 1 through 8 enable or inhibit the eight data input lines D0 through D7. A three bit binary input word (ABC) enables one of the eight gates depending upon the input code. The six inverters at the data select inputs generate the normal and complement signals needed by the select gates. This three input word is an address code that designates which data input line is selected. If the binary input is 101, data input D5 is selected. The strobe enable line enables or inhibits all eight select gates. Both the normal (W) and complement (Y) output signals are available. Another name for the multiplexer is data selector.

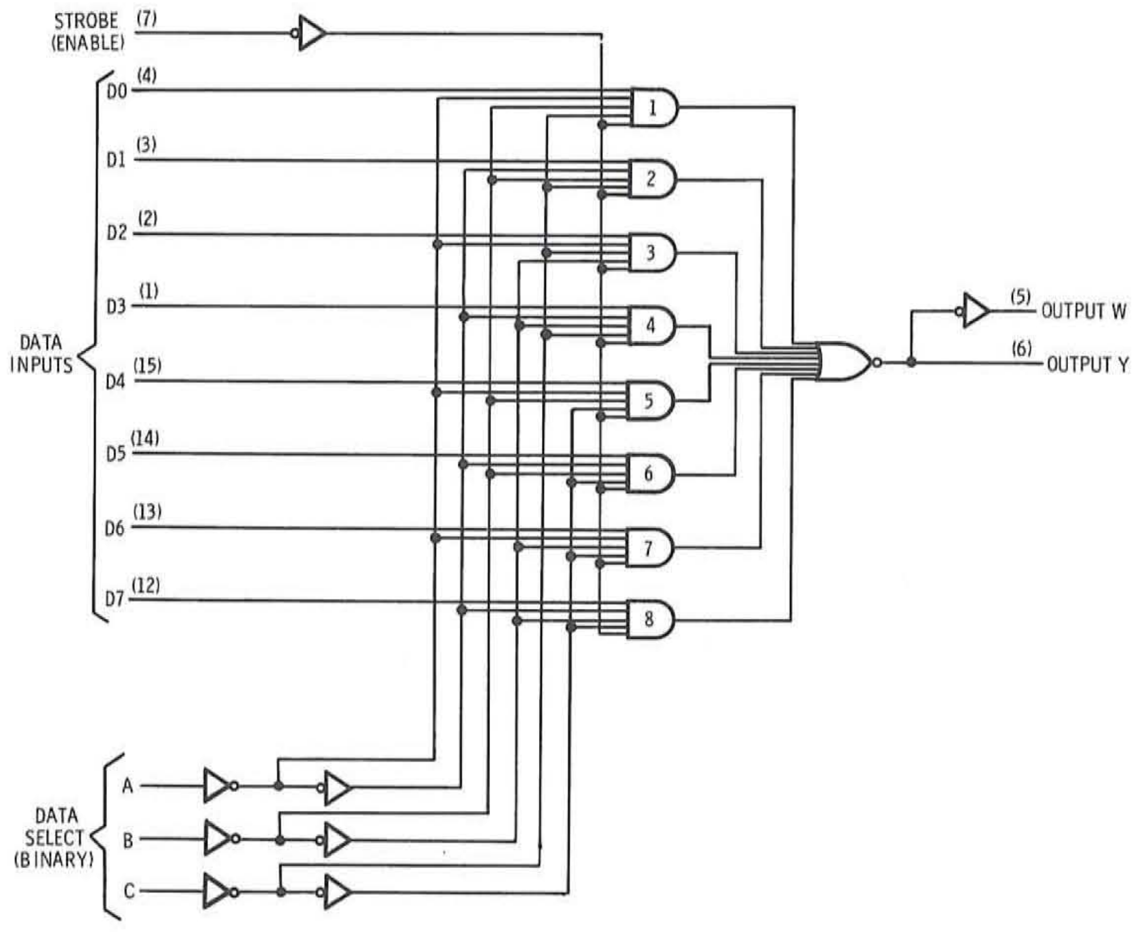
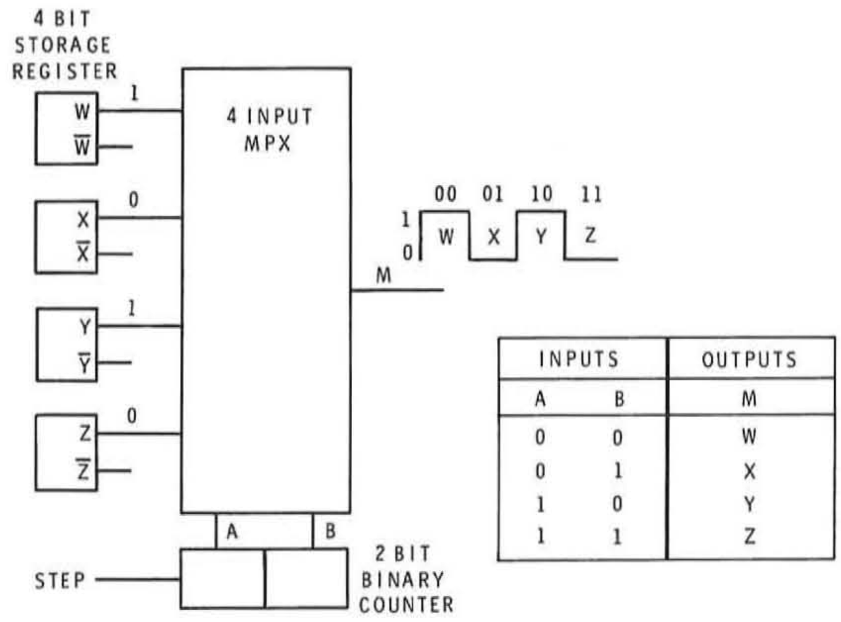


Figure 8-33. 8 input multiplexer TTL IC type 74151.

Multiplexer Applications

Besides providing a convenient means of selecting one of several inputs to be connected to its single output, a multiplexer has several special applications which make it even more useful. Besides its data selector application, multiplexers are also used to provide parallel to serial data conversion, serial pattern generation and the simplified implementation of Boolean functions. Let's consider these important applications here.

Parallel to Serial Conversion. One of the most common applications of a multiplexer is parallel to serial data conversion. A parallel binary word is applied to the inputs of a multiplexer. Then by sequencing through the input enabling codes, the output of the multiplexer becomes a serial representation of the parallel input word. This function is illustrated in Figure 8-34. Here we show a four input multiplexer. (Multiplexer is often abbreviated MPX or MUX). The simple block diagram is often used to represent multiplexers in order to simplify their illustration. A two bit binary input word AB from a counter is used to select the desired input. Input word WXYZ is stored in a 4 bit storage register. The output of each of the flip-flops in the register is connected to one input of the multiplexer. As the two bit counter is incremented, the AB input select code is sequenced through its four states 00 through 11. The output (M) of the multiplexer is equal to the state of the flip-flop connected to the enabled input. This is illustrated by the truth table in Figure 8-34. By sequencing through the four



INPUTS		OUTPUTS
A	B	M
0	0	W
0	1	X
1	0	Y
1	1	Z

Figure 8-34. Four input multiplexer used as a parallel-to-serial converter.

input states at a fixed rate, the parallel input word is converted to a serial output word. When the AB inputs are 00, the state of the W flip-flop appears at the multiplexer output. When the AB input state is 01, the state of the X flip-flop appears at the multiplexer output. Similarly, input select states 10 and 11 cause the states of flip-flops Y and Z respectively to appear at the multiplexer output. Depending upon how the inputs are connected to the register, the multiplexer can cause either the LSB or the MSB to occur first.

Serial Binary Word Generator. Another application of the multiplexer in digital circuits is the generation of a serial binary word. This application is virtually identical to the parallel to serial conversion technique just discussed. The primary difference is that for binary word generation, the serial word generated at the output of the multiplexer is generally a fixed value rather than one that can change as in the case of the parallel to serial converter. There are some occasions that require the generation of a single fixed serial word for some special function.

Figure 8-35 shows an eight input multiplexer used to generate a fixed serial binary output word. Notice that the eight inputs are connected to either +5 volts (binary 1) or ground (binary 0). The three bit input word ABC is used to select which of the inputs is routed to the output. By sequencing through the three bit input words from 000 through 111 with a binary counter, the binary states applied to inputs 1 through 8 are sequentially connected to the output. The binary word 10011010 is generated at the output. Each time the three bit input word is sequenced through the 000 to 111 state, this serial output word will be generated. Again, depending upon the application, the connections to the multiplexer input can be made such that either the MSB or LSB occurs at the output first. In this case the MSB appears at the output first. The truth table in Figure 8-35 completely defines the function of this circuit.

INPUTS			OUTPUT	
C	B	A	M	
0	0	0	0	1
0	0	1	1	0
0	1	0	2	0
0	1	1	3	1
1	0	0	4	1
1	0	1	5	0
1	1	0	6	1
1	1	1	7	0

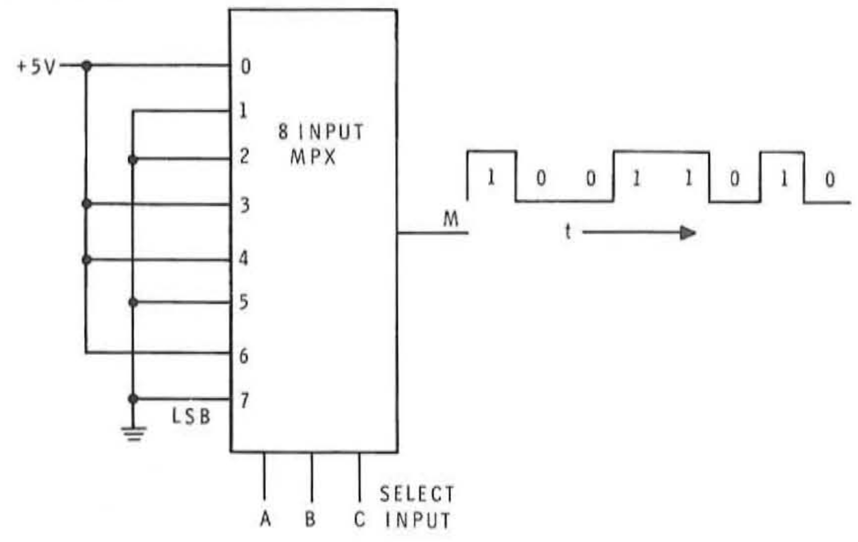


Figure 8-35. 8 input multiplexer used to generate the serial binary word 10011010.

Boolean Function Generation. Mutiplexers can greatly simplify the implementation of Boolean functions in the sum-of-products form. A close look at the multiplexer circuit in Figure 8-33 shows that it inherently implements the sum-of-products for all input combinations. The products $\overline{A}\,\overline{B}\,\overline{C}$ through ABC are developed by gates 1 through 8. By connecting a binary 1 or binary 0 to the appropriate data inputs, the products desired in the output can be selected.

For example, suppose that you wish to implement the Boolean function indicated below.

$$M = A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\,B\,C$$

By studying the logic diagram for the multiplexer in Figure 8-33, you can determine which gates generate each Boolean product. These are indicated in Table A for your convenience. Note that gate 1 generates the product $\overline{A}\,\overline{B}\,\overline{C}$. When a binary 1 is applied to the D0 input, a binary 1 will appear at the output if the data select inputs are 000. By applying a binary 0 to DØ, the 000 input state will be ignored and a binary 0 will appear at the output W.

Table A

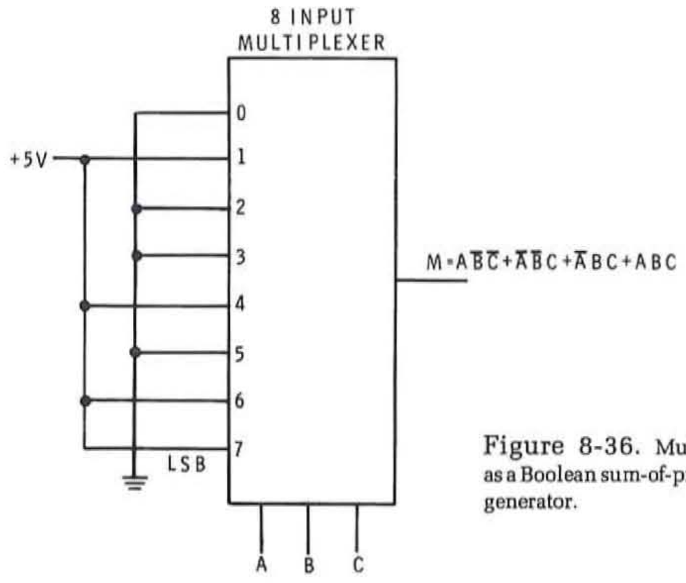
Data Input	Gate	Output
DØ	1	$\overline{A}\,\overline{B}\,\overline{C}$
D1	2	$A\,\overline{B}\,\overline{C}$
D2	3	$\overline{A}\,B\,\overline{C}$
D3	4	$A\,B\,\overline{C}$
D4	5	$\overline{A}\,\overline{B}\,C$
D5	6	$A\,\overline{B}\,C$
D6	7	$\overline{A}\,B\,C$
D7	8	$A\,B\,C$

Therefore, you can see that the desired Boolean products can be selected by applying a binary 1 to the appropriate input associated with the gate generating that product. Those products you wish to delete from the output, you apply a binary 0 to the gate generating that product.

To generate the expression indicated earlier then, binary 1 states are applied to the D1, D4 D6 and D7 inputs. These are gates 2, 5, 7 and 8 in Figure 8-33. The complete Boolean function generator is shown in Figure 8-36.

Other more complex Boolean functions can also be implemented with multiplexers by connecting other input variables to the multiplexer inputs instead of a fixed binary 1 or binary 0 level. Four variable sum-of-products from inputs A, B, C, and D for example, can be implemented with an eight input multiplexer by connecting the D and $\overline{D}$ input states to selected multiplexer inputs to implement the desired function.

By using standard MSI multiplexer packages, the implementation of Boolean functions is greatly simplified. With this technique it is not necessary to interconnect multiple SSI logic gate packages to implement the desired function. This greatly reduces the number of integrated circuits used, the power consumption, size and the need for interconnection.



8 INPUT MULTIPLEXER
+5V
0
1
2
3
4
5
6
7
LSB
$M = A\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + ABC$
A
B
C

Figure 8-36. Multiplexer used as a Boolean sum-of-products function generator.

Self Test Review

11. Which of the following definitions best describes a digital multiplexer?
    a. a circuit which can route a single input to one of several outputs.
    b. a circuit that recognizes a specific input code.
    c. a circuit that connects one of several inputs to any of several outputs.

DEMULTIPLEXERS

A demultiplexer is a logic circuit that is basically the reverse of a multiplexer. Where the multiplexer has multiple inputs and a single output, the demultiplexer has a single input and multiple outputs. The input can be connected to any one of the multiple outputs. The demultiplexer is also known as a data distributor or data router.

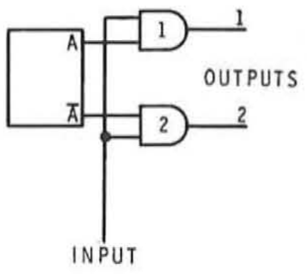


Figure 8-42
Two output demultiplexer.

A simple two output demultiplexer circuit is shown in Figure 8-42. The single input is applied to both AND gates 1 and 2. The A flip-flop selects which gate is enabled. When the A flip-flop is set, gate 1 will be enabled and gate 2 will be inhibited. The input therefore will pass through gate 1 to output number 1. Resetting the flip-flop enables gate 2 and the input is passed to output number 2.

A four output data distributor is shown in Figure 8-43. Here the single input is applied to four gates simultaneously. As in the multiplexer, additional inputs on the select gates are used for decoding. A two bit word AB from a counter is used to select which gate is enabled. If the two input binary word AB is 11, gate 4 will be enabled and the input will pass through gate 4. The other three gates will be inhibited at this time.

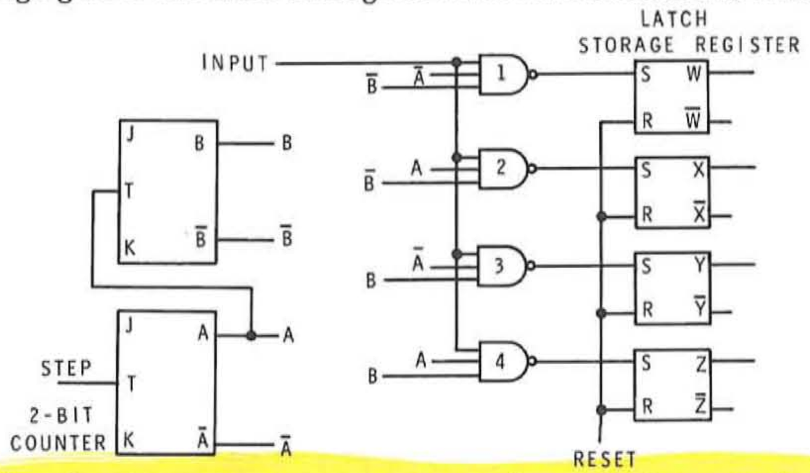


Figure 8-43. A four output demultiplexer used as a serial to parallel converter.

The data distributor shown in Figure 8-43 is being used as a serial to parallel converter. This is one typical application of a demultiplexer circuit. A four bit serial word is applied to the input. As the input bits occur, the two bit counter is incremented. This causes the gates in the distributor to be enabled one at a time, sequentially from top to bottom. The step input to the two bit counter is in synchronism with the occurrence of the bits in the serial word.

The latch storage register with flip-flops WXYZ is initially reset prior to the application of the serial input. The flip-flops in the storage register are connected to the output of the data distributor and are sequentially set or left reset as the serial word occurs. Once each of the four gates has been enabled in sequence, the register contains the serial input word. Its outputs can then be observed simultaneously. The serial input word has been converted to a parallel output word.

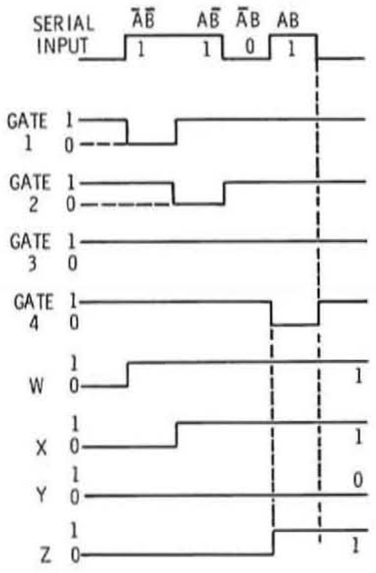


Figure 8-44. Waveforms of a serial to parallel conversion with a demultiplexer.

Figure 8-44 shows the waveforms of the circuit in Figure 8-43. The input is the serial number 1101. The waveforms show the outputs of gates 1 through 4 and the flip-flop outputs WXYZ. The first bit of the serial input is a binary 1. It occurs during the $\overline{A}\,\overline{B}$ input sequence. During this time, gate 1 is enabled and, since the input is a binary 1, its output will go low. This will set the W latch, causing the W output to go high. The $A\,\overline{B}$ input selection sequence is next. Note that this is synchronized with the next input, which is also a binary 1. This input state causes gate 2 to be enabled. Since the input signal is a binary 1 at this time, the output of gate 2 will go low, thereby setting the X flip-flop. The X output goes high as indicated. During the next input selection sequence $\overline{A}\,B$, the serial input word is 0. Gate 3 is enabled. The input is binary 0 at this time so the output of gate 3 remains high. This has no effect on the Y flip-flop so it remains reset. The AB input selection sequence is next. It occurs in synchronism with the next serial bit which is a binary 1. Gate 4 is enabled and, with the binary 1 input, its output is low. This sets the Z flip-flop, causing its

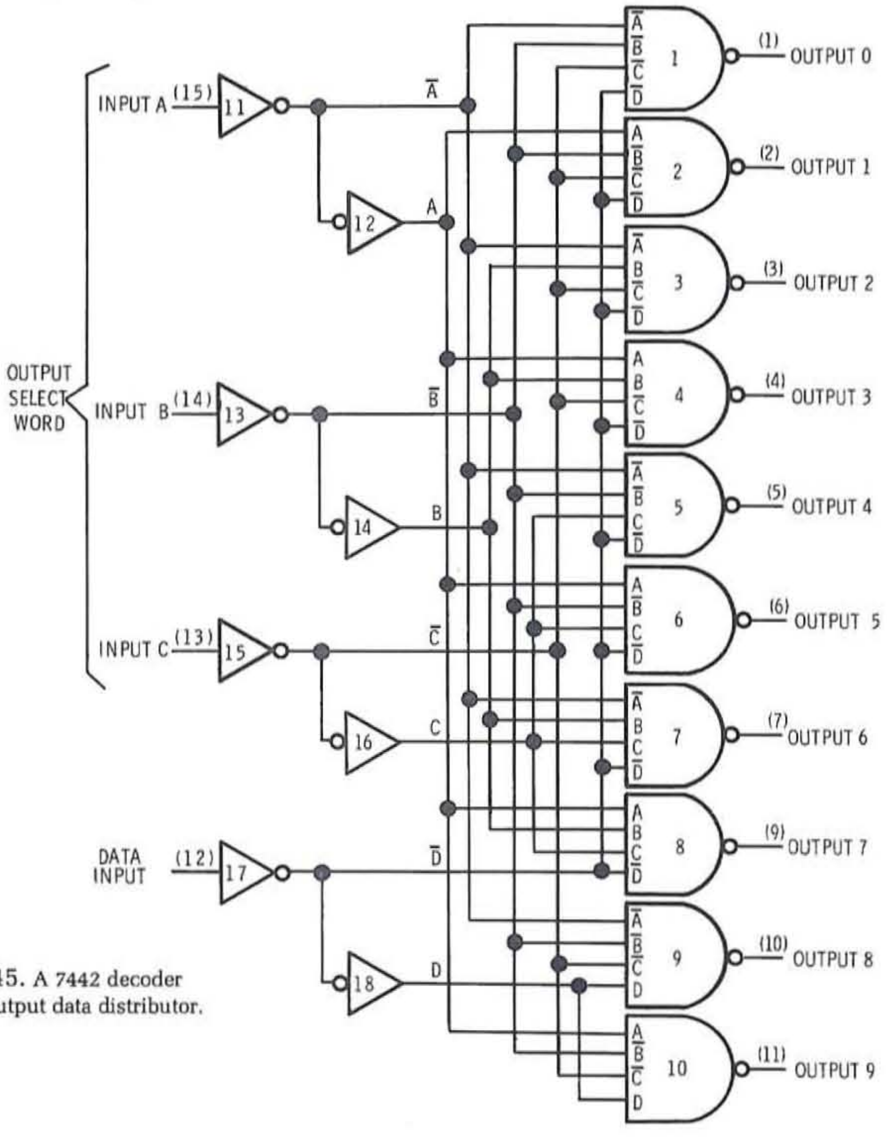


Figure 8-45. A 7442 decoder used as an 8 output data distributor.

output to go high. Looking at the states of the flip-flop after the fourth serial input bit has occurred, you can see that the parallel output 1101 is available. Note that all of the reset inputs to the latch flip-flops in the storage register are connected together to form a common reset line. Prior to the application of the serial input, a low signal is applied to the reset input to clear the register to the 0000 state.

A close look at the data distributor circuit in Figure 8-43 shows that it is essentially a decoder where the decode gates all have a common input. Because of this particular configuration, a standard MSI decoder circuit can often be used as a data distributor. Figure 8-45 shows how a 7442 BCD to decimal decoder can be used as an 8 output data distributor. When this circuit is used as a data distributor, inputs A, B and C are used to select the desired output. These three inputs will enable one of the gates 1 through 8. The data input is applied to the D input of the circuit. Note that data input is inverted by inverter 17 and then applied to gates 1 through 8. The data input will appear at the output of the gate selected by the three bit input word ABC. For example, if the input state is 000, gate 1 will be enabled. The data applied to the D input will appear at the output of gate number 1. In this application gates 9 and 10 of the decoder are not used.

Self Test Review

16. Another name for a demultiplexer is ______________________ ______________________.

17. A typical application for a demultiplexer is ______________________ ______________________.

18. In Figure 8-43, if A is set and B is reset and the input is binary 1 gate ______________________ will be enabled and latch ______________________ will be binary ______________________.

19. In Figure 8-45, what input code (CBA) must be applied to connect the input to the output of gate 6?
    a. 010
    b. 011
    c. 101
    d. 110

Answers

16. data distributor or data router
17. serial to parallel conversion
18. 2, X, 1
19. c. 101

EXCLUSIVE OR

One of the most widely used of all combinational logic circuits is the exclusive OR. It occurs so frequently in logic circuits that it is often considered to be one of the basic logic functions such as AND, OR and NOT. The exclusive OR is a two input combinational logic circuit that produces a binary 1 output when one, but not both, of its inputs is binary 1.

The standard OR logic circuit is generally referred to as an inclusive OR. The OR circuit produces a binary 1 output if any one or more of its inputs are binary 1. The exclusive OR produces a binary 1 output only if the two inputs are complementary. The table below compares the output for the standard inclusive OR and exclusive OR circuits. The inputs are A and B, the output is C.

Inclusive OR

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Exclusive OR

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

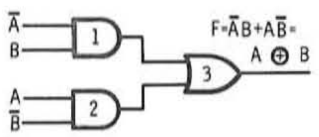


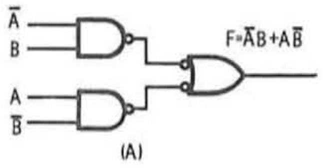
Figure 8-46.
Basic exclusive OR logic circuit.

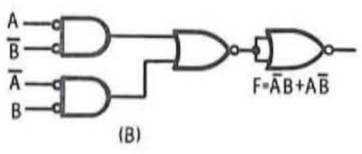
The exclusive OR logic function can be written as a Boolean expression. By using the technique you learned earlier, you can write the logic equation from the truth table. By observing the input conditions that produce binary 1 outputs, you can write the sum-of-products output. The exclusive OR function is indicated below.

$$C = \overline{A}\,B + A\,\overline{B}$$

A special symbol is used to designate the exclusive OR function in Boolean expressions. Like the plus sign represents OR and the dot represents the AND function, the symbol $\oplus$ represents the exclusive OR function. The exclusive OR of inputs A and B is expressed as indicated below.

$$C = A \oplus B = \overline{A}\,B + A\,\overline{B}$$





The exclusive OR function can be simply implemented with standard AND and OR gates as shown in Figure 8-46. The expression X-OR is often used as a short hand method for indicating the exclusive OR function.

Figure 8-47. Implementation of the X-OR function with NAND gates (A) and NOR gates (B).

Figure 8-47 shows several ways of implementing the exclusive OR function with NAND and NOR gates. The exclusive OR function implemented with NAND gates is illustrated in Figure 8-47A. The NOR implementation of the X-OR function is shown in Figure 8-47B.

Figure 8-48 shows how the exclusive OR function can be performed using the wired OR connection. Here the open collector outputs of TTL or DTL gates can be connected together to produce the OR function required by the X-OR operation.

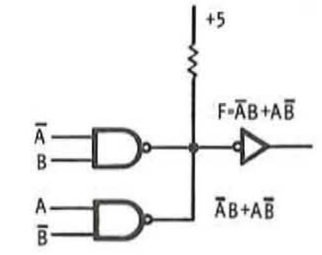


Figure 8-48. The X-OR function complemented with the wired-OR connection.

The exclusive OR circuits in Figures 8-46, 8-47, and 8-48 all assume that both the normal and complement versions of the A and B input signals are available. If they are not, then input inverters can be used to produce them. In some circuits this means extra components and a greater number of interconnections. The exclusive OR circuit in Figure 8-49 avoids this problem. Only the A and B input signals are required in order to generate the X-OR function at the output. This circuit can be readily constructed for example from a standard quad two input NAND gate such as the TTL 7400.

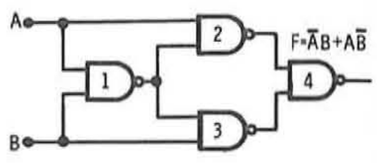


Figure 8-49. X-OR circuit not requiring complement inputs.

In order to avoid the necessity of drawing the exact logic diagram for each exclusive OR circuit used, the simplified symbol shown in Figure 8-50 is used. This symbol can be used to represent any of the exclusive OR circuits that we have described so far.

With modern integrated circuits, it is generally not necessary to actually construct exclusive OR circuits from individual logic gates. Instead exclusive OR circuits are available in MSI form. For example, the 7486 TTL integrated circuit contains four completely independent X-OR circuits.

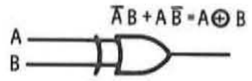


Figure 8-50. Standard symbol for an exclusive OR circuit.

Exclusive NOR

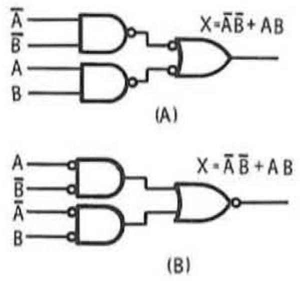
An often used version of the exclusive OR is the exclusive NOR (X-NOR) circuit. The truth table for this circuit is given below.

A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

Note that the output of the equivalent circuit is a binary 1 when inputs A and B are equal. If both inputs are 0 or both inputs are 1, the output will be a binary 1. As a result, the exclusive NOR is sometimes referred to as an equivalence circuit or a comparator. Comparing this to the exclusive OR function you can see from the truth table that the output of the X-NOR is the complement of the X-OR.

The Boolean equation of the X-NOR circuit can be written from the truth table. It is indicated below.

$$C = \overline{A}\,\overline{B} + A\,B$$



$\overline{A}$
$\overline{B}$
A
B
$X = \overline{A}\overline{B} + AB$
(A)
A
$\overline{B}$
$\overline{A}$
B
$X = \overline{A}\,\overline{B} + A\,B$
(B)

Figure 8-51. Methods of implementing of exclusive NOR or equivalence function with NAND gates (A) and NOR gates (B).

Since the form of this equation is similar to that for the exclusive OR, (sum-of-products) the equivalence function can be implemented by using any one of the exclusive OR circuits given previously by simply rearranging the inputs. Alternately, all of the previously given exclusive OR circuits can also be used to perform the equivalence operation by leaving the inputs as designated and complementing the output. Figure 8-51 shows several methods of implementing the exclusive NOR function. The simplified symbol shown in Figure 8-52 is frequently used to indicate the X-NOR operation.

Applications of the Exclusive OR

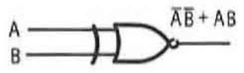


Figure 8-52. Symbol for the exclusive NOR function.

As indicated earlier there are many applications for the exclusive OR logic circuit. There are many special combinational circuits that take advantage of the special characteristics of the exclusive OR. Let's take a look at some of the most widely used applications of the exclusive OR and the exclusive NOR circuits.

Binary Adder. A binary adder is a circuit that adds two binary numbers. The output of the adder is the sum of the two input numbers. A binary adder is the basic computational circuit used in digital computers, electronic calculators, microprocessors and other digital equipment employing mathematical operations.

The basic rules for a binary addition are very simple. These are indicated below.

```
  0    0    1    1
 +0   +1   +0   +1
 --   --   --   --
  0    1    1   10   ┌carry
                ↑____┘
```

These rules indicate how two single bit numbers are added. Naturally, these rules can be extended to multibit numbers. Several examples of the addition of multibit numbers are shown below.

```
         1  carry     |          111  carries   |
  10    1010          |     7    0111           |    12    1100
+ 3    +0011          |   +11   +1011           |   +10   +1010
----   -----          |   ---   -----           |   ---   -----
  13    1101          |    18   10010           |    22   10110
```

A close look at the rules for binary addition indicated above show that if put in truth table form they would be identical to the logical function of an exclusive OR circuit. The exclusive OR inputs A and B are the two single bits to be added while exclusive OR output C is the single bit sum. As you can see, the exclusive OR is a binary adder. The only function not taken care of by the exclusive OR circuit is the carry function. When you're adding two binary 1 bits, a binary 1 carry will be generated. This carry operation can be simply implemented with an AND gate that will produce a binary 1 output only when both inputs are binary 1. Combining the AND gate and the exclusive OR we can develop a basic single bit binary adder circuit as shown in Figure 8-53. This circuit is generally referred to as a half adder.

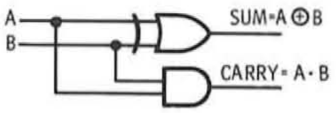


Figure 8-53.
Half adder circuit.

To add multibit numbers, we must provide an adder circuit for each of the two corresponding bits to be added. However, the half adder circuit does not provide for a carry input from a lower order bit position. Therefore, an adder circuit must be developed that will add together the two input bits then add to that sum the carry from the next least significant bit position. Such a circuit combines two half adder circuits to form a full adder. This circuit is shown in Figure 8-54. The half adder made up of exclusive OR gate 4 and AND gate number 1 performs the addition of the two input bits A and B. The output of exclusive OR gate 4 is the sum of these two bits. To this sum is added the carry input (Ci) from the adjacent lower order bit position. The sum of bits A and B is added to the carry input in the half adder circuit made up of exclusive OR gate 5 and AND gate 2. The output of exclusive OR gate 5 is the correct sum. Note that because two half adders are used there will be two carry outputs. Since a carry can be generated from either the addition of the two inputs A and B or the addition of their sum and the carry, the two carry outputs are ORed together in gate 3 to produce a correct carry output (Co) that will feed the next most significant bit adder in a multibit adder.

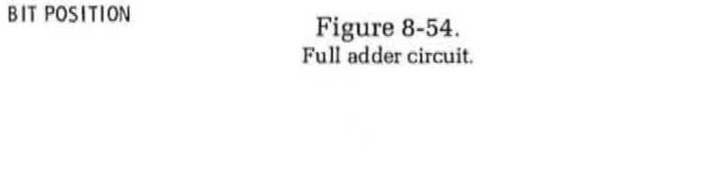

half adder
1
2
3
4
5
A
B
Ci
CARRY INPUT FROM NEXT LEAST SIGNIFICANT BIT POSITION
half adder
CARRY OUT TO NEXT MOST SIGNIFICANT BIT POSITION (CO)
SUM

Figure 8-54.
Full adder circuit.

Figure 8-55 shows a block diagram of an adder circuit used to produce the sum of two four bit binary numbers. The inputs are two four bit binary numbers A and B. Input number A is made up of bits A1, A2, A3 and A4. Input number B consists of bits B1, B2, B3, and B4. Each of the corresponding bits of the two numbers is added or summed in an adder circuit. Note that the least significant bits A1 and B1 are added in a half adder. Since there is no lesser significant bit, no carry input is required and a half adder circuit will suffice. All other bit positions require a full adder circuit to accommodate the carry input from the next lower order bit position. The output is a four bit parallel sum of the two input numbers with bits S1, S2, S3, and S4. The carry output of the most significant bit full adder also represents the fifth or most significant output bit in those situations where the four bit input numbers produce a five bit sum.

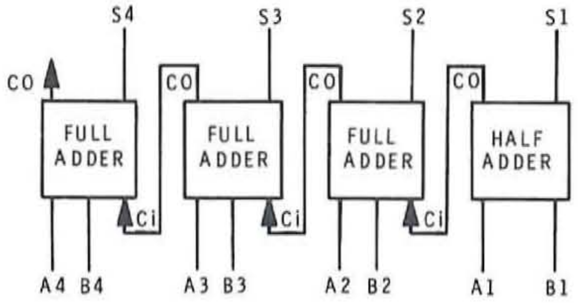


Figure 8-55.
Four bit parallel adder.

While the adder circuits described here can be constructed of exclusive OR gates and other logic elements, it is generally unnecessary since single bit adders and four bit adders like those discussed here are available as complete MSI integrated circuits in a single easy to use package.

Parity Generator/Checker. A parity generator is a combinational logic circuit that generates a single output that indicates the presence or absence of a bit error in a binary word. In digital applications requiring the storage of binary data in an electronic memory or in the transmission of binary information from one location to another, there is the likelihood of an error being made. Because of electrical noise or circuit failure, a binary 1 bit may be stored or transmitted as a binary 0. A binary 0 bit could be stored, transmitted or received as a binary 1. In most electronic equipment it is desirable to know when such errors occur. A parity generator circuit performs this function.

The parity generator circuit looks at the binary word to be stored or transmitted and generates a single output known as a parity bit. This parity bit is then added to the other bits of the word and stored or transmitted with it. When the stored word is retrieved from memory for use or when a transmitted word has been received, a parity check operation is performed. The parity checker generates a parity bit from the received data and compares this bit with the parity bit stored or transmitted with the original information. If the two parity bits are identical, no error exists. A difference in parity bits designates an error.

The method of generating a parity bit is to observe the binary word to be stored or transmitted and determine the number of binary 1's in that word. A parity bit will be generated based on this information such that the total number of binary 1's in the word including the parity bit will be either odd or even. The table in Figure 8-56 shows all 16 possible combinations of four bit binary words. The odd and even parity bits for these words are designated in the adjacent columns. Note that for odd parity, a binary 1 or binary 0 parity bit is added to make the total number of bits in the word, including the parity bit, odd or even.

TABLE I

A	B	C	D	ODD PARITY	EVEN PARITY
0	0	0	0	1	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	0

Figure 8-56. Odd and even parity bits for a four bit word.

The basic circuit element used to generate the parity bit is the exclusive OR circuit. A look at the truth table for an exclusive OR circuit will reveal that it is basically an odd or even detector circuit. If the two inputs are equal or even, the exclusive OR output is a binary 0. But if the inputs are odd or complementary, the output is a binary 1. The exclusive OR gate then can be used to compare two binary bits and indicate whether they are odd or even, equal or unequal. An exclusive OR gate then is used to monitor each two bit group in a binary word. These exclusive OR outputs are then further compared with other exclusive OR circuits until a single output bit indicating odd or even is generated.

Figure 8-57 shows how exclusive OR gates are cascaded or pyramided to produce a parity generator circuit. A 4-bit binary number input to the parity generator circuit is stored in a register made up of flip-flops A, B, C and D. X-OR gate 1 monitors bits A and B while X-OR gate 2 monitors bits C and D. The outputs of these two X-OR circuits are then monitored by X-OR gate 3. The result is an even parity output bit. Inverter 4 generates the complement or the odd parity output. Using your knowledge of the exclusive OR circuit and the information in Table I, trace the various binary states from the flip-flop outputs to the output of the parity generator circuit to be sure that you fully understand its operation.

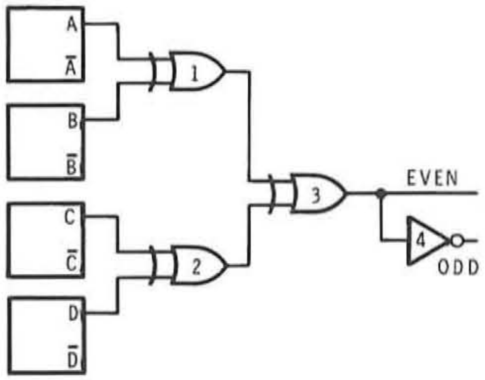


Figure 8-57.
A 4-bit parity generator circuit.

A parity bit for any size binary word can be generated by simply using as many exclusive OR gates as necessary to monitor all input bits. Additional exclusive OR gates are then used to monitor the output states of the exclusive OR gates used in monitoring the input bits. This cascading or pyramiding of exclusive OR gates is continued until a single output bit is generated.

Once the parity bit has been generated, it is generally stored or transmitted along with the input word. When the binary word and its parity bit are read from memory or received at the remote location, it can be tested for bit errors in a parity checker circuit. A parity checker consists of a parity generator circuit identical to those just discussed. This parity generator looks at the stored or received word and again generates a parity bit. This bit is then compared to the parity bit stored or transmitted along with the word. This comparison takes place in another exclusive OR circuit. Figure 8-58 shows a parity checker circuit for a 4-bit binary word with parity bit.

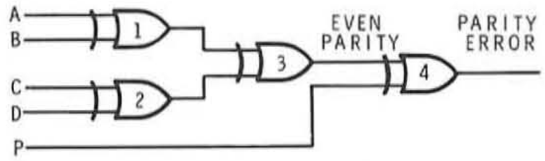


Figure 8-58.
A 4-bit parity checker.

Note that the parity generator circuit consists of exclusive OR gates 1, 2, and 3 and is identical to the parity generator circuit in Figure 8-57. Exclusive OR gate 4 compares the output of the parity generator with the received parity bit P. In this circuit we are assuming the use of even parity. If the internally generated parity bit is the same as the received parity bit, the output of the exclusive OR circuit will be binary 0 indicating no parity error. However, if the two parity bits are different, the exclusive OR output will be binary 1 indicating a parity error.

The output of a parity checker circuit can then be used in a variety of ways to indicate the occurrence of a parity error. It can be used to turn on an indicator light indicating an error state. It can be used to initiate a series of logic operations that will either accept or reject the data depending on the error state. Or it may be desirable simply to count and record the number of parity errors that occur.

As you probably realize, a parity error detection method does not ensure complete freedom from or knowledge about all possible error conditions. The parity technique assumes that an error will occur in only one bit position of a word. If parity errors occur in two bit positions, it is possible for the word to be transmitted incorrectly while no parity error will be indicated. This situation rarely occurs since in most electronic storage and transmission systems the reliability is sufficient to eliminate the possibility of multibit errors. However, errors in a single bit position are common. The parity detection and checking process is a very reliable and useful indication of errors.

Even more sophisticated combinational logic circuits have been developed to detect when more than one bit error is produced. In systems requiring ultra high reliability and performance, such sophisticated circuits can be used to detect and even correct any bit error that occurs. In some high speed computers, multiple bit errors are automatically detected and corrected before the information is processed.

While parity generator and checker circuits can be constructed with individual exclusive OR gates, they are also available in MSI form. Figure 8-59 shows a typical commercial parity generator/checker MSI circuit. This circuit is capable of performing either the generation or checking function. As a generator, it can monitor up to nine input bits. The ninth bit is applied to either the odd or even input as required by the application. Both odd and even parity bit outputs are provided. In the checking function, this circuit will monitor an 8 bit word and generate an appropriate parity bit which is then compared with a received parity bit applied to either the odd or even input. The error indication appears at the odd or even output depending upon whether the odd or even parity convention is used.

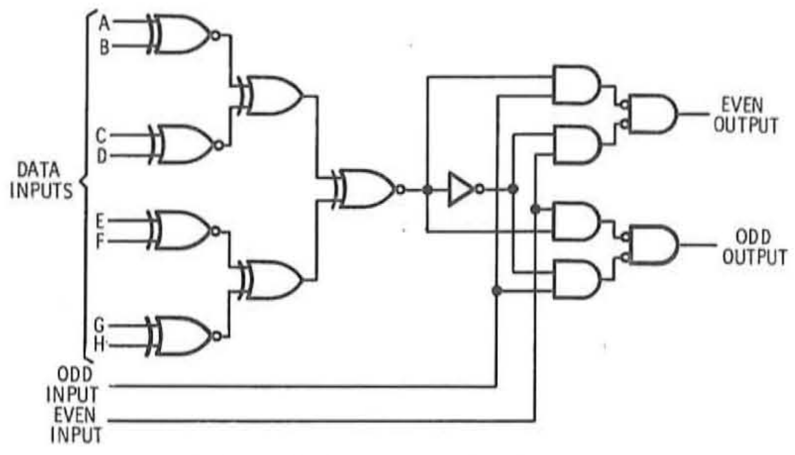


Figure 8-59. A commercial MSI parity generator/checker IC.

Binary Comparators. A binary comparator is a combinational logic circuit that looks at two parallel binary input words and generates a binary 1 output signal if the two numbers are equal. If the numbers or words are not the same, the output will be a binary 0.

As you saw earlier, the exclusive NOR circuit is essentially a single bit binary comparator. When the two inputs are alike, the output is binary 1. When the input bits are different, the output is binary 0. By using an exclusive NOR circuit for each pair of bits in the two numbers to be compared, a complete binary comparator circuit can be constructed.

Figure 8-60 shows a four bit binary comparator circuit. Word 1 with bits A1, A2, A3, and A4 are stored in the A register. The word to be compared is stored in the B register with bits B1, B2, B3, and B4. Each pair of bits is applied to an exclusive NOR circuit. The outputs of the exclusive NOR's are fed to a four input AND gate. When the two binary words are alike, the outputs of the exclusive NOR's will be binary 1. With all binary 1 inputs to the AND gate, the output will be binary 1 indicating the equality of the two words. If any one or more bits of the input words are different, the output of the related exclusive NOR circuit will be binary 0. Naturally, this will inhibit the AND gate and produce a binary 0 output which indicates inequality. Additional exclusive NOR gates and AND gates inputs can be added as required to compare any size binary number.

Commercially available MSI binary comparators are available thus eliminating the need to assemble such circuits from individual gates. Typically, these comparators are designed for comparing two four bit binary words. In addition to providing an output that indicates the equality of the two words, most MSI comparators also generate two additional output signals, one indicating when one word is greater than the other and another indicating when one word is less than another. Figure 8-61 shows a block diagram of such a comparator. If input word A has a binary value that is numerically larger than word B, the A greater than B output (A > B) will be binary 1. The A < B output will be binary 1 if A is less than B.

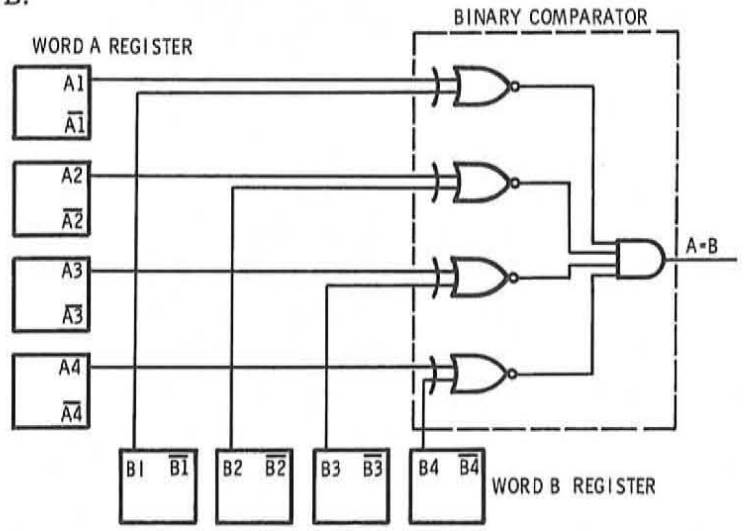


Figure 8-60.
Four bit binary comparator.

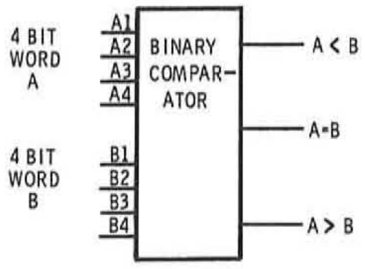


Figure 8-61. Typical 4 bit commercial MSI binary comparator circuit.

Self Test Review

20. When the inputs to an exclusive OR circuit are alike, the output is binary ____________________.

21. Prove that the circuit in Figure 8-49 does perform the exclusive OR function. Write the output equation of the circuit using the NAND relationship for each gate. Then using Boolean algebra and De Morgan's theorem, reduce this expression to the exclusive OR formula.

22. Using Boolean algebra and De Morgan's theorem, show that the complement of the exclusive OR function is the equivalence function. Or prove that :

$$\overline{\overline{A}\,B + A\,\overline{B}} = \overline{A}\,\overline{B} + A\,B$$

23. An exclusive NOR gate is also a(n)
    a. adder
    b. comparator
    c. subtractor
    d. decoder

24. The expression $\overline{A \oplus B}$ indicates the
    a. exclusive OR
    b. inclusive OR
    c. exclusive NOR

25. Add the following binary numbers.

a.		b.		c.	
	011		11111		1010
	+101		+10001		+1011

26. Write the odd parity bit code for the XS3 BCD code.

27. How many exclusive NOR's are needed to make a comparator for two six bit words?
    a. 2
    b. 3
    c. 6
    d. 12

Other frequently used code converters are binary to Gray and Gray to binary. There are many applications where the Gray or cyclical code must be used in order to minimize errors when changing from one state to another. While the Gray code is good for minimizing errors in generating certain types of data, the Gray code cannot be used in arithmetic operations. To permit arithmetic operations to be performed, Gray to binary code conversion is necessary.

Both Gray to binary and binary to Gray code converter circuits are shown in Figure 8-68. The Gray to binary circuit is shown in Figure 8-68A while the binary to Gray circuit is shown in Figure 8-68B. The most significant bit of both the Gray and binary words will be the same and therefore no code conversion is necessary. Note the use of exclusive OR circuits to perform the code conversion. The Gray and equivalent binary codes are shown in Figure 8-69.

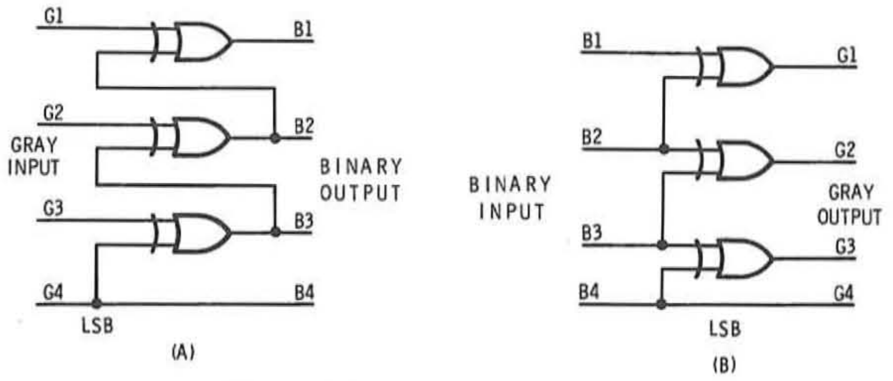


Figure 8-68. Gray to binary (A) and binary to Gray (B) code converters

DECIMAL	BINARY				GRAY			
D	B4	B3	B2	B1	G4	G3	G2	G1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Figure 8-69.
Binary and Gray codes.

While code conversion is most often accomplished with combinational logic circuits, many types of code converters use sequential circuits. Various combinations of flip-flops, counters and shift registers can be used to perform code conversion. A simple example is the serial Gray to binary code converter shown in Figure 8-70. Here a serial Gray code is applied to the JK inputs of a JK flip-flop, MSB first. The normal flip-flop output is a serial binary code.

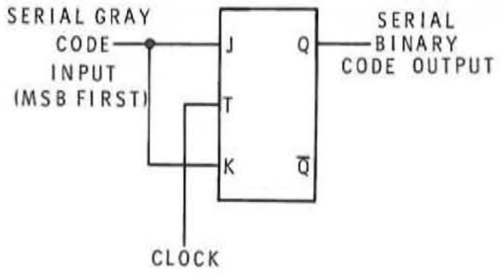


Figure 8-70. Serial Gray to binary sequential code converter.

Self Test Review

28. The most commonly used code converters use the ______________ and ______________ codes.

29. Code converters can be either combinational or sequential logic circuits.
    a. True
    b. False

30. Code converters can process either serial or parallel data.
    a. True
    b. False

31. Most code converters are
    a. sequential
    b. combinational

Answers

28. binary, BCD
29. a. True
30. a. True
31. b. combinational

permit presetting of the register from the data switches. You depressed the A logic switch to generate the clock pulse that loads the 7495 register with the number 0111. You then returned the mode control to binary 0 so that the register would shift right.

Next, you set the number 0101 into the data switches. The 74151 multiplexer converts this parallel word into a serial word as it is sequenced by the two bit binary counters made up of a 7476 dual JK flip-flop IC.

Next, you used the A logic switch to generate shift pulses that cause the addition to occur. The number in the 7495 shift register is shifted out into the adder. These same pulses increment the binary counter and sequence the multiplexer. The sum is stored in the 7495. The correct sum should have been 1100.

The binary adder in this circuit operates just like the full adder circuit discussed previously in the unit. The only difference is the addition of a JK flip-flop from a 7476 IC to use as a carry memory. Since we are adding a single bit at a time, some means must be provided for remembering the occurrence of a carry so that it can be added to the next most significant bits in sequence.

The JK flip-flop stores this carry. Note that the output of one gate in the 7400 IC is labeled carry-out (Co). This is applied to the JK inputs. Assume that the LSBs of the numbers to be added are applied to the adder. The carry output line at this time will have on it a binary state that indicates the presence of a carry bit if the states of the input bit are such that it produces a carry. Assuming that a carry is produced, this carry signal will be loaded into the JK flip-flop when the first clock pulse occurs. At the same time, the adder is generating the proper binary sum of the two LSBs. This is applied to the serial input of the 7495 shift register. Therefore, when the first clock pulse occurs, the first sum bit is loaded into the 7495 shift register and the carry state is stored in the JK flip-flop.

The least significant bits have now been added and lost. The next most significant bits now appear at the adder inputs. The proper sum is generated and appears at the serial input of the 7495 shift register. At the same time, the carry state from the previous two bits stored in the JK flip-flop is applied to the carry input (Ci) of the adder. This ensures that the previously generated carry is added to the next two most significant bits. The proper sum then is generated and is loaded into the shift register when the next clock pulse occurs. Also during the next clock pulse time the carry state of the presently monitored bits is stored in the JK carry flip-flop. This action is continued until all four bits have been added. At this time the proper sum is in the 7495 shift register.

READ ONLY MEMORIES

A read only memory (ROM) is an electronic circuit used to permanently store binary information. Practical read only memories are available for storing as many as 65,000 bits of data. Normally the ROM is organized to store equal length multibit words. For example a typical ROM might be capable of storing 512 eight bit words or 4096 bits.

The main feature of a read only memory is that the binary information contained in the memory is permanently stored there. The data is written into the memory when it is manufactured. The contents of the memory cannot usually be changed thereafter. This is in contrast to many electronic read/write memories that can both store and read out data. This type of memory is generally referred to as random access memory (RAM). You can think of it as many storage registers.

The general organization of both read/write memories and read only memories is basically the same. Both contain a number of memory locations where data can be stored. In the ROM, the data is stored there permanently and can be read out in any order. In the RAM, data may be written into or read out of any portion of the memory at any time. While the read/write memory is more flexible, it is also more expensive. It is this type of memory that is normally used as the main storage section of a digital computer. Our discussion here will center on the ROM which is a useful circuit in implementing digital systems.

ROM Operation

Figure 8-76 shows a general block diagram of a read only memory. It consists of three major sections: the address decoder, the memory storage elements, and the output circuits.

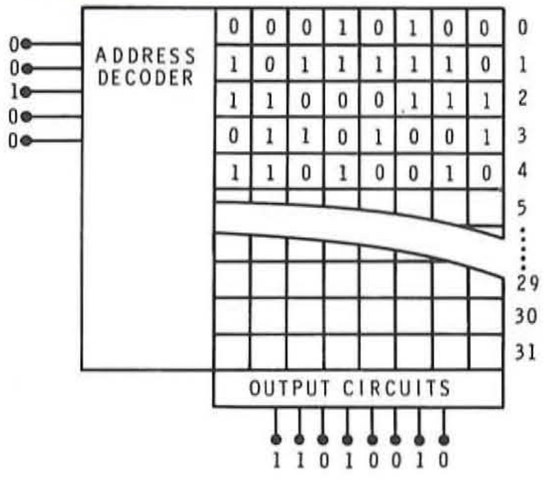


Figure 8-76. General block diagram of a read only memory.

The address decoder is like any binary decoder in that it accepts a multibit binary input word and decodes all possible input states. Only one of the decoder outputs will be activated. In the address decoder of Figure 8-76, there are five input bits meaning that a total of $2^5 = 32$ different states can be decoded. This five bit input word specifies one of 32 individual memory locations. This input word is generally referred to as the address.

The main body of the memory consists of electronic circuits or components that are used to store the binary data. These storage elements are arranged so that a specific number of multibit binary words may be stored. The organization in Figure 8-76 permits 32 eight bit words to be stored. The memory locations are designated 0 through 31. Applying a five bit address code to the input will cause the contents of the addressed location to appear at the output. Note that if the address input code is 00100, the contents of memory location 4 appear at the output. All other memory locations are ignored at this time. The output circuits buffer the memory contents so that the data can be used in other logic circuits.

ROM Construction

There are many different ways to implement read only memories with electronic components. Any component or circuit capable of storing a binary 1 or binary 0 condition can be used. Magnetic cores and capacitors are examples of elements that have been used to store binary data in a ROM. Most modern read only memories, however, are semiconductor circuits. Both bipolar and MOS types are used. Since ROMs are capable of storing a significant amount of data they are generally classified as large scale integrated (LSI) circuits. Most ROMs, both bipolar and MOS types, are housed in standard dual in-line packages. Because of the wide variety of possible applications, ROMs are considered to be custom circuits. The user specifies the memory contents prior to the manufacture of the device. In this section, we investigate the most popular types of integrated circuit ROMs in use today.

Diode Matrix ROM. Figure 8-77 shows a read only memory constructed with a one of eight decoder and a diode matrix. The one of eight decoder accepts a three bit address input word and generates all possible decode output combinations. This means that the decoder will recognize the three bit input number applied to it and enable only one of the eight outputs. For example, if the binary input number is 011, the number 3 output line will go low. All other decoder output lines will be high at this time. The decoder is similar in operation to the type 7442 discussed earlier.

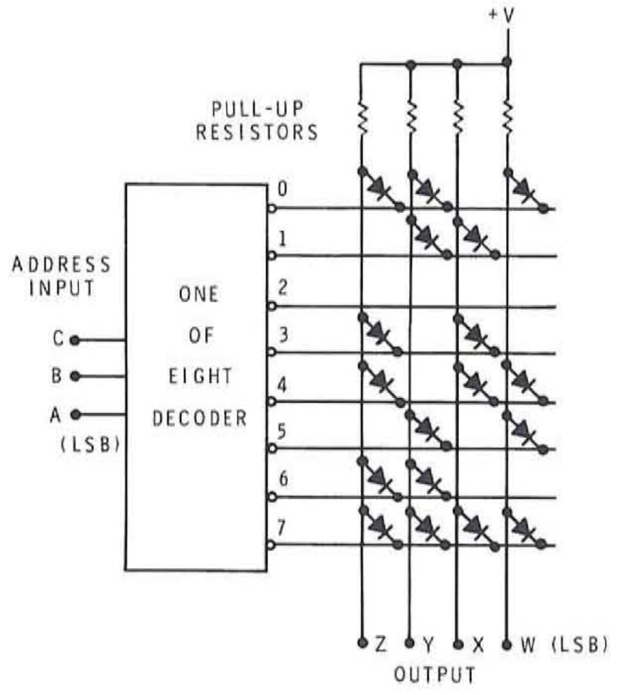


Figure 8-77. A ROM constructed from a decoder and diode matrix.

When the number 3 decoder output line goes low, it brings the cathode ends of the diodes connected to this line low. The diodes conduct through their associated pull-up resistors. This forces lines X and Z low. Lines W and Y are high at this time because of the pull-up resistors. Since all other decoder outputs are high, the other diodes in the circuit are cut off at this time. Observing the output lines ZYXW then you see the output code 0101. At address location 011 (3), the binary number 0101 is stored.

Consider the effect of applying the address 110 to the decoder input. This will bring decoder output 6 low causing output lines Y and Z to go low. Lines W and X will be high at this time. This means that the output number is 0011. The contents of memory location 110 (6) is the four bit number 0011.

A close look at the ROM in Figure 8-77 should reveal that the data is stored in the memory as either the presence or absence of a diode. In this circuit a diode connection between the decoder output and the output line causes a binary 0 to be read out when that address line is enabled. The absence of a diode causes a binary 1 to be read out. Another way to look at the read only memory is to consider each output line with its associated diodes and pull-up resistors as a diode OR gate. A low on any diode input causes the output to go low.

There are some commercial read only memory ICs designed and constructed exactly like that shown in Figure 8-77. Integrated circuit ROMs are constructed initially so that a diode is connected at each possible memory location. This means that all memory locations are initially programmed with binary 0's. To store data in the memory, an external pulse signal is applied to the output lines in such a manner as to reverse bias certain diodes and cause them to be destroyed. By destroying a diode and causing it to open, a binary 1 state is programmed. Such ROMs can be programmed by the manufacturer or the user. Read only memories that permit the user to store the data that he needs are called programmable read only memories (PROMs).

PROMS

RAM

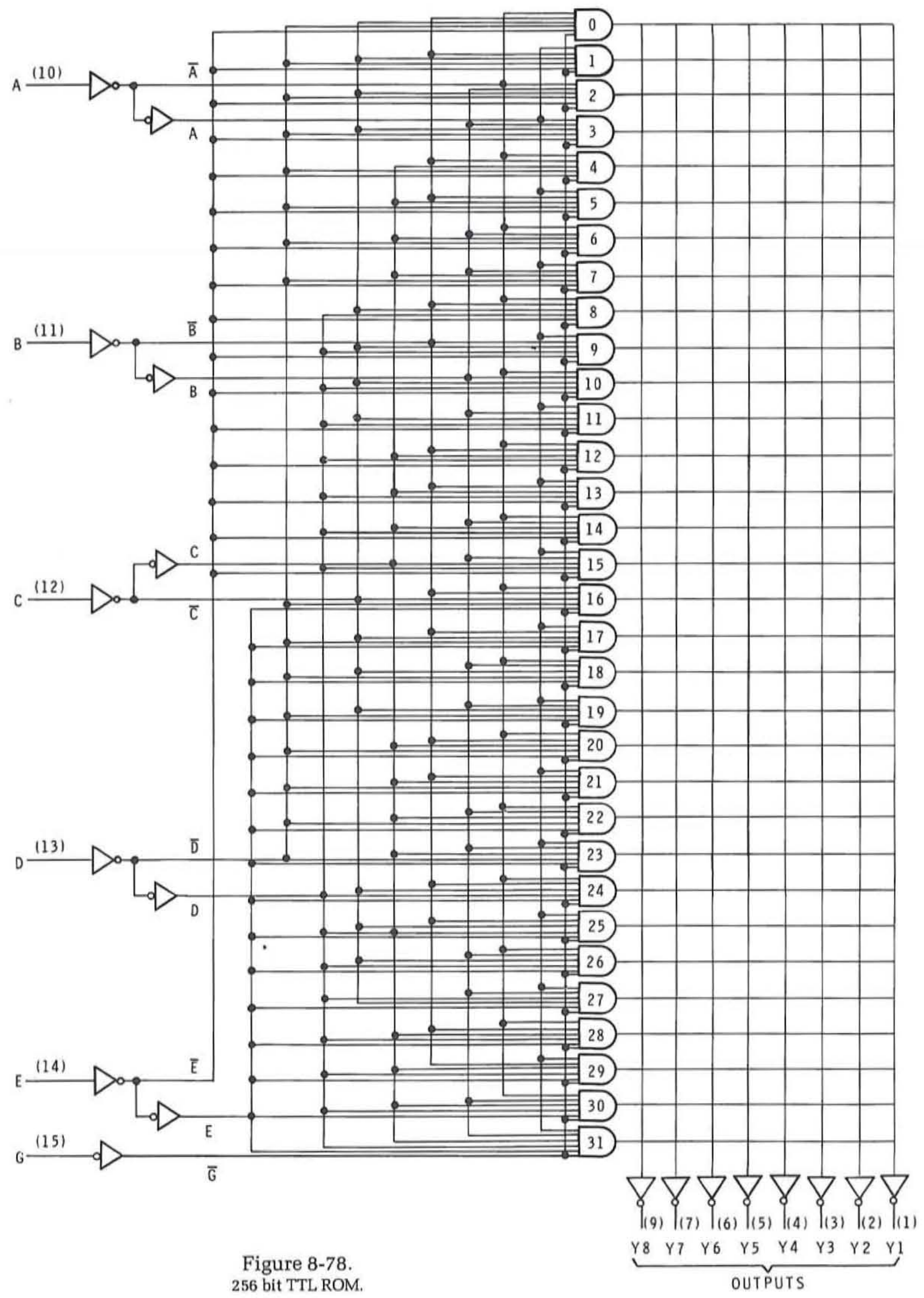


Figure 8-78.
256 bit TTL ROM.

Bipolar ROM. A typical commercial bipolar read only memory is shown in Figure 8-78. This 256 bit ROM uses TTL circuitry. The memory is organized as 32 eight bit words. The address inputs labeled A through E are used to select one of the 32 words stored in the memory. Note that the circuitry is basically a 1 of 32 decoder.

Figure 8-79 shows the detailed circuitry of the ROM. Illustrated here is one of the 32 address decoding gates and the 8 output buffer circuits. The output of each decoding gate is a transistor with eight emitters. These emitters can be interconnected to the eight output buffers. The programming of the memory is done by either connecting or leaving open these emitter connections. If an emitter is connected to an output buffer, the output voltage will go low when that decoding gate is addressed. If the emitter is not connected, a high level voltage is read out of the associated buffer when the gate is addressed.

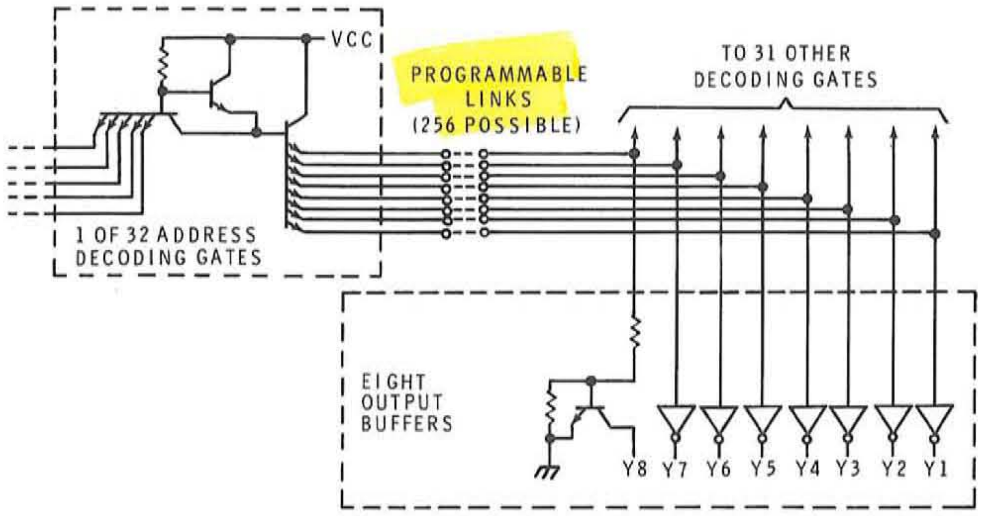

VCC
PROGRAMMABLE LINKS (256 POSSIBLE)
TO 31 OTHER DECODING GATES
1 OF 32 ADDRESS DECODING GATES
EIGHT OUTPUT BUFFERS
Y8
Y7
Y6
Y5
Y4
Y3
Y2
Y1

Figure 8-79.
TTL ROM circuit details.

The decoding gate output emitters to be used are connected to the respective inputs of the eight output buffers when the integrated circuit is manufactured. The user specifies the memory contents, and the manufacturer produces special masks which will cause the interconnecting metalization of the integrated circuit to be properly arranged to store the desired data. Note that the output buffers have an open collector output. This permits the outputs to be wire-ORed with other similar memories so that the storage capability can be expanded. Three state output circuitry is used on some TTL ROMs. Input line G on the ROM in Figure 8-78 is used to enable or disable the circuitry so that this device can be combined with others similar to it in forming a memory with many more locations. This line is often referred to as a chip select line and is used as an extra address bit input in expanded memories.

Figure 8-80 shows two ways that a standard size ROM can be used to make larger memories. Figure 8-80A shows two ROMs connected to form a memory for 32 sixteen bit words. Each ROM can store 32 eight bit words as indicated by the designation 32 x 8 or 32 by 8. The five address lines are in parallel thus each ROM is enabled at the same time. One half of the 16 bit word is stored in the upper ROM and the other 8 bit segment in the lower ROM. Since the two ROMs are addressed simultaneously, both parts of the word will be read out at the same time. Input line G enables the memory when low.

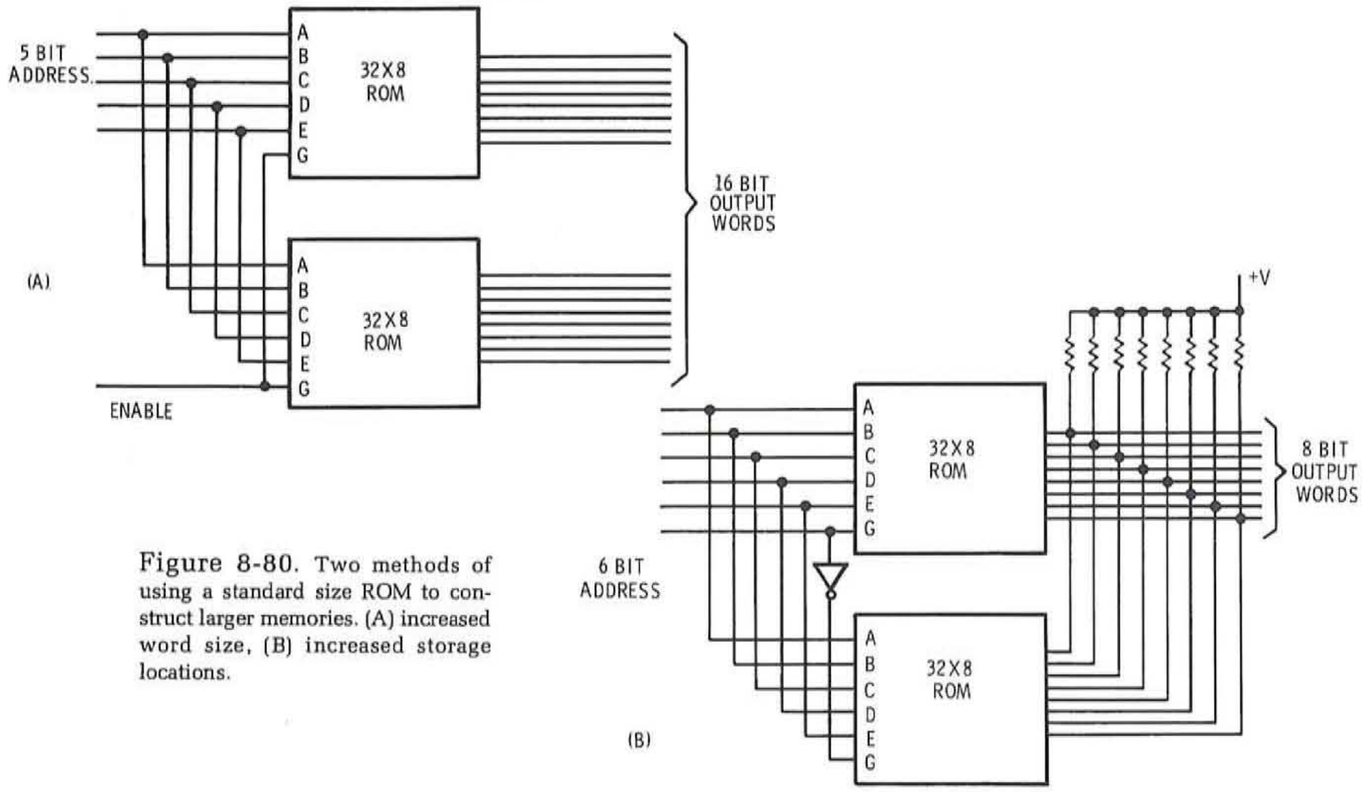


Figure 8-80. Two methods of using a standard size ROM to construct larger memories. (A) increased word size, (B) increased storage locations.

Figure 8-80B shows how the 32 x 8 ROMs can be used to form a memory for storing 64 eight bit words. Thirty-two of the words are in the upper ROM and the other thirty-two are in the lower ROM. The ROM outputs are wire-ORed. The five address lines A through E are in parallel. The chip enable lines (G) are used as a sixth address line. Remember, it takes six bits to address 64 words ($2^6 = 64$.). This sixth input line is the MSB of the address. The five lower order bits address both ROMs simultaneously. But only one of the two ROMs will be enabled by input G. The inverter keeps these lines complementary.

If the input address (GEDCBA) 101101 is applied, location 13 (01101) in each ROM will be addressed. However, input G is binary 1. This disables the upper ROM so all of its output lines are high. Input G to the lower ROM is low because of the inverter. Therefore, this ROM is enabled and the word at location 01101 is read out.

MOS ROMS. Many read only memories are implemented with metal oxide semiconductor integrated circuits. MOSFET circuitry lends itself well to the implementation of a read only memory. Because of the small size of most MOSFET circuits, many logic and memory elements can be constructed in a small space. This high density circuitry permits read only memories with a very high bit content to be readily manufactured. Many thousands of bits of data can be stored on a silicon chip approximately 1/10 inch square. Such MOS ROMs are low in cost and consume very little power.

The basic organization and structure of an MOS ROM is essentially the same as any read only memory. An address decoder selects the desired word. The presence or absence of a semiconductor device in a matrix network specifies a binary 1 or binary 0 stored in the addressed location. In MOS ROMs, the basic storage element is an enhancement mode MOSFET. ? The presence of an MOSFET programs a binary 1. The absence of such a device means a binary 0 has been programmed.

Figure 8-81 shows the basic internal structure of a typical PMOS ROM. P type material is diffused into the substrate in long strips called bit lines as indicated. These P-type diffusions form the source and drain connections

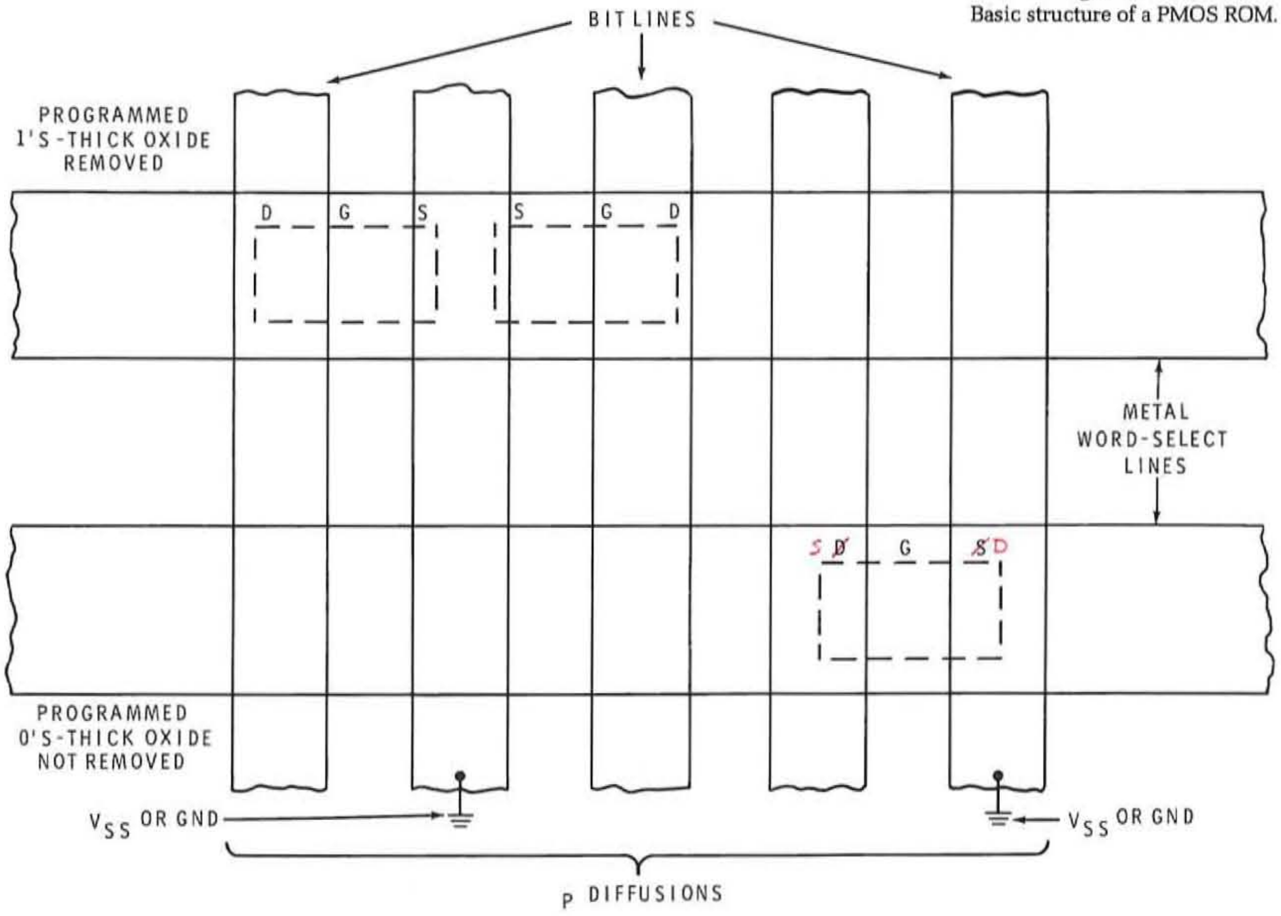


Figure 8-81.
Basic structure of a PMOS ROM.

of the MOSFETs. Perpendicular to the P diffusion areas are metal word select lines. These metal areas form the gate elements of the MOSFETs. Figure 8-81 shows several examples of how the MOSFETs are formed. The source (S), gate (G), and drain (D) of each MOSFET is identified. To program the memory, the MOSFETs formed by this structure are either enabled or disabled by appropriate masking operations during manufacturing. As indicated earlier, if the MOSFET is enabled, a binary 1 will be stored in that location. Disabling the MOSFET causes a binary 0 to be stored in the selected location.

In the MOSFET ROM structure, the metal word select lines are connected to the gates of the MOSFETs where binary 1s are stored. These metal word select lines are driven by the outputs of a decoder. The source terminals of the MOSFETs are connected either to ground or to the source supply voltage V_{SS}. The drain connections of the MOSFETs are designated as the bit lines. If one of the metal word select lines goes negative, the MOSFETs associated with that word will conduct and ground (or V_{SS}) will appear on the bit line.

Figure 8-82 shows the MOSFET ROM circuit. Q1 is a MOSFET formed by the process illustrated in Figure 8-81. The presence or absence of this transistor is a function of the masking process carried out during manufacturing. Note that the gate of Q1 is enabled by the output of decoder X. If the word select line is negative, Q1 will conduct and a binary 1 bit will appear on the bit line. However, this binary 1 may or may not reach the output of the ROM depending upon the state of Q2. Q2 and decoder Y are also used in selecting the desired output word.

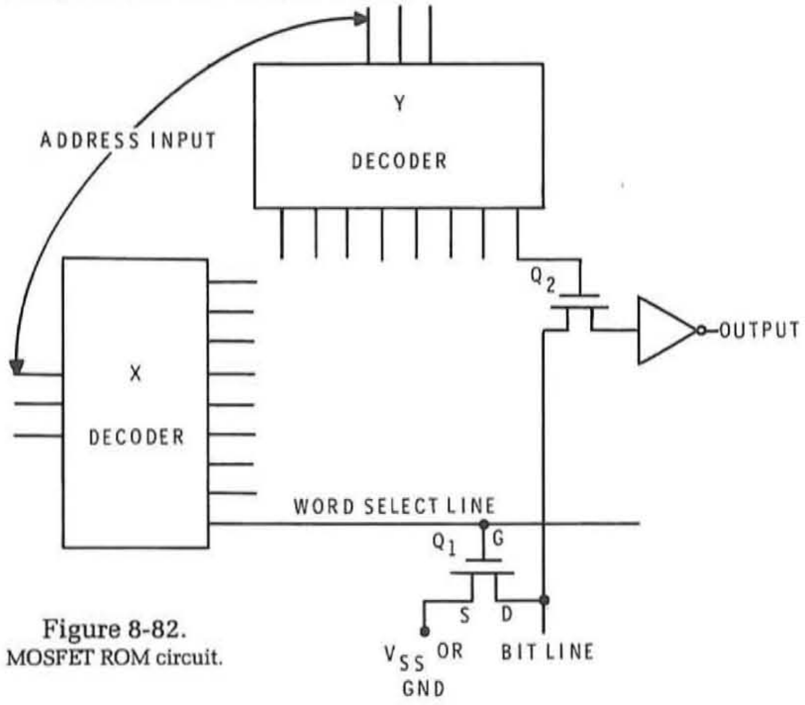


Figure 8-82.
MOSFET ROM circuit.

Most MOS ROMs use an XY matrix decoding method. In Figure 8-82, two 1 of 8 decoders are illustrated. Two 3 bit words are used to address a particular word. The two 3 bit input numbers are simply treated as a single six bit address. Six bits define $2^6 = 64$ bit locations. By using two 1 of 8 decoders, a total of 64 words can be addressed. The word in memory is selected by enabling each decoder with the appropriate three bit word. If the Y decoder enables Q2, Q2 will conduct and connect the bit line to the output buffer. If the decoder does not enable Q2, the output on the bit line shown will not appear at the output despite the fact the word select line may have enabled Q1.

Access Time. Like any logic circuit, a ROM has propagation delay. This means that there is a finite time between the application of an input address and the appearance of data at the output. This propagation delay is referred to as access time. This is the time it takes to find a word in the ROM and read it out. For bi-polar ROMs, this access time is usually less than 100 nanoseconds and can be as low as 20 nanoseconds. For MOS ROMs, the access time is typically several hundred nanoseconds.

ROM Applications

The ROM is extremely versatile in implementing logic functions. An appropriately programmed ROM can often be used to replace many different types of combinational circuits. It is particularly useful in replacing complex logic functions with multiple inputs and outputs. The ROM offers the advantages of faster and easier design, lower cost, smaller size and often lower power consumption.

Combinational logic circuits generate output signals that are a function of

1. the input states
2. the types of gates used and
3. the particular unique interconnection of these gates.

The desired output states for a given set of inputs are produced by properly interconnecting the correct types of logic gates. This same logical function can be simulated by a ROM. The desired inputs are applied to the ROM address lines. These inputs specify a unique memory location. In this memory location is a binary bit pattern whose output states duplicate those produced by an equivalent combinational logic circuit. Instead of actually generating the desired output function with a logic circuit, we store the desired output states in the memory and read them out when the proper inputs appear on the address lines.

A ROM performs what is known as a table look up function. All of the memory locations can be considered to be entries in a large table of numbers. By applying an address to the ROM, we are in effect looking up one unique number in the table. In a sense, the ROM does not perform a logic operation. The desired output states for a given set of input conditions are simply stored in the memory.

The following examples will illustrate some of the many applications of a ROM.

Random Logic. A read only memory can be used to quickly and easily implement any random logic function involving multiple inputs and multiple outputs. To design such a combinational logic circuit with standard logic gates, you first develop a truth table that defines the operation to be performed. From the truth table the Boolean equations are then written. Boolean algebra is then used to minimize these equations. From the equations, the logic circuit is developed and then implemented with standard NAND gates, NOR gates and inverters.

When a read only memory is used, the only design step is the implementation of the truth table. The truth table defines the inputs and outputs. This is all of the information that is necessary to develop a read only memory that will perform the desired logic function. The input logic states are assigned as addresses of a read only memory. In the memory locations corresponding to the addresses are stored binary words that cause the output lines to assume the desired states with the given input address. By using a ROM you can go from truth table to finished logic circuit in one simple step. Design time is reduced considerably.

If the function to be implemented involves only a few inputs and a few outputs, such a circuit is best implemented by conventional means with logic gates. However, if the number of inputs and outputs is four or more, the use of a ROM becomes practical. Since a ROM costs more than standard SSI logic circuits, it is not practical or economical to use a ROM where very simple functions must be implemented. Four inputs and four outputs are generally regarded as the decision point between a read only memory vs. a conventional logic circuit.

Code Conversion. As we indicated earlier, code conversion refers to any multi-input/multi-output combinational logic circuit. A code converter is nothing more than a special application of such a logic circuit. Since read only memories can readily replace multi-input/multi-output combinational logic circuits, a ROM provides a simple and low cost means of code conversion.

To use a ROM as a code converter, the input code is made equal to the binary address code in the ROM. In the memory location specified by the input or address code is the desired output code. No complex logic functions must be implemented to achieve this result. The desired output codes are simply stored in the memory locations and are read out when the equivalent input code is applied. All of the most commonly used code conversion processes mentioned earlier have been implemented with read only memories.

Arithmetic Operations. Arithmetic operations are some of the most difficult functions to implement with digital circuitry. Simple combinational logic circuits have been developed to perform additions and subtractions. Various algorithms have been developed for using addition and subtraction along with other digital operations such as shifting to perform multiplication and division. More complex mathematical functions such as the trigonometric and logarithmic functions are even more difficult to implement. The read only memory provides a very simple and direct method of implementing the more complex arithmetic operations.

The multiplication of two binary numbers requires a significant amount of logic circuitry. While there are numerous methods for carrying out multiplication, all of them require an extensive amount of circuitry. Multiplication can be performed with a read only memory without the need for complex circuitry or high cost. In addition, the ROM can provide this function at lower cost, in a smaller space and at a significantly higher speed.

The truth table shown in Figure 8-83 illustrates the concept of multiplying two binary numbers using a ROM. To simplify the explanation we will use only two bit binary numbers. Multiplying two binary numbers produces a product whose length is twice that of one of the input numbers. The two bit binary numbers serving as the multiplier and multiplicand will form a four bit product. The two input numbers are grouped so that they form a single four bit binary input number which serves as the address input to the ROM. At the address formed by the input numbers, the correct four bit product corresponding to the two bit numbers is stored. When any combination of the two bit input numbers appears on the ROM address lines, the correct product is read out. For example when one input is 10 (2) and the other is 11 (3), the input address formed is 1011. At this location is the number 0110 (6) which is the product of 2 and 3. By using a larger ROM, numbers requiring more bits can be used.

INPUTS				OUTPUTS			
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Figure 8-83.
Truth table for 2 bit ROM multiplier.

A ROM is particularly useful in handling complex mathematical operations such as the trigonometric and logarithmic functions. Instead of having the digital circuitry actually compute the sine, cosine or tangent of a number, the ROM simply stores the trigonometric functions corresponding to the angles. In the same way, a ROM can be used to store the logarithms of specified input numbers. In these applications, the ROM is virtually a log table or trig function table. The desired angle or input number is applied to or assigned a binary address that is applied to the ROM. At the address representing the desired input angle or number is stored the correct trigonometric function or logarithm.

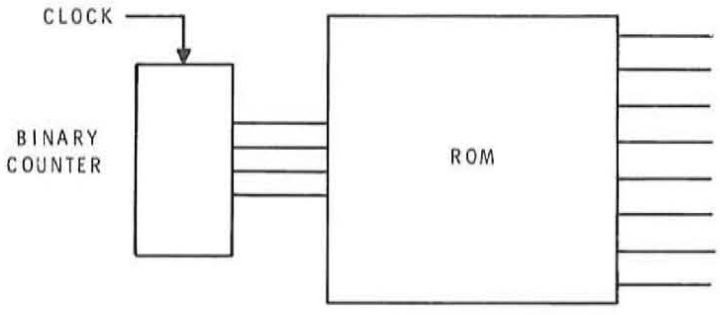
Microprogramming

Microprogramming is a technique originally developed to systematize the automatic control logic in a digital computer. The heart of a microprogrammed control unit is a read only memory. The read only memory is combined with other logic elements to perform sequential logic operations. This combination is called a microprogrammed controller.

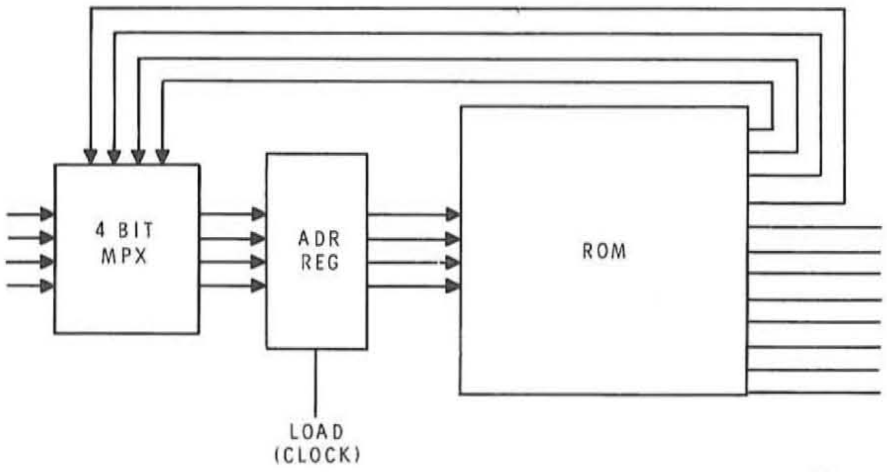
Most sequential operations are carried out by counters and shift registers in combination with combinational logic circuits. These circuits are used to generate a sequence of timing pulses that will control the operations in other parts of the digital system. The signals generated may increment counters, cause data transfers to take place between registers, enable or inhibit various logic gates, select a multiplexer channel or permit a decoding operation to take place. All of these operations will be timed so that they occur in the correct sequence to perform the desired operation.

In large digital systems the control logic circuitry can become very complex. This is particularly true of the control logic in a digital computer. By the use of microprogrammed controller, the entire network of sequential and combinational logic circuits can be replaced by a very simple circuit containing a read only memory. Figure 8-84 shows several methods of implementing sequential logic functions with a ROM.

In Figure 8-84A, the ROM is driven by a binary counter. A periodic clock signal increments the four bit binary counter. The counter output is used as the ROM address input. The address decoder is assumed to be part of the ROM itself. The ROM output consists of parallel 8 bit words. Since there are four input bits, the ROM therefore contains sixteen 8 bit words. As the binary counter is incremented, a sequence of 8 bit words appears at the ROM output. The words stored in the ROM are programmed such that the binary states appearing at the ROM outputs will cause the desired logic operations to take place in the correct sequence. Here the eight output lines can be used for a variety of control purposes. The states of these outputs are strictly a function of the bits stored in the ROM. The rate of change of the ROM outputs is a function of the frequency of the clock pulses stepping the binary counter.



(A)



(B)

Figure 8-84. Microprogrammed controllers using a ROM.

A more sophisticated version of this same circuit is shown in Figure 8-84B. Again, a ROM is the main circuit element with the desired output states specified by the ROM contents. Note, however, that four of the ROM output bits are fed back around to the inputs of a 4-bit multiplexer. Another group of four input bits is applied to the multiplexer as well. The multiplexer can select either of the two 4-bit sources as an address and feed them to an address register. The address register in turn selects a specific ROM word. In this circuit, the four bits define sixteen words in the ROM. The output is a 12-bit word, eight bits for controlling external operations and four bits which are used to determine the next address of the word in the ROM.

To operate this circuit, a four bit starting address is applied to the multiplexer from an external source. This is applied to the address register. One specific word in the ROM is addressed and its outputs appear. At this point the state of the multiplexer is changed so that the next input to the address register will come from the four bits in the current ROM output word. This permits the ROM to select the next word that should appear at the output. By repeatedly loading the address register with a clock signal, the ROM addresses are sequenced and a desired pattern of output pulses is produced. With this arrangement, the ROM words addressed can be either sequential or in any desired order. The four bit address output from the ROM could specify the next location in sequence. Alternately, it can select any other word in memory. All of this is determined beforehand in the design of the circuitry. Once the proper sequence of operations is determined, the contents of the ROM can be specified.

The term micro programming is applicable to these circuits in the sense that the words stored in the ROM make up a specific program for carrying out a specific function. Each word in the ROM is referred to as a microinstruction. The bits of that word appearing at the ROM outputs cause certain operations to occur; in other words, that word instructs the external circuitry in the function to be performed. Each word or microinstruction stored in the memory makes up a microprogram. The microprogram defines the complete operation to be initiated. The circuits given in Figure 8-84 are only a few of many different ways that microprogrammed operations can be implemented.

Self Test Review

32. The term RAM generally refers to a:
    a. read/write memory.
    b. read only memory.
    c. either of the above.

33. The binary input word to a ROM (or RAM) is often referred to as a (n) ____________________.

34. A 1024 bit ROM is organized to store 4 bit words. This ROM contains ____________________ words and requires a(n) ____________________ bit input address.

35. Data is written into a ROM when it is:
    a. operating
    b. addressed
    c. manufactured
    d. Data is never written into a ROM.

36. Refer to Figure 8-77. Write the binary contents of each memory location in the spaces provided.

Address	Binary Data
0	
1	
2	
3	
4	
5	
6	
7	

37. The total bit capacity of the ROM in Figure 8-77 is __________.

38. What is the word size of a 16 x 4 ROM?

a. 4 bits
b. 8 bits
c. 16 bits
d. 64 bits

39. The propagation delay in a ROM is called ____________________ ____________________.

40. On a given size silicon chip which type of device will produce the largest memory?

a. bipolar
b. MOS

41. A ROM can be used as a BCD to seven segment decoder. Such a ROM will have (how many?) ____________________ inputs, ____________________ outputs and a total bit capacity of ____________________. The word organization is ________________ x ________________.

42. A ROM could be used to perform a square root operation.

a. True
b. False

43. The main logic element in a microprogrammed controller is a ____________________.

44. Another name for the words stored in a microprogrammed ROM is ____________________.

45. Microprogrammed controllers are
    a. combinational circuits.
    b. sequential circuits.

Answers

32. a. Read/write memory

33. address

34. 256, 8. A 1024 bit memory organized into 4 bit words contains 1024 ÷ 4 = 256 words. It takes an 8 bit address to locate any one of the words (2^8 = 256).

35. c. manufactured. When a programmable ROM is used, the user can store data into it once as the storage is usually permanent.

36.

Address	Binary Data
0	0010
1	1001
2	1111
3	0101
4	0100
5	1010
6	0011
7	0000

37. 32 There are 8 four bit words. (8 x 4 = 32 bits)

38. a. 4 bits. In the designation 16 × 4, the second number, usually the smaller of the two, refers to the word size. The first number refers to the number of words in the memory.

39. access time

40. b. MOS

41. 4 inputs, 7 outputs, 70 bits, 10 x 7 organization. A BCD to 7 segment decoder will have 4 inputs (the BCD input code) and 7 outputs (one for each segment). The BCD code will address 10 memory locations, one for each of the digits 0 through 9. The word in each location will have 7 bits. Therefore, the total bit capacity is 10 x 7 = 70. The organization of course is 10 x 7.

42. a. True   A ROM can be used for square root operations. The number whose square root is to be found is applied as a binary address to the ROM. In the corresponding memory location will be the binary number representing the square root of the input.

43. ROM

44. microinstructions

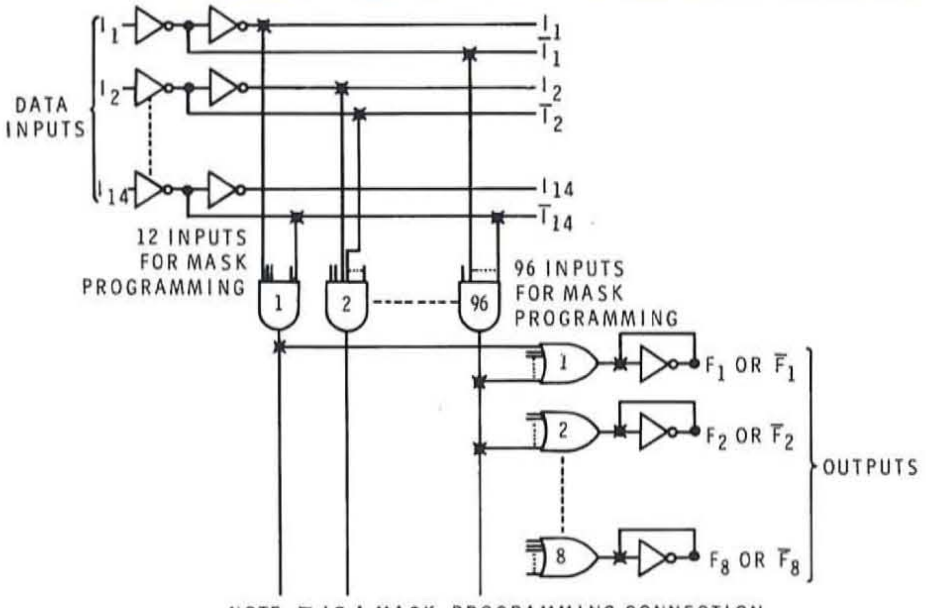
45. b. sequential

PROGRAMMABLE LOGIC ARRAYS

A programmable logic array (PLA) is an integrated circuit logic network that can be used to perform many different types of combinational logic functions. It offers the digital designer an alternative to the use of combinational logic circuits made with standard SSI and MSI ICs or read only memories. For many applications the PLA offers a significant improvement in performance over both conventional logic circuit implementation and read only memories.

Basically the PLA is a bipolar or MOS logic network that can be programmed during manufacturing to produce a wide variety of combinational logic functions. It is capable of translating any input code into any output code. The circuit is designed to generate a large number of sums of partial products.

Figure 8-85 shows a general logic diagram of a PLA. The multiple binary inputs (I_1 through I_{14}) are applied to inverters which are used to generate both the normal and complement versions of the input signals. The inverter outputs may then be interconnected to one of many AND gates. In one particular commercial PLA, a total of 96 twelve input AND gates are provided. These AND gates generate the product terms of the input variables. Up to 14 indifferent input variables may be handled by this particular PLA. The products or partial products of the inputs formed by the AND



DATA INPUTS
I_1
I_2
I_{14}
I_1
$\overline{I}_1$
I_2
$\overline{I}_2$
I_{14}
$\overline{I}_{14}$
12 INPUTS FOR MASK PROGRAMMING
1
2
96
96 INPUTS FOR MASK PROGRAMMING
1
2
8
F_1 OR $\overline{F}_1$
F_2 OR $\overline{F}_2$
F_8 OR $\overline{F}_8$
OUTPUTS

NOTE: ✱ IS A MASK-PROGRAMMING CONNECTION

Figure 8-85. General logic diagram of a programmable logic array.

gates are then connected to OR gates to form the output sums (F1 through F8). The selection of which input variables are applied to which AND gates and the choice of which AND gate outputs are connected to the eight available output OR gates is determined during the manufacturing of the device. By properly designing the mask that specifies the interconnections of the logic gates on the chip, a huge number of circuit configurations are possible. Note also in Figure 8-85 that even the use of an inverter on the outputs of the OR gates is programmable.

To design a logic circuit with a PLA involves basically the same procedure as developing the logic for any combinational logic circuit or selecting an ROM. The procedure generally starts with the truth table that defines the output states for each combination of input states. The output equations are written in sum-of-products form. The required product terms are listed and these are converted into the appropriate mask programming instructions for making the IC.

The PLA is particularly valuable in implementing large complex combinational logic circuits. Simple functions are readily implemented with SSI logic gates. More complex functions can be handled by one of the many available MSI functions. But when there are many input variables and many output variables, the use of standard SSI and MSI packages also becomes complex and cumbersome. The PLA can be used to generate the desired complex function and house it in a single integrated circuit package.

The PLA also offers numerous advantages over the read only memory for implementing some complex logic functions. Of course, a ROM can be used to handle logic functions involving any number of inputs or any number of outputs. However, the read only memory becomes very inefficient when all possible combinations of the input variables are not used. For example, a four input logic circuit has 16 possible different combinations. The design application may only call for the use of nine of these. The four bits on input to a ROM to be used in implementing the function define 16 memory locations, seven of which would not be used. Despite the fact the seven locations would not be used they are still present in the device and are essentially wasted. However, by using a PLA, the same logic function can be implemented more economically. PLAs offer the digital designer another option in implementing combinational logic circuits. For large, complex logic functions involving four or more inputs and outputs, it offers advantages over SSI and MSI combinational circuits and ROMs for some applications.

Self Test Review

46. The logic function performed by a PLA is determined during manufacturing.

    a. True
    b. False

47. A PLA could be used to perform code conversion.

    a. True
    b. False

48. A PLA is an alternative to what other types of logic circuits? Check all that apply.

    a. Sequential
    b. MSI functional combinational
    c. SSI combinations
    d. ROMs

49. The logic output equation of a PLA is in the ____________ of ____________ form.

50. PLAs are used primarily in implementing small simple logic functions.

    a. True
    b. False

Answers

46. a. True

47. a. True

48. b, c, d. A PLA is a combinational circuit that can replace SSI, MSI combinational circuits and ROMs in large complex applications.

49. sum-of-products

50. b. False

EXAMINATION

UNIT 8

COMBINATIONAL LOGIC CIRCUITS

The purpose of this exam is to help you review the key facts in this unit. The problems are designed to test your retention and understanding by making you apply what you have learned. This exam is not so much a test as it is another learning method. Be fair to yourself and work every problem first before checking the answers.

1. Draw a logic gate decoder that will recognize the number 56 in binary form.

2. Draw a logic gate decoder that will detect the presence of the number 56 in BCD form.

3. A 1 of 16 decoder has how many inputs?

   A. 1
   B. 4
   C. 8
   D. 16

4. The Boolean equation for a decoder that recognizes the number 11_{10} is: (Note: D = LSB)

   A. $\overline{A}\,B\,\overline{C}\,\overline{D}$
   B. $\overline{A}\,\overline{B}\,C\,D$
   C. $\overline{A}\,B\,\overline{C}\,D$
   D. $A\,\overline{B}\,C\,D$

5. Which of the following statements is true about the 7442 TTL decoder IC? Check all that apply.

   A. Active low outputs
   B. Active high outputs
   C. One of ten decoder
   D. All outputs low except enabled output
   E. Can be used as an octal decoder
   F. All outputs high if numbers between 1010 and 1111 are applied.
   G. Decode 8421 BCD code

6. What type of combinational circuit would you use to generate a binary output code from an input pushbutton?
   A. decoder
   B. multiplexer
   C. encoder
   D. exclusive OR

7. What logic circuit is analogous to a single pole mechanical selector switch?
   A. decoder
   B. multiplexer
   C. encoder
   D. exclusive OR

8. Which of the following is *not* a typical application for a multiplexer?
   A. serial to parallel converter
   B. Boolean function generator
   C. serial word generator
   D. data selector

9. In the 74151 TTL multiplexer the desired input channel is selected by
   A. the strobe input
   B. disabling all other inputs
   C. applying a signal to that channel
   D. a 3 bit address input code

10. Which *two* circuits below can be used as serial-to-parallel data converters?
    A. demultiplexer
    B. encoder
    C. multiplexer
    D. exclusive OR
    E. shift register

11. Another name for exclusive NOR is
    A. exclusive OR
    B. comparator
    C. wired-OR
    D. sum-of-products

12. Write the logic equation for the circuit in Figure 8-86. This circuit is a(n)

    A. exclusive OR
    B. exclusive NOR
    C. decoder
    D. multiplexer
    E. comparator

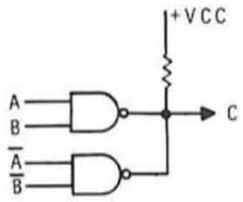


Figure 8-86.
Circuit for exam question 12.

13. Study the circuit in Figure 8-87. Assume clock pulses are applied to the A flip-flop. Consider the flip-flop output code sequences generated when the control input is first binary 0 then binary 1. What is this circuit?

    A. shift register
    B. BCD counter
    C. up/down counter
    D. code converter

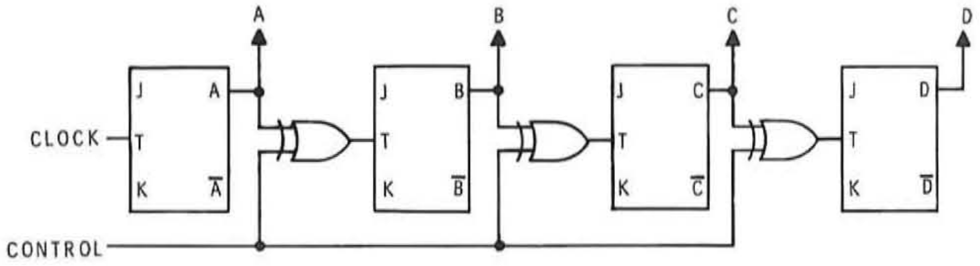


Figure 8-87.
Circuit for exam question 13.

14. Draw an odd parity generator circuit for a 7 bit binary input code.

15. In a serial binary adder,

    A. all bits are added simultaneously
    B. the bits are added sequentially, LSB first
    C. the bits are added sequentially, MSB first

16. Add the following binary numbers.

    A. 1000111 B. 1110001010
    1011110 1010101110
    _______ __________

17. A serial binary adder uses 8 bit shift registers to store the numbers to be added (augend and addend). The clock frequency is 250 KHz. How long does it take to make an addition?

    A. 4 microseconds
    B. 32 microseconds
    C. 64 microseconds
    D. 500 microseconds

18. Which of the following is *not* a suitable application for a ROM?
    A. binary to ASCII code converter.
    B. modulo 13 counter.
    C. cube root arithmetic.
    D. rectangular to polar coordinate converter.

19. A ROM has 10 input address bits and an 8 bit output word. How many bits does the ROM contain?
    A. 1024
    B. 2048
    C. 4096
    D. 8192

20. The term given to a sequential logic circuit using a ROM is
    A. counter
    B. microprogrammed controller
    C. code converter
    D. random logic sequencer

21. Which of the following is *not* a characteristic of a PLA?
    A. multiple inputs
    B. multiple outputs
    C. sequential logic circuit
    D. sum-of-products generator

ANSWERS
UNIT 8
COMBINATIONAL LOGIC CIRCUITS

1. See Figure 8-88.

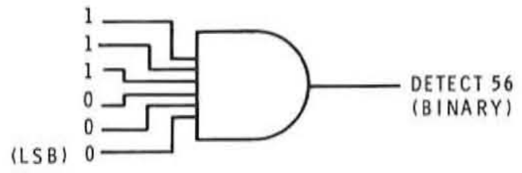


Figure 8-88.
Answer to exam question 1.

2. See Figure 8-89.

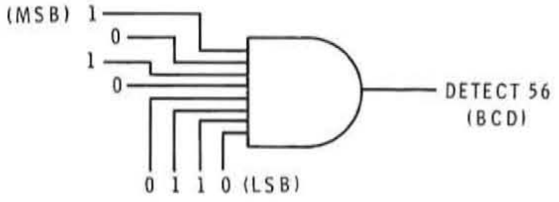


Figure 8-89.
Answer to exam question 2.

3. B. Four. 16 output states implies a 4-bit input. ($2^4 = 16$.)

4. D. $A \overline{B} C D$. $A \overline{B} C D = 1011_2 = 11_{10}$

5. A, C, E, F, G

6. C. encoder

7. B. multiplexer

8. A. serial-to-parallel converter. A multiplexer can be used as a parallel-to-serial converter.

9. D. a 3-bit address input code.

10. A, E demultiplexer, shift register

11. B. comparator

12. A. exclusive OR. The wired OR NAND gates in Figure 8-86 generate the equation $C = \overline{A B + \overline{A}\,\overline{B}} = \overline{A} B + A \overline{B}$

13. C. up/down counter. The circuit in Figure 8-87 is a 4 bit binary ripple up/down counter. The JK flip-flops are coupled by X-OR circuits that control the up/down mode. When the control input is binary 0, the normal flip-flop outputs are coupled to the T inputs of the next flip-flop in sequence thereby creating an up counter. When the control input is binary 1, the X-OR gates invert the normal output giving the effect of driving the T inputs from the complement output of the previous flip-flop which produces down counting.

14. See Figure 8-90.

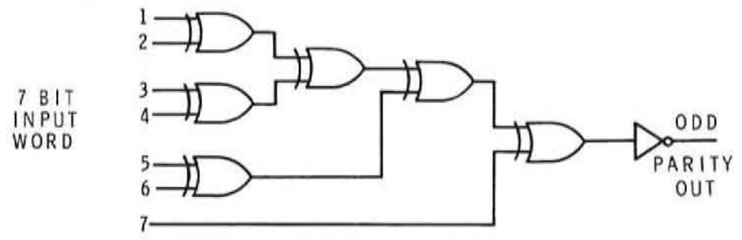


Figure 8-90. 7 bit odd parity generator. Answer to exam question 14.

15. B. The bits are added sequentially, LSB first.

```
                   carries
16. A.    111                   B. 11     11
         1000111                  1110001010
         1011110                  1010101110
         -------                  ----------
        10100101                 11000111000
```

17. B. 32 microseconds. In a serial adder, the addition is complete when both words in the 8 bit registers are shifted out. This takes 8 shift or clock pulses. With a 250 KHz clock with a period of 1/250000 = .000004 second or 4 microseconds, the total add time is 4 x 8 = 32 microseconds.

18. B. modulo 13 counter. A ROM is not a sequential circuit.

19. D. 8192. A ten bit address word defines 2^{10} = 1024 words at 8 bits each for a total of 8192 bits.

20. B. microprogrammed controller

21. C. sequential logic circuit. A PLA is a combinational logic circuit.