CHAPTER 3

SPECIAL LOGIC CIRCUITS

LEARNING OBJECTIVES

Upon completion of this chapter, you should be able to do the following:

1. Recognize the types of special logic circuits used in digital equipment.
2. Identify exclusive OR and exclusive NOR circuits and interpret their respective Truth Tables.
3. Identify adder and subtracter circuits.
4. Identify the types of flip-flops used in digital equipment and their uses.
5. Identify counters, registers, and clock circuits.
6. Describe the elements that make up logic families — RTL, DTL, TTL, CMOS.

INTRODUCTION

Figure 3-1 is a portion of a typical logic diagram. It is similar to the diagrams you will encounter as your study of digital circuitry progresses.

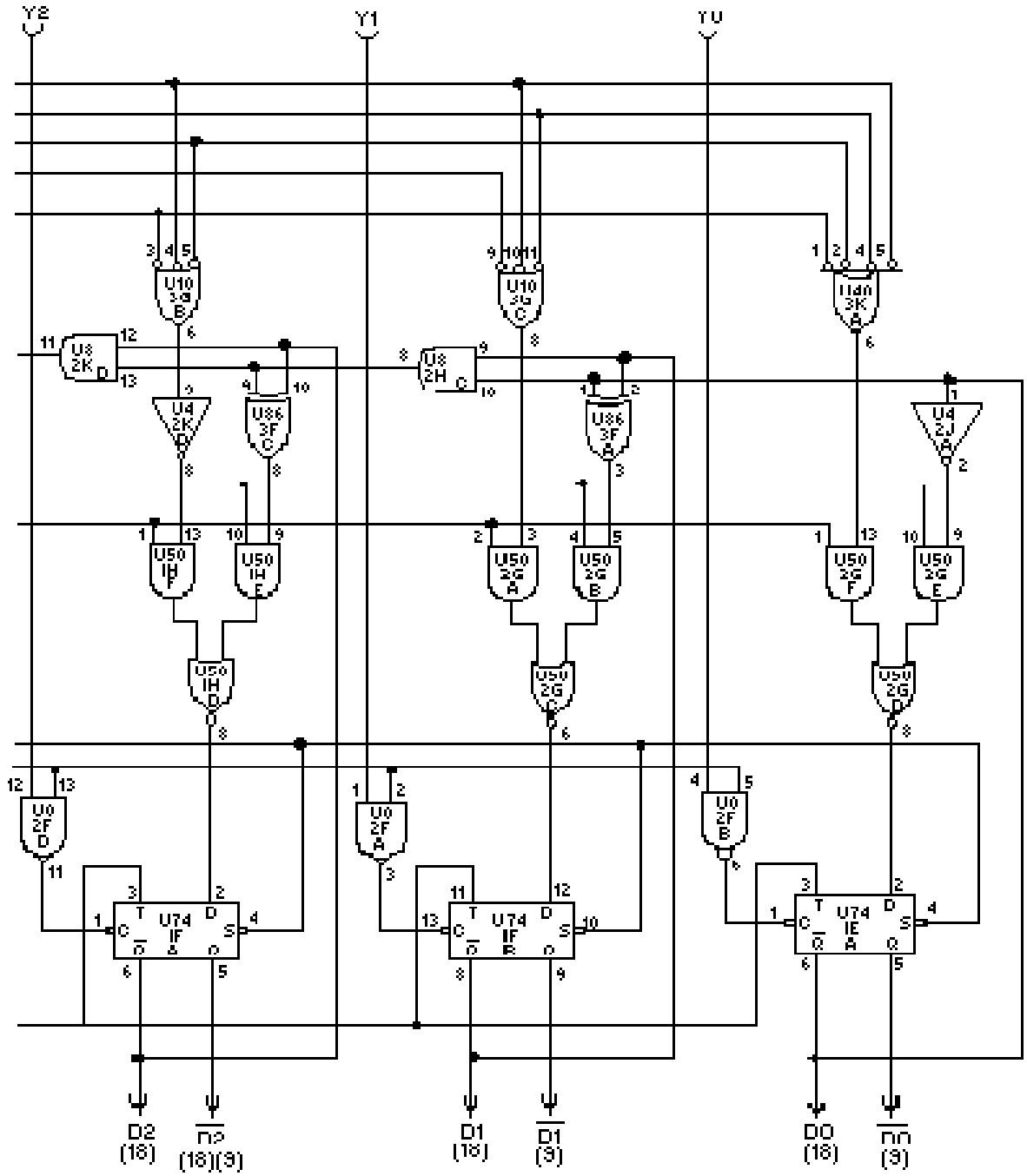


Figure 3-1. —Typical logic diagram.

Look closely at the figure. You will see many familiar logic gates. You will also see several that you may not recognize.

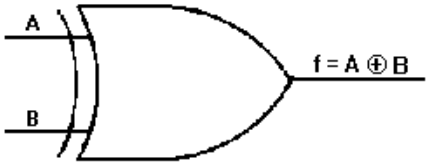
Digital equipment must be capable of many more operations than those described in chapter 2. Provisions must be made for accepting information; performing arithmetic or logic operations; and transferring, storing, and outputting information. Timing circuits are included to ensure that all operations occur at the proper time.

In this chapter you will become acquainted with the logic circuits used to perform the operations mentioned above.

THE EXCLUSIVE OR GATE

The exclusive OR gate is a modified OR gate that produces a HIGH output when only one of the inputs is HIGH. You will often see the abbreviation X-OR used to identify this gate. When both inputs are HIGH or when both inputs are LOW, the output is LOW.

The standard symbol for an exclusive OR gate is shown in figure 3-2 along with the associated Truth Table. The operation function sign for the exclusive OR gate is ⊕.



A	B	f
0	0	0
0	1	1
1	0	1
1	1	0

Figure 3-2. —Exclusive OR gate and Truth Table.

If you were to observe the input and output signals of an X-OR gate, the results would be similar to those shown in figure 3-3. At T_0, both inputs are LOW and the output is LOW. At T_1, A goes to HIGH and remains HIGH until T_2. During this time the output is HIGH. At T_3, B goes HIGH and remains HIGH through T_5. At T_4, A again goes HIGH and remains HIGH through T_5. Between $T_3$and T_4, the output is HIGH. At T_4, when both A and B are HIGH, the output goes LOW.

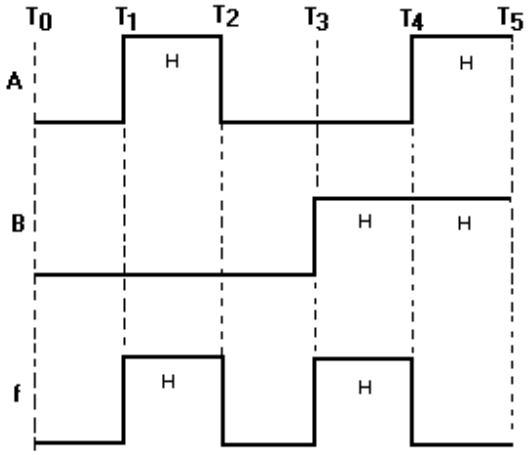
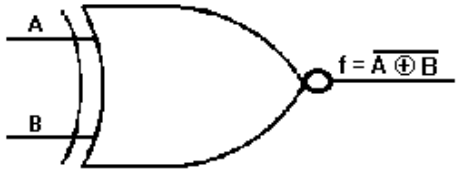


Figure 3-3. —Exclusive OR gate timing diagram.

THE EXCLUSIVE NOR GATE

The exclusive NOR (X-NOR) gate is nothing more than an X-OR gate with an inverted output. It produces a HIGH output when the inputs are either all HIGH or all LOW. The standard symbol and the Truth Table are shown in figure 3-4. The operation function sign is ⊕ with a vinculum over the entire expression.



A	B	f
0	0	1
0	1	0
1	0	0
1	1	1

Figure 3-4. —Exclusive NOR gate and Truth Table.

A timing diagram for the X-NOR gate is shown in figure 3-5. You can see that from T_0 to T_1, when both inputs are LOW, the output is HIGH. The output goes LOW when the inputs are opposite; one HIGH and the other LOW. At time T_3, both inputs go HIGH causing the output to go HIGH.

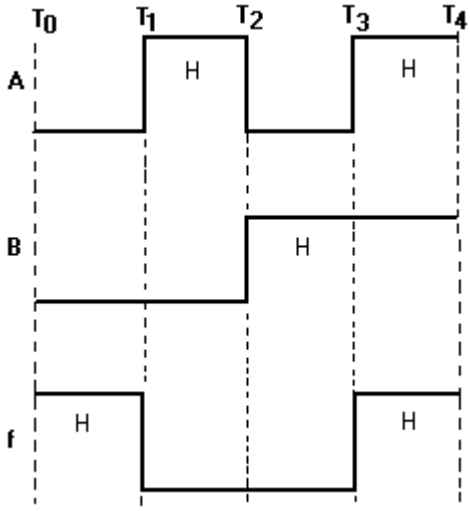


Figure 3-5. —Exclusive NOR gate timing diagram.

Q1. What is the sign of operation for the X-OR gate?

Q2. What will be the output of an X-OR gate when both inputs are HIGH?

Q3. A two-input X-OR gate will produce a HIGH output when the inputs are at what logic levels?

Q4. What type of gate is represented by the output Boolean expression $\overline{T \oplus R}$ *?*

Q5. What will be the output of an X-NOR gate when both inputs are LOW?

ADDERS

Adders are combinations of logic gates that combine binary values to obtain a sum. They are classified according to their ability to accept and combine the digits. In this section we will discuss quarter adders, half adders, and full adders.

QUARTER ADDER

A quarter adder is a circuit that can add two binary digits but will not produce a carry. This circuit will produce the following results:

0 plus 0 = 0

0 plus 1 = 1

1 plus 0 = 1

1 plus 1 = 0 (no carry)

You will notice that the output produced is the same as the output for the Truth Table of an X-OR. Therefore, an X-OR gate can be used as a quarter adder.

The combination of gates in figure 3-6 will also produce the desired results. When A and B are both LOW (0), the output of each AND gate is LOW (0); therefore, the output of the OR gate is LOW (0). When A is HIGH and B is LOW, then $\overline{B}$ is HIGH and AND gate 1 produces a HIGH output, resulting in a sum of 1 at gate 3. With A LOW and B HIGH, gate 2 output is HIGH, and the sum is 1. When both A and B are HIGH, neither AND gate has an output, and the output of gate 3 is LOW (0); no carry is produced.

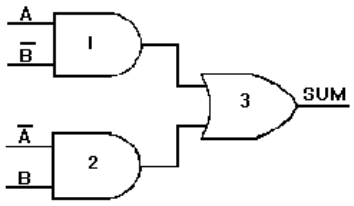
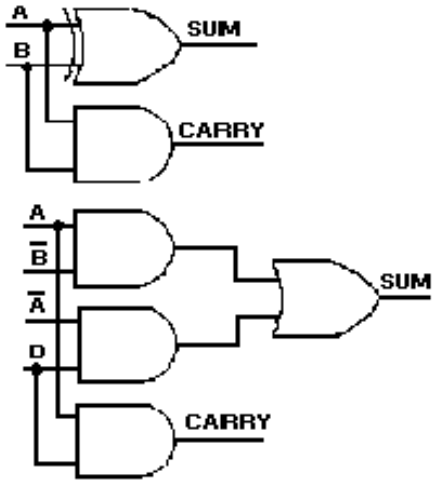


Figure 3-6. —Quarter adder.

HALF ADDER

A half adder is designed to combine two binary digits and produce a carry.

Figure 3-7 shows two ways of constructing a half adder. An AND gate is added in parallel to the quarter adder to generate the carry. The SUM column of the Truth Table represents the output of the quarter adder, and the CARRY column represents the output of the AND gate.



A	D	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

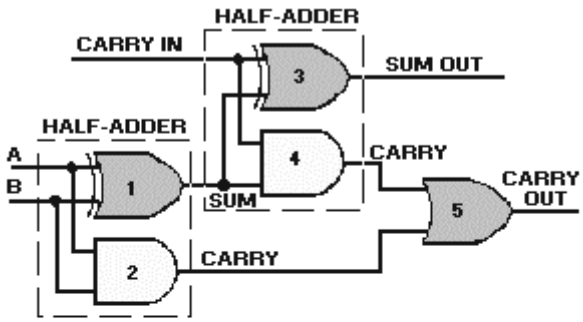
Figure 3-7. —Half adders and Truth Table.

We have seen that the output of the quarter adder is HIGH when either input, but not both, is HIGH. It is only when both inputs are HIGH that the AND gate is activated and a carry is produced. The largest sum that can be obtained from a half adder is 10_2 (1_2 plus 1_2).

FULL ADDER

The full adder becomes necessary when a carry input must be added to the two binary digits to obtain the correct sum. A half adder has no input for carries from previous circuits.

One method of constructing a full adder is to use two half adders and an OR gate as shown in figure 3-8. The inputs A and B are applied to gates 1 and 2. These make up one half adder. The sum output of this half adder and the carry-from a previous circuit become the inputs to the second half adder. The carry from each half adder is applied to gate 5 to produce the carry-out for the circuit.



A	B	CARRY IN	SUM OUT	CARRY OUT
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Figure 3-8. —Full adder and Truth Table.

Now let's add a series of numbers and see how the circuit operates.

First, let's add 1 and 0. When either A or B is HIGH, gate 1 has an output. This output is applied to gates 3 and 4. Since the carry-in is 0, only gate 3 will produce an output. The sum of 1_2 and 0 is 1_2.

Now let's add 1_2 and 1_2. If A and B are both HIGH, the output of gate 1 is LOW. When the carry-in is 0 (LOW), the output of gate 3 is LOW. Gate 2 produces an output that is applied to gate 5, which produces the carry-out. The sum of 1_2 and 1_2 is 10_2, just as it was for the half adder.

When A and B are both LOW and the carry-in is 1, only gate 3 has an output and produces a sum of 1_2 with no carry-out.

Now, let's add A or B and a carry-in. For example, let's assume that A is HIGH and B is LOW. With these conditions, gate 1 will have an output. This output and the carry-in applied to gates 3 and 4 will produce a sum out of 0 and a carry of 1. This carry from gate 4 will cause gate 5 to produce a carry-out. The sum of A and a carry (1_2 plus 1_2) is 10_2.

When A, B, and the carry-in are all HIGH, a sum of 1 and a carry-out are produced. First, consider A and B. When both are HIGH, the output of gate 1 is LOW, and the output of gate 2 is HIGH, giving us a carry-out at gate 5. The carry-in produces a 1 output at gate 3, giving us a sum of 1. The output of the full adder is 11_2. The sum of 1_2 plus 1_2 plus 1_2 is 11_2.

PARALLEL ADDERS

The adders discussed in the previous section have been limited to adding single-digit binary numbers and carries. The largest sum that can be obtained using a full adder is 11_2.

Parallel adders let us add multiple-digit numbers. If we place full adders in parallel, we can add two- or four-digit numbers or any other size desired.

Figure 3-9 uses STANDARD SYMBOLS to show a parallel adder capable of adding two, two-digit binary numbers. In previous discussions we have depicted circuits with individual logic gates shown. Standard symbols (blocks) allow us to analyze circuits with inputs and outputs only. One standard symbol may actually contain many and various types of gates and circuits. The addend would be input on the A inputs (A_2 = MSD, A_1 = LSD), and the augend input on the B inputs (B_2 = MSD, B_1 = LSD). For this explanation we will assume there is no input to C_0 (carry from a previous circuit).

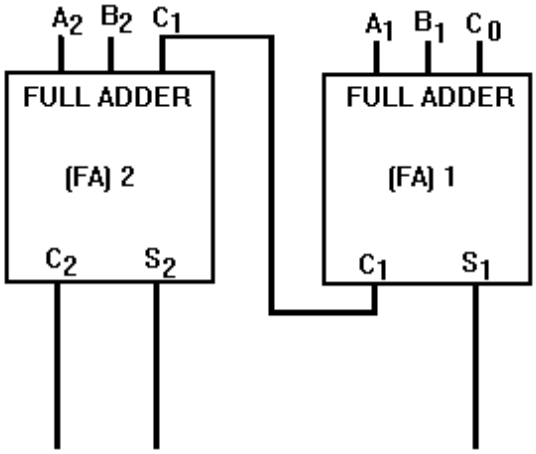


Figure 3-9. —Parallel binary adder.

Now let's add some two-digit numbers. To add 10_2 (addend) and 01_2 (augend), assume there are numbers at the appropriate inputs. The addend inputs will be 1 on A_2 and 0 on A_1. The augend inputs will be 0 on B_2 and 1 on B_1. Working from right to left, as we do in normal addition, let's calculate the outputs of each full adder.

With A_1 at 0 and B_1 at 1, the output of adder 1 will be a sum (S_1) of 1 with no carry (C_1). Since A_2 is 1 and B_2 is 0, we have a sum (S_2) of 1 with no carry (C_2) from adder 1. To determine the sum, read the outputs (C_2, S_2, and S_1) from left to right. In this case, $C_2 = 0$, $S_2 = 1$, and $S_1 = 1$. The sum, then, of 10_2 and 01_2 is 011_2 or 11_2.

To add 11_2 and 01_2, assume one number is applied to A_1 and A_2, and the other to B_1 and B_2, as shown in figure 3-10. Adder 1 produces a sum (S_1) of 0 and a carry (C_1) of 1. Adder 2 gives us a sum (S_2)

of 0 and a carry (C_2) of 1. By reading the outputs (C_2, S_2, and S_1), we see that the sum of 11_2 and 01_2 is 100_2.

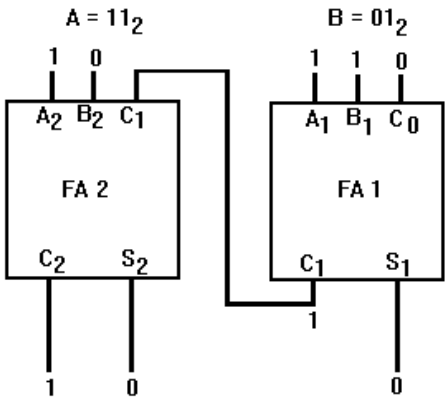


Figure 3-10. —Parallel addition.

As you know, the highest binary number with two digits is 11_2. Using the parallel adder, let's add 11_2 and 11_2.

First, apply the addend and augend to the A and B inputs. Calculate the output of each full adder beginning with full adder 1. With A_1 and B_1 at 1, S_1 is 0 and C_1 is 1. Since all three inputs (A_2, B_2, and C_1) to full adder 2 are 1, the output will be 1 at S_2 and 1 at C_2. The output of the circuit, as you read left to right, is 110_2, the sum of 11_2 and 11_2.

Parallel adders may be expanded by combining more full adders to accommodate the number of digits in the numbers to be added. There must be one full adder for each digit.

Q6. What advantage does a half adder have over a quarter adder?

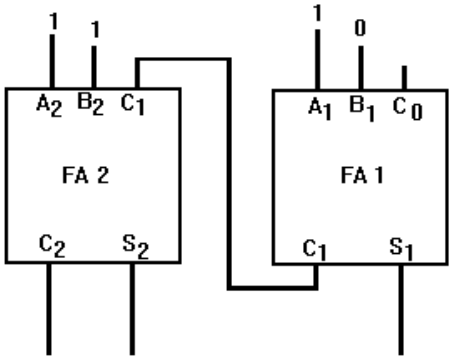
Q7. An X-OR gate may be used as what type of adder?

Q8. What will be the output of a half adder when both inputs are 1s?

Q9. What type of adder is used to handle a carry from a previous circuit?

Q10. How many full adders are required to add four-digit numbers?

Q11. With the inputs shown below, what will be the output of S_1, S_2, and C_2?



Q12. What is the output of C_1?

SUBTRACTION

Subtraction is accomplished in computers by the R's complement and add method. This is the same method you used in chapter 1 to subtract binary numbers.

R's complement subtraction allows us to use fewer circuits than would be required for separate add and subtract functions. Adding X-OR gates to full adders, as shown in figure 3-11, enables the circuit to perform R's complement subtraction as well as addition.

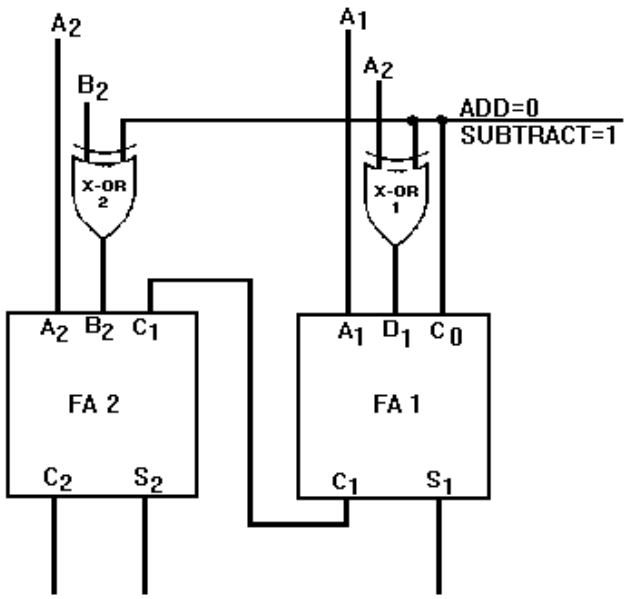


Figure 3-11. —R's complement adder/subtracter.

To add two numbers using this circuit, the addend and augend are applied to the A and B inputs. The B inputs are applied to one input of the X-OR gates. A control signal is applied to the other input of the X-OR gates. When the control signal is LOW, the circuit will add; and when it is HIGH, the circuit will subtract.

In the add mode, the outputs of the X-OR gates will be the same as the B inputs. Addition takes place in the same manner as described in parallel addition.

Before we attempt to show subtraction, let's review R's complement subtraction. To subtract 10_2 from 11_2, write down the minuend (11_2). Perform the R's complement on the subtrahend. Now add the minuend and the complemented subtrahend.

11_2 minuend

$+ \quad 10_2$ R's complement

101 Difference

Disregard the most significant 1, and the difference between 11_2 and 10_2 is 01_2. The most significant 1 will not be used in the example shown in the following paragraph.

Now let's subtract 10_2 from 11_2 using the adder/subtracter circuit. The minuend (11_2) is input on the A terminals, and the subtrahend (10_2) is input on the B terminals. In the subtract mode, a 1 from the control circuit is input to each of the X-OR gates and to the C_0 carry input. By applying a 1 to each of the X-OR gates, you find the output will be the complement of the subtrahend input at B_1 and B_2. Since B_1 is a 0, the output of X-OR 1 will be 1. The input B_2 to X-OR 2 will be inverted to a 0. The HIGH input to C_0 acts as a carry from a previous circuit. The combination of the X-OR gates and the HIGH at C_0 produces the R's complement of the subtrahend. The full adders add the minuend and the R's complement of the subtrahend and produce the difference. The output of C_2 is not used. The outputs of S_2 and S_1 are 0 and 1, respectively, indicating a difference of 01_2. Therefore, 11_2 minus 10_2 equals 01_2.

Q13. What type of logic gates are added to a parallel adder to enable it to subtract?

Q14. How many of these gates would be needed to add a four-digit number?

Q15. In the add mode, what does the output of C_2 indicate?

Q16. In the subtract mode, a 1 at C_0 performs what portion of the R's complement?

Q17. In the subtract mode, which portion of the problem is complemented?

FLIP-FLOPS

Flip-flops (FFs) are devices used in the digital field for a variety of purposes. When properly connected, flip-flops may be used to store data temporarily, to multiply or divide, to count operations, or to receive and transfer information.

Flip-flops are bistable multivibrators. The types used in digital equipment are identified by the inputs. They may have from two up to five inputs depending on the type. They are all common in one respect. They have two, and only two, distinct output states. The outputs are normally labeled Q and $\overline{Q}$ and should always be complementary. When Q = 1, then $\overline{Q}$ = 0 and vice versa.

In this section we will discuss four types of FFs that are common to digital equipment. They are the R-S, D, T, and J-K FFs.

R-S FLIP-FLOP

The R-S FF is used to temporarily hold or store information until it is needed. A single R-S FF will store one binary digit, either a 1 or a 0. Storing a four-digit binary number would require four R-S FFs.

The standard symbol for the R-S FF is shown in figure 3-12, view A. The name is derived from the inputs, R for reset and S for set. It is often referred to as an R-S LATCH. The outputs Q and $\overline{Q}$ are complements, as mentioned earlier.

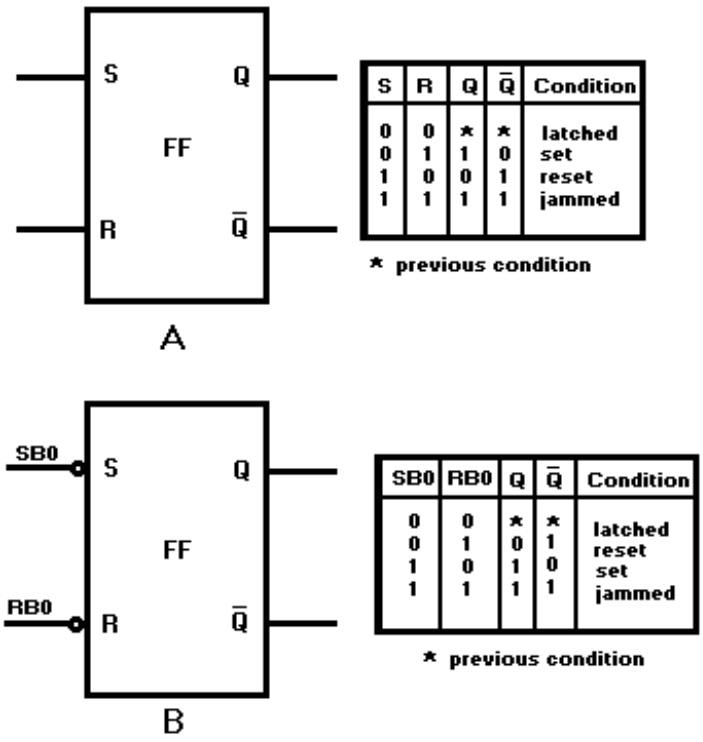


Figure 3-12. —R-S flip-flop: A. Standard symbol; B. R-S FF with inverted inputs.

The R-S FF has two output conditions. When the Q output is HIGH and $\overline{Q}$ is LOW, the FF is set. When Q is LOW and $\overline{Q}$ is HIGH, the FF is reset. When the R and S inputs are both LOW, the Q and $\overline{Q}$ outputs will both be HIGH. When this condition exists, the FF is considered to be JAMMED and the outputs cannot be used. The jammed condition is corrected when either S or R goes HIGH.

To set the flip-flop requires a HIGH on the S input and a LOW on the R input. To reset, the opposite is required; S input LOW and R input HIGH. When both R and S are HIGH, the FF will hold or "latch" the condition that existed before both inputs went HIGH.

Because the S input of this FF requires a logic LOW to set, a more easily understood symbol is shown in figure 3-12, view B. Refer to this view while reading the following paragraph.

In our description of R-S FF operation, let's assume that the signals applied to the S and R inputs are the LSDs of two different binary numbers. Let's also assume that these two binary numbers represent the speed and range of a target ship. The LSDs will be called SB0 (Speed Bit 0) and RB0 (Range Bit 0) and will be applied to the S and R inputs respectively. Refer to figure 3-12, view B, and figure 3-13. At time T_0, both SB0 and RB0 are HIGH, as a result, both Q and $\overline{Q}$ are HIGH. This is the jammed state and as mentioned earlier, cannot be used in logic circuitry. At T_1, SB0 goes LOW and RB0 remains HIGH; Q goes LOW and $\overline{Q}$ remains HIGH; the FF is reset. At T_2 RB0 goes LOW and SB0 remains LOW; the FF is latched in the reset condition. At T_3, SB0 goes HIGH and RB0 remains LOW; the FF sets. At T_4 SB0 goes LOW and RB0 goes HIGH; the FF resets. When SB0 and RB0 input conditions reverse at T_5, the FF sets. The circuit is put in the latch condition at T_6 when SB0 goes LOW. Notice that the output changes states ONLY when the inputs are in opposite states.

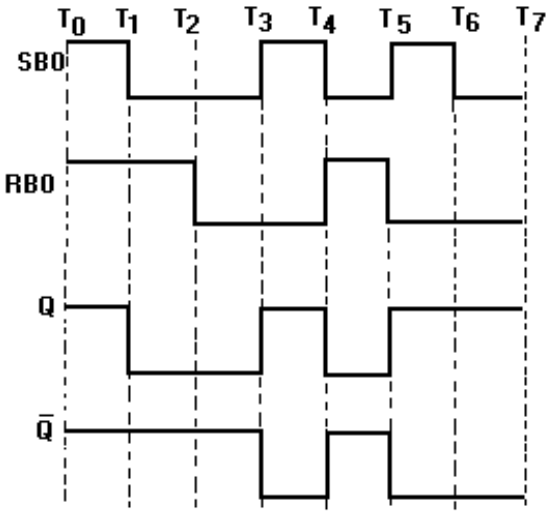


Figure 3-13. —R-S flip-flop with inverted inputs timing diagram.

Figure 3-14 shows two methods of constructing an R-S FF. We can use these diagrams to prove the Truth Table for the R-S FF.

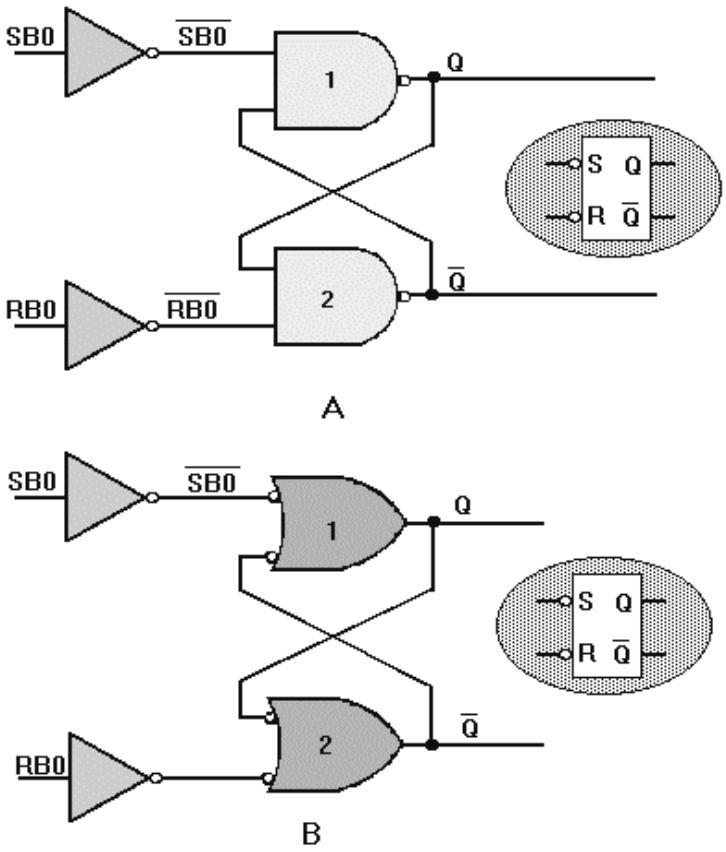


Figure 3-14. —R-S FF construction: A. Using cross-coupled NAND gates; B. Using cross-coupled OR gates.

Look at figure 3-14, view A. Let's assume SB0 is HIGH and RB0 is LOW. You should remember from chapter 2 that the output of an inverter is the complement of the input. In this case, since SB0 is HIGH, $\overline{SBO}$ will be LOW. The LOW input to NAND gate 1 causes the Q output to go HIGH. This HIGH Q output is also fed to the input of NAND gate 2. The other input to NAND gate 2, $\overline{RBO}$, is HIGH. With both inputs to gate 2 HIGH, the output goes LOW. The LOW $\overline{Q}$ output is also fed to NAND gate 1 to be used as the "latch" signal. If SB0 goes LOW while this condition exists, there will be no change to the outputs because the FF would be in the latched condition; both SB0 and RB0 LOW.

When RB0 is HIGH and SB0 is LOW, $\overline{RBO}$ being LOW drives the output, $\overline{Q}$, to a HIGH condition. The HIGH $\overline{Q}$ and HIGH $\overline{SBO}$ inputs to gate 1 cause the output, Q, to go LOW. This LOW is also fed to NAND gate 2 to be used as the latch signal. Since SB0 is LOW, the FF will again go into the latched mode if RB0 goes LOW.

The cross-coupled OR gates in figure 3-14, view B, perform the same functions as the NAND gate configuration of view A. A HIGH input at SB0 produces a HIGH Q output, and a LOW at RB0 produces a LOW $\overline{Q}$ output. The cross-coupled signals ($\overline{Q}$ to gate 1 and Q to gate 2) are used as the latch signals just as in view A. You can trace other changes of the inputs using your knowledge of basic logic gates.

Q18. What are R-S FFs used for?

Q19. How many R-S FFs are required to store the number 100101_2?

Q20. For an R-S FF to change output conditions, the inputs must be in what states?

Q21. How may R-S FFs be constructed?

TOGGLE FLIP-FLOP

The toggle, or T, flip-flop is a bistable device that changes state on command from a common input terminal.

The standard symbol for a T FF is illustrated in figure 3-15, view A. The T input may be preceded by an inverter. An inverter indicates a FF will toggle on a HIGH-to-LOW transition of the input pulse. The absence of an inverter indicates the FF will toggle on a LOW-to-HIGH transition of the pulse.

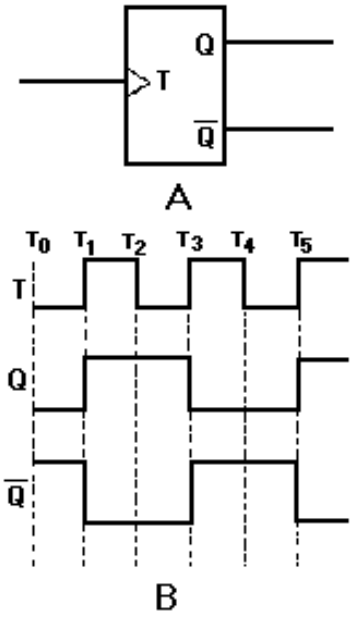


Figure 3-15. —Toggle (T) flip-flop: A. Standard symbol; B. Timing diagram.

The timing diagram in figure 3-15, view B, shows the toggle input and the resulting outputs. We will assume an initial condition (T_0) of Q being LOW and $\overline{Q}$ being HIGH. At T_1, the toggle changes from a LOW to a HIGH and the device changes state; Q goes HIGH and $\overline{Q}$ goes LOW. The outputs remain the same at T_2 since the device is switched only by a LOW-to-HIGH transition. At T_3, when the toggle goes HIGH, Q goes LOW and $\overline{Q}$ goes HIGH; they remain that way until T_5.

Between T_1 and T5, two complete cycles of T occur. During the same time period, only one cycle is observed for Q or $\overline{Q}$. Since the output cycle is one-half the input cycle, this device can be used to divide the input by 2.

The most commonly used T FFs are J-K FFs wired to perform a toggle function. This use will be demonstrated later in this section.

Q22. How many inputs does a T FF have?

Q23. What is the purpose of using T FFs?

D FLIP-FLOP

The D FF is a two-input FF. The inputs are the data (D) input and a clock (CLK) input. The clock is a timing pulse generated by the equipment to control operations. The D FF is used to store data at a predetermined time and hold it until it is needed. This circuit is sometimes called a delay FF. In other words, the data input is delayed up to one clock pulse before it is seen in the output.

The simplest form of a D FF is shown in figure 3-16, view A. Now, follow the explanation of the circuit using the Truth Table and the timing diagram shown in figure 3-16, views B and C.

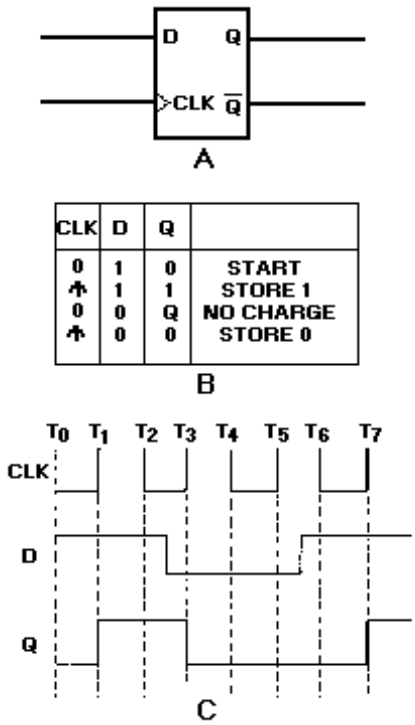


Figure 3-16. —D flip-flop: A. Standard symbol; B. Truth Table; C. Timing diagram.

Depending on the circuit design, the clock (CLK) can be a square wave, a constant frequency, or asymmetrical pulses. In this example the clock (CLK) input will be a constant input at a given frequency. This frequency is determined by the control unit of the equipment. The data (D) input will be present when there is a need to store information. Notice in the Truth Table that output Q reflects the D input only when the clock transitions from 0 to 1 (LOW to HIGH).

Let's assume that at T_0, CLK is 0, D is 1, and Q is 0. Input D remains at 1 for approximately 2 1/2 clock pulses. At T_1, when the clock goes to 1, Q also goes to 1 and remains at 1 even though D goes to 0 between T_2 and T_3. At T_3, the positive-going pulse of the clock causes Q to go to 0, reflecting the condition of D. The positive-going clock pulse at T_5 causes no change in the output because D is still LOW. Between T_5 and T_6, D goes HIGH, but Q remains LOW until T_7 when the clock goes HIGH.

The key to understanding the output of the D FF is to remember that the data (D) input is seen in the output only after the clock has gone HIGH.

You may see D FF symbols with two additional inputs — CLR (clear) and PR (preset). These inputs are used to set the start condition of the FF — CLR sets Q to 0; PR sets Q to 1. Figure 3-17 shows the standard symbol with the CLR and PR inputs. Since these inputs are preceded by inverters (part of the FF), a LOW-going signal is necessary to activate the FF. These signals (CLR and PR) override any existing condition of the output.

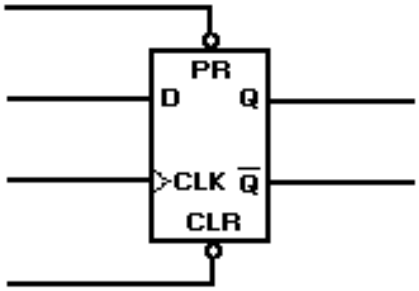


Figure 3-17. —D flip-flop with PR and CLR inputs.

You may also see an inverter at the clock input. In this case, the output will change on the negative-going transition of the clock pulse.

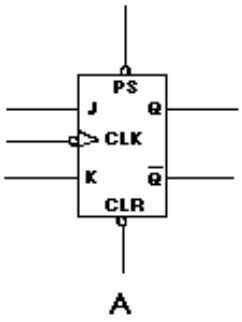
Q24. What are the inputs to a D FF?

Q25. How long is data delayed by a D FF?

Q26. What condition must occur to have a change in the output of a D FF?

J-K FLIP-FLOP

The J-K FF is the most widely used FF because of its versatility. When properly used it may perform the function of an R-S, T, or D FF. The standard symbol for the J-K FF is shown in view A of figure 3-18.

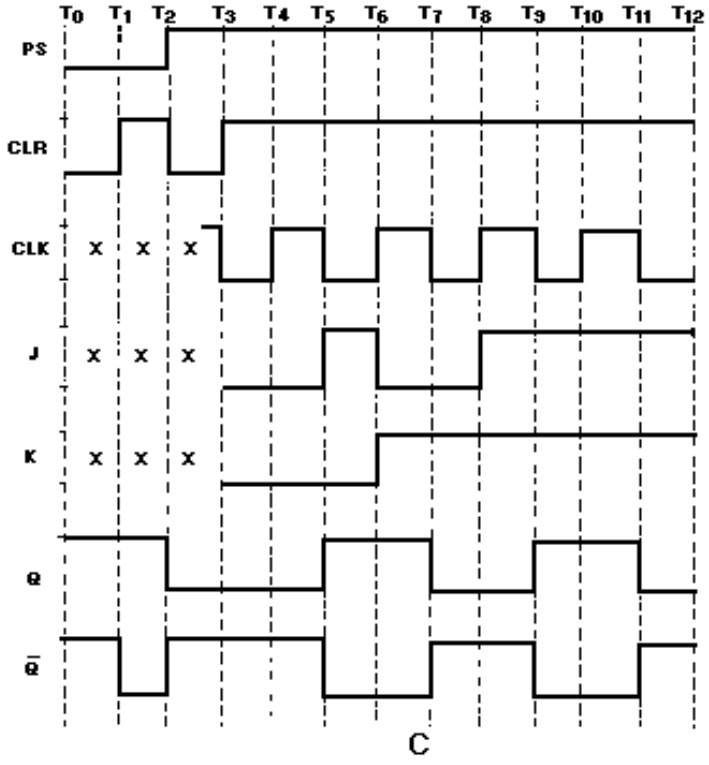


A

	INPUTS					OUTPUTS	
	PRESET	CLEAR	CLOCK	J	K	Q	$\overline{Q}$
1	0	0	X	X	X	1	1
2	0	1	X	X	X	1	0
3	1	0	X	X	X	0	1
4	1	1	↓	0	0	0	$\overline{Q}$
5	1	1	↓	1	0	1	0
6	1	1	↓	0	1	0	1
7	1	1	↓	1	1	TOGGLE	
8	1	1	0	X	X	Q	$\overline{Q}$

X = IRRELEVANT

B



C

Figure 3-18. —J-K flip-flop: A. Standard symbol; B. Truth Table; C. Timing diagram.

The J-K is a five-input device. The J and K inputs are for data. The CLK input is for the clock; and the PS and CLR inputs are the preset and clear inputs, respectively. The outputs Q and $\overline{Q}$ are the normal complementary outputs.

Observe the Truth Table and timing diagram in figure 3-18, views B and C, as the circuit is explained.

Line 1 of the Truth Table corresponds to T_0 in the timing diagram. The PS and CLR inputs are both LOW. The CLK, J, and K inputs are irrelevant. At this point the FF is jammed, and both Q and $\overline{Q}$ are HIGH. As with the R-S FF, this state cannot be used.

At T_1, PS remains LOW while CLR goes HIGH. The Q output remains HIGH and $\overline{Q}$ goes LOW. The FF is in the PRESET condition (line 2 of the Truth Table).

At T_2, PS goes HIGH, CLR goes LOW, Q goes LOW, and $\overline{Q}$ goes HIGH. At this point the FF is CLEARED (line 3 of the Truth Table). The condition of the CLK, J, and K inputs have no effect on the PS and CLR actions since these inputs override the other inputs. Starting at T_3, PS and CLR will be held at HIGHs so as not to override the other actions of the FF. Using the PS and CLR inputs only, the circuit will function as an R-S FF.

Between T_2 and T_3, the CLK input is applied to the device. Since the CLK input has an inverter, all actions will take place on the negative-going transition of the clock pulse.

Line 4 of the Truth Table shows both PS and CLR HIGH, a negative-going CLK, and J and K at 0, or LOW. This corresponds to T_3 on the timing diagram. In this condition the FF holds the previous condition of the output. In this case the FF is reset. If the circuit were set when these inputs occurred, it would remain set.

At time T_5, we have a negative-going clock pulse and a HIGH on the J input. This causes the circuit to set, Q to go HIGH, and $\overline{Q}$ to go LOW. See line 5 of the Truth Table.

At T_6, J goes LOW, K goes HIGH, and the clock is in a positive-going transition. There is no change in the output because all actions take place on the negative clock transition.

At T_7, when J is LOW, K is HIGH; the clock is going negative, the FF resets, Q goes LOW, and $\overline{Q}$ goes HIGH (line 6).

With both J and K HIGH and a negative-going clock (as at T_9 and line 7), the FF will toggle or change state with each clock pulse. It will continue to toggle as long as J and K both remain HIGH.

Line 8 of the Truth Table indicates that as long as the clock is in any condition other than a negative-going transition, there will be no change in the output regardless of the state of J or K.

As mentioned at the beginning of this section, J-K FFs may be used as R-S, T, or D FFs.

Figure 3-19 shows how a J-K can be made to perform the other functions.

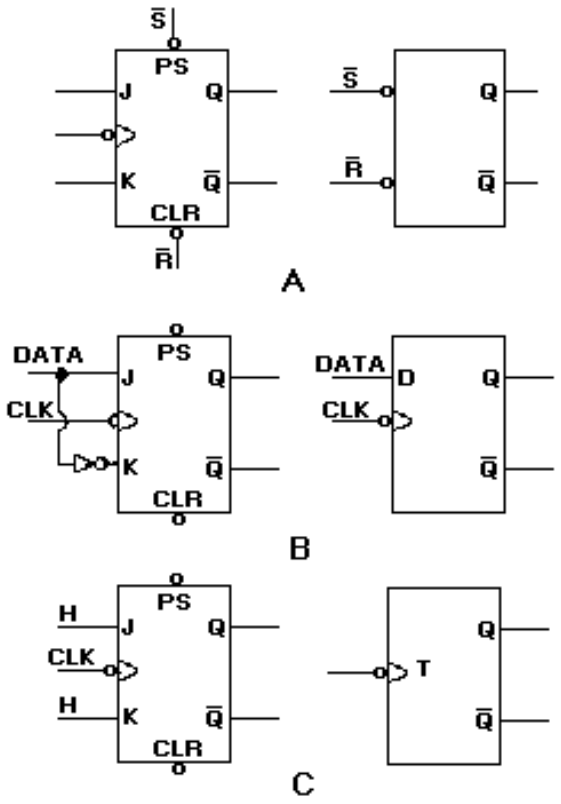


Figure 3-19. —J-K versatility: A. Using just the PS and CLR inputs; B. Data applied to the J input; C. Both J and K inputs held HIGH.

In view A, using just the PS and CLR inputs of the J-K will cause it to react like an R-S FF.

In view B, data is applied to the J input. This same data is applied to the K input through an inverter to ensure that the K input is in the opposite state. In this configuration, the J-K performs the same function as a D FF.

View C shows both the J and K inputs held at 1, or HIGH. The FF will change state or toggle with each negative-going transition of the clock just as a T FF will.

Now you can see the versatility of the J-K FF.

Q27. What type of FF can be used as an R-S, a T, or a D FF?

Q28. What will be the output of Q if J is HIGH, PS and CLR are HIGH, and the clock is going negative?

Q29. Assume that K goes HIGH and J goes LOW; when will the FF reset?

Q30. What logic levels must exist for the FF to be toggled by the clock?

Q31. What two inputs to a J-K FF will override the other inputs?

Q32. How is the J-K FF affected if PS and CLR are both LOW?

CLOCKS AND COUNTERS

Clocks and counters are found in all types of digital equipment. Although they provide different functions, they are all constructed of circuits with which you are familiar. By changing the way the circuits are interconnected, we can build timing circuits, multipliers and dividers, and storage units. In this section we will discuss the purpose, construction, and operation of these important digital circuits.

CLOCKS

Clocks have been mentioned in the preceding section with regard to their action with FFs. You will recall that the clock is a timing signal generated by the equipment to control operations. This control feature is demonstrated in both the D and J-K FFs. Remember that the clock output had to be in a certain condition for the FFs to perform their functions.

The simplest form of a clock is the astable or free-running multivibrator. A schematic diagram of a typical free-running multivibrator is shown in figure 3-20 along with its output waveforms. This multivibrator circuit is called free running because it alternates between two different output voltages during the time it is active. Outputs 1 and 2 will be equal and opposite since Q_1 and Q_2 conduct alternately. The frequency of the outputs may be altered within certain limits by varying the values of R_2C_1 and R_3C_2. You may want to review the operation of the astable multivibrator in NEETS, Module 9, *Introduction to Wave-Generation and Wave-Shaping Circuits*. Although the astable multivibrator circuit seems to produce a good, balanced square wave, it lacks the frequency stability necessary for some types of equipment.

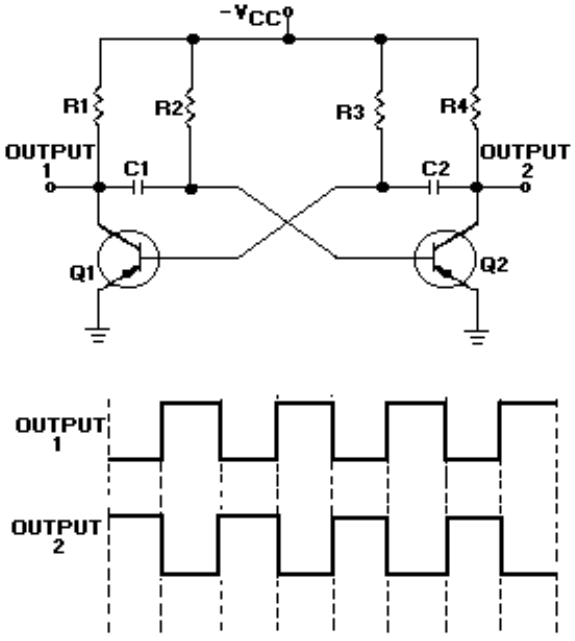


Figure 3-20. —Free-running multivibrator.

The frequency stability of the astable multivibrator can be increased by applying a trigger pulse to the circuit. The frequency of the trigger must be higher than the free-running frequency of the multivibrator. The output frequency will match the trigger frequency and produce a more stable output.

Another method of producing a stable clock pulse is to use a triggered monostable or one-shot multivibrator. You will recall from NEETS, Module 9, that a one-shot multivibrator has one stable state and will only change states when acted on by an outside source (the trigger). A block diagram of a monostable multivibrator with input and output signals is shown in figure 3-21. The duration of the output pulse is dependent on the charge time of an RC network in the multivibrator. Each trigger input results in a complete cycle in the output, as shown in figure 3-21. Trigger pulses are supplied by an oscillator.

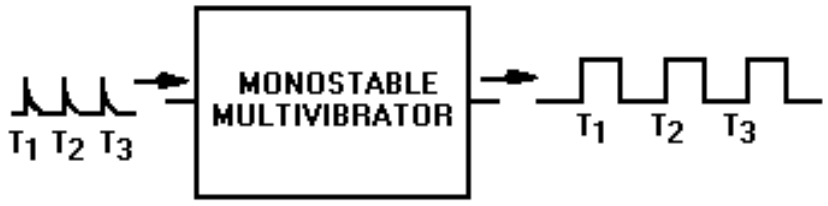


Figure 3-21. —Monostable multivibrator block diagram.

The circuits described previously are very simple clocks. However, as the complexity of the system increases, so do the timing requirements. Complex systems have multiphase clocks to control a variety of operations. Multiphase clocks allow functions involving more than one operation to be completed during a single clock cycle. They also permit an operation to extend over more than one clock cycle.

A block diagram of a two-phase clock system is shown in figure 3-22, view A. The astable multivibrator provides the basic timing for the circuit, while the one-shot multivibrators are used to shape the pulses. Outputs Q and $\overline{Q}$ are input to one-shot multivibrators 1 and 2, respectively. The resulting outputs are in phase with the inputs, but the duration of the pulse is greatly reduced as shown in view B.

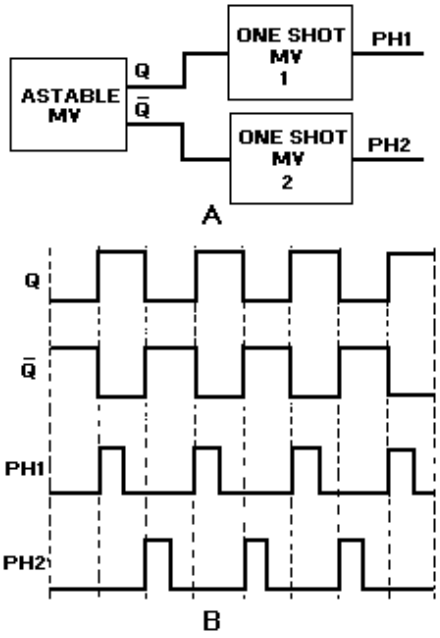


Figure 3-22. —Two-phase clock: A. Block diagram; B. Timing diagram.

Clocks are designed to provide the most efficient operation of the equipment. During the design phase, the frequency, pulse width, and the number of phases required is determined; and the clock circuit is built to meet those requirements.

Most modern high-speed equipment uses crystal-controlled oscillators as the basis for their timing networks. Crystals are stable even at extremely high frequencies.

Q33. What is a clock with regard to digital equipment?

Q34. What is the simplest type of clock circuit?

Q35. What is needed to use a monostable or one-shot multivibrator for a clock circuit?

Q36. What type of clock is used when more than one operation is to be completed during one clock cycle?

COUNTERS

A counter is simply a device that counts. Counters may be used to count operations, quantities, or periods of time. They may also be used for dividing frequencies, for addressing information in storage, or for temporary storage.

Counters are a series of FFs wired together to perform the type of counting desired. They will count up or down by ones, twos, or more.

The total number of counts or stable states a counter can indicate is called MODULUS. For instance, the modulus of a four-stage counter would be 16_{10}, since it is capable of indicating 0000_2 to 1111_2. The term *modulo* is used to describe the count capability of counters; that is, modulo-16 for a four-stage binary counter, modulo-11 for a decade counter, modulo-8 for a three-stage binary counter, and so forth.

Ripple Counters

Ripple counters are so named because the count is like a chain reaction that ripples through the counter because of the time involved. This effect will become more evident with the explanation of the following circuit.

Figure 3-23, view A, shows a basic four-stage, or modulo-16, ripple counter. The inputs and outputs are shown in view B. The four J-K FFs are connected to perform a toggle function; which, you will recall, divides the input by 2. The HIGHs on the J and K inputs enable the FFs to toggle. The inverters on the clock inputs indicate that the FFs change state on the negative-going pulse.

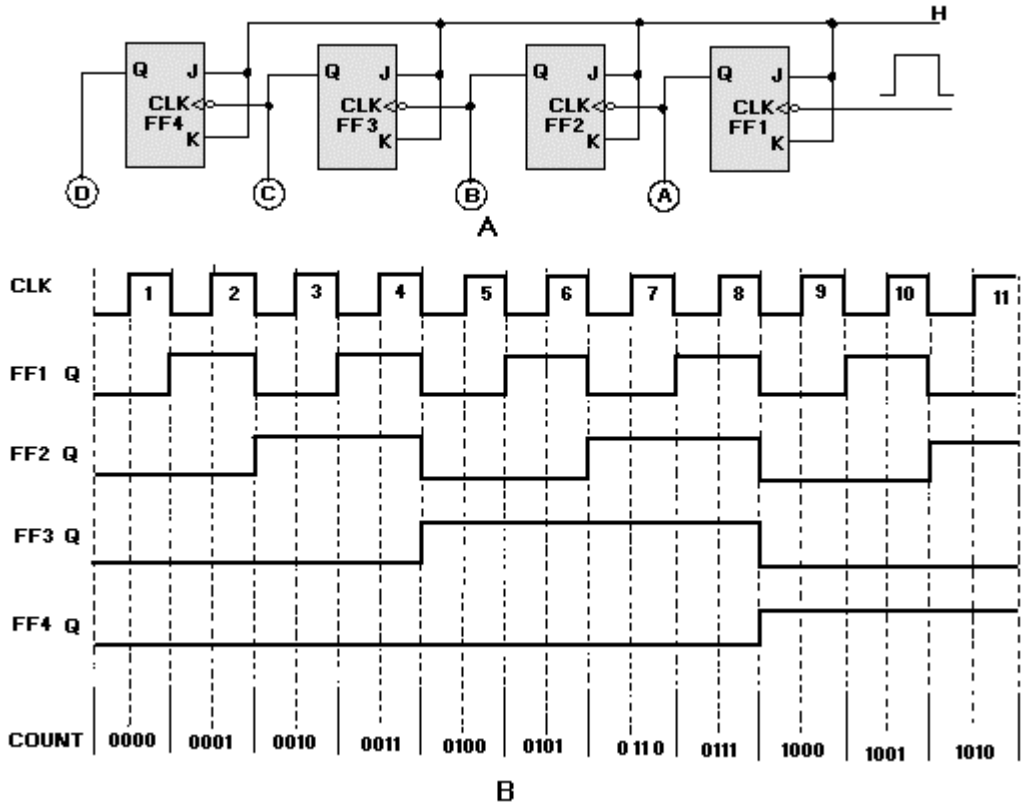


Figure 3-23. —Four-stage ripple counter: A. Logic diagram; B. Timing diagram.

Assume that A, B, C, and D are lamps and that all the FFs are reset. The lamps will all be out, and the count indicated will be 0000_2. The negative-going pulse of clock pulse 1 causes FF1 to set. This lights lamp A, and we have a count of 0001_2. The negative-going pulse of clock pulse 2 toggles FF1, causing it to reset. This negative-going input to FF2 causes it to set and causes B to light. The count after two clock pulses is 0010_2, or 2_{10}. Clock pulse 3 causes FF1 to set and lights lamp A. The setting of FF1 does not affect FF2, and lamp B stays lit. After three clock pulses, the indicated count is 0011_2.

Clock pulse 4 causes FF1 to reset, which causes FF2 to reset, which causes FF3 to set, giving us a count of 0100_2. This step shows the ripple effect.

This setting and resetting of the FFs will continue until all the FFs are set and all the lamps are lit. At that time the count will be 1111_2 or 15_{10}. Clock pulse 16 will cause FF1 to reset and lamp A to go out. This will cause FF2 through FF4 to reset, in order, and will extinguish lamps B, C, and D. The counter would then start at 0001_2 on clock pulse 17. To display a count of 16_{10} or 10000_2, we would need to add another FF.

The ripple counter is also called an ASYNCHRONOUS counter. Asynchronous means that the events (setting and resetting of FFs) occur one after the other rather than all at once. Because the ripple count is asynchronous, it can produce erroneous indications when the clock speed is high. A high-speed clock can cause the lower stage FFs to change state before the upper stages have reacted to the previous clock pulse. The errors are produced by the FFs' inability to keep up with the clock.

Synchronous Counter

High-frequency operations require that all the FFs of a counter be triggered at the same time to prevent errors. We use a SYNCHRONOUS counter for this type of operation.

The synchronous counter is similar to a ripple counter with two exceptions: The clock pulses are applied to each FF, and additional gates are added to ensure that the FFs toggle in the proper sequence.

A logic diagram of a three-state (modulo-8) synchronous counter is shown in figure 3-24, view A. The clock input is wired to each of the FFs to prevent possible errors in the count. A HIGH is wired to the J and K inputs of FF1 to make the FF toggle. The output of FF1 is wired to the J and K inputs of FF2, one input of the AND gate, and indicator A. The output of FF2 is wired to the other input of the AND gate and indicator B. The AND output is connected to the J and K inputs of FF3. The C indicator is the only output of FF3.

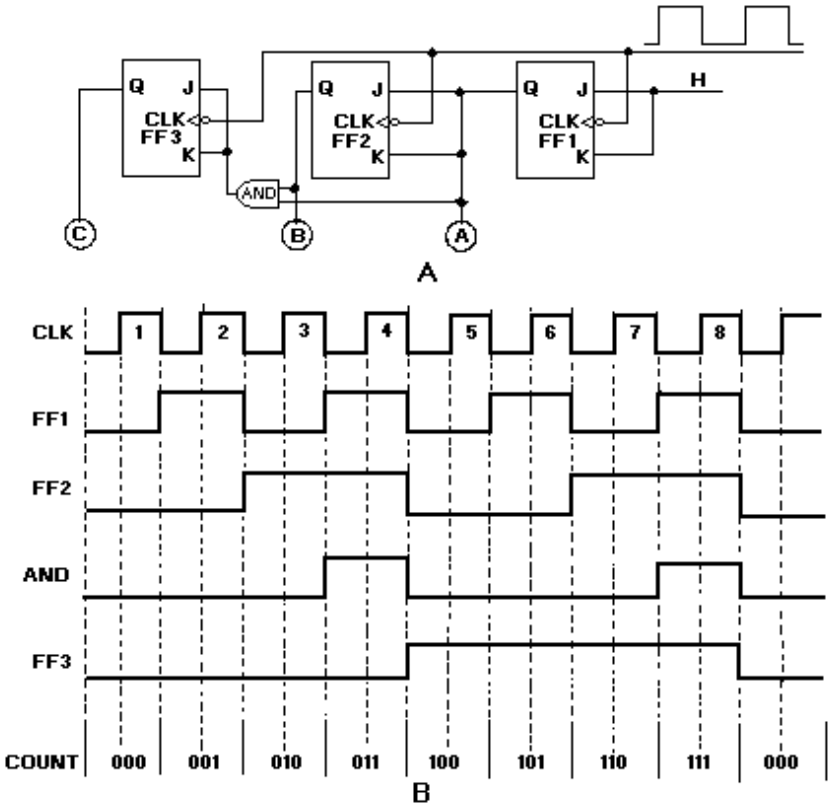


Figure 3-24. —Three-stage synchronous counter: A. Logic diagram; B. Timing Diagram.

During the explanation of this circuit, you should follow the logic diagram, view A, and the pulse sequences, view B.

Assume the following initial conditions: The outputs of all FFs, the clock, and the AND gate are 0; the J and K inputs to FF1 are HIGH. The negative-going portion of the clock pulse will be used throughout the explanation.

Clock pulse 1 causes FF1 to set. This HIGH lights lamp A, indicating a binary count of 001. The HIGH is also applied to the J and K inputs of FF2 and one input of the AND gate. Notice that FF2 and

FF3 are unaffected by the first clock pulse because the J and K inputs were LOW when the clock pulse was applied.

As clock pulse 2 goes LOW, FF1 resets, turning off lamp A. In turn, FF2 will set, lighting lamp B and showing a count of 010_2. The HIGH from FF2 is also felt by the AND gate. The AND gate is not activated at this time because the signal from FF1 is now a LOW. A LOW is present on the J and K inputs of FF3, so it is not toggled by the clock.

Clock pulse 3 toggles FF1 again and lights lamp A. Since the J and K inputs to FF2 were LOW when pulse 3 occurred, FF2 does not toggle but remains set. Lamps A and B are lit, indicating a count of 011_2. With both FF1 and FF2 set, HIGHs are input to both inputs of the AND gate, resulting in HIGHs to J and K of FF3. No change occurred in the output of FF3 on clock pulse 3 because the J and K inputs were LOW at the time.

Just before clock pulse 4 occurs, we have the following conditions: FF1 and FF2 are set, and the AND gate is outputting a HIGH to the J and K inputs of FF3. With these conditions all of the FFs will toggle with the next clock pulse.

At clock pulse 4, FF1 and FF2 are reset, and FF3 sets. The output of the AND gate goes to 0, and we have a count of 100_2.

It appears that the clock pulse and the AND output both go to 0 at the same time, but the clock pulse arrives at FF3 before the AND gate goes LOW because of the transit time of the signal through FF1, FF2, and the AND gate.

Between pulses 4 and 8, FF3 remains set because the J and K inputs are LOW. FF1 and FF2 toggle in the same sequence as they did on clock pulses 1, 2, and 3.

Clock pulse 7 results in all of the FFs being set and the AND gate output being HIGH. Clock pulse 8 causes all the FFs to reset and all the lamps to turn off, indicating a count of 000_2 . The next clock pulse (9) will restart the count sequence.

Q37. What is the modulus of a five-stage binary counter?

Q38. An asynchronous counter is also called a _____ counter.

Q39. J-K FFs used in counters are wired to perform what function?

Q40. What type of counter has clock pulses applied to all FFs?

Q41. In figure 3-24, view A, what logic element enables FF3 to toggle with the clock?

Q42. What is the largest count that can be indicated by a four-stage counter?

Decade Counter

A decade counter is a binary counter that is designed to count to 10_{10}, or 1010_2. An ordinary four-stage counter can be easily modified to a decade counter by adding a NAND gate as shown in figure 3-25. Notice that FF2 and FF4 provide the inputs to the NAND gate. The NAND gate outputs are connected to the CLR input of each of the FFs.

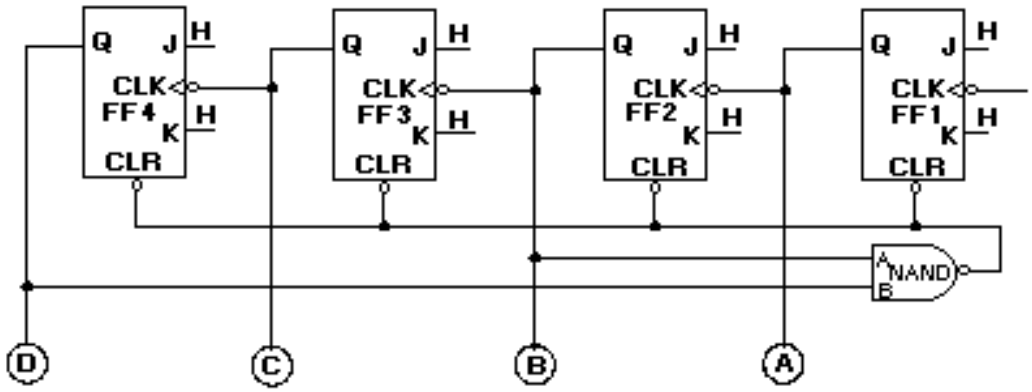


Figure 3-25. —Decade counter.

The counter operates as a normal counter until it reaches a count of 1010_2, or 10_{10}. At that time, both inputs to the NAND gate are HIGH, and the output goes LOW. This LOW applied to the CLR input of the FFs causes them to reset to 0. Remember from the discussion of J-K FFs that CLR and PS or PR override any existing condition of the FF. Once the FFs are reset, the count may begin again. The following table shows the binary count and the inputs and outputs of the NAND gate for each count of the decade counter:

BINARY COUNT		NAND GATE INPUTS	NAND GATE OUTPUT
*******	A	B	*******
0000	0	0	1
0001	0	0	1
0010	1	0	1
0011	1	0	1
0100	0	0	1
0101	0	0	1
0110	1	0	1
0111	1	0	1
1000	0	1	1
1001	0	1	1
1010	1	1	0

Changing the inputs to the NAND gate can cause the maximum count to be changed. For instance, if FF4 and FF3 were wired to the NAND gate, the counter would count to 1100_2 (12_{10}), and then reset.

Q43. How many stages are required for a decade counter?

Q44. In figure 3-25, which two FFs must be HIGH to reset the counter?

Ring Counter

A ring counter is defined as a loop of bistable devices (flip-flops) interconnected in such a manner that only one of the devices may be in a specified state at one time. If the specified condition is HIGH,

then only one device may be HIGH at one time. As the clock, or input, signal is received, the specified state will shift to the next device at a rate of 1 shift per clock, or input, pulse.

Figure 3-26, view A, shows a typical four-stage ring counter. This particular counter is composed of R-S FFs. J-K FFs may be used as well. Notice that the output of each AND gate is input to the R, or reset side, of the nearest FF and to the S, or set side, of the next FF. The Q output of each FF is applied to the B input of the AND gate that is connected to its own R input.

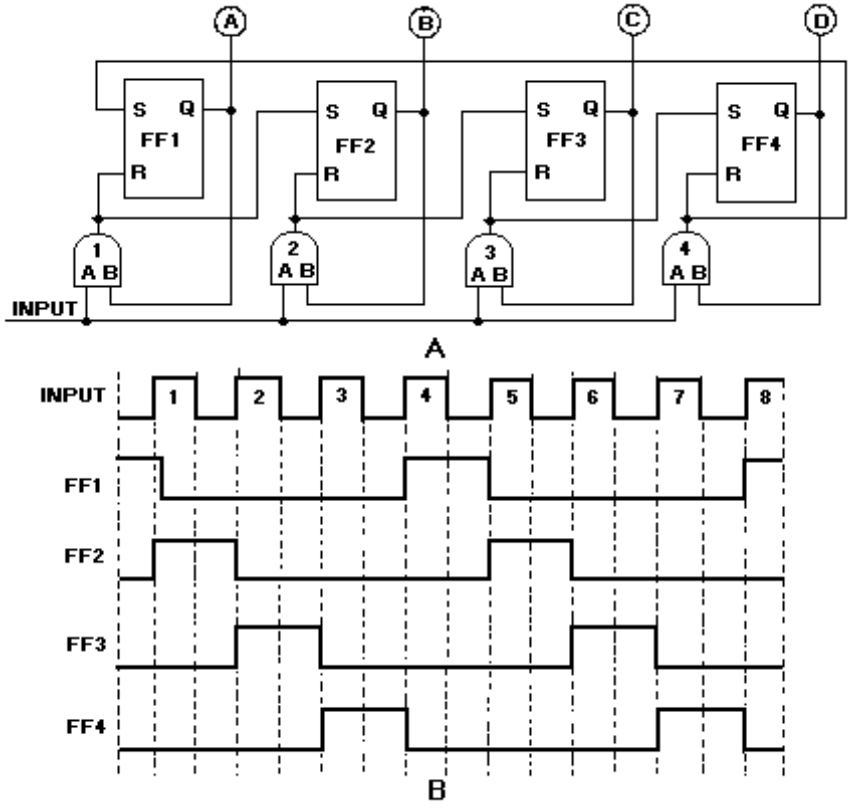


Figure 3-26. —Ring counter: A. Logic diagram; B. Timing diagram.

The circuit input may be normal CLK pulses or pulses from elsewhere in the equipment that would indicate some operation has been completed.

Now, let's look at the circuit operation and observe the signal flow as shown in figure 3-26, view B.

For an initial condition, let's assume that the output of FF1 is HIGH and that the input and FF2, FF3, and FF4 are LOW. Under these conditions, lamp A will be lit; and lamps B, C, and D will be extinguished. The HIGH from FF1 is also applied to the B input of AND gate 1.

The first input pulse is applied to the A input of each of the AND gates. The B inputs to AND gates 2, 3, and 4 are LOW since the outputs of FF2, FF3, and FF4 are LOW. AND gate 1 now has HIGHs on both inputs and produces a HIGH output. This HIGH simultaneously resets FF1 and sets FF2. Lamp A then goes out, and lamp B goes on. We now have a HIGH on AND gate 2 at the B input. We also have a LOW on AND gate 1 at input B.

Input pulse 2 will produce a HIGH output from AND gate 2 since AND gate 2 is the only one with HIGHs on both inputs. The HIGH from AND gate 2 causes FF2 to reset and FF3 to set. Indicator B goes out and C goes on.

Pulse 3 will cause AND gate 3 to go HIGH. This results in FF3 being reset and FF4 being set. Pulse 4 causes FF4 to reset and FF1 to set, bringing the counter full circle to the initial conditions. As long as the counter is operational, it will continue to light the lamps in sequence — 1, 2, 3, 4; 1, 2, 3, 4, etc.

As we stated at the beginning of this section, only one FF may be in the specified condition at one time. The specified condition shifts one position with each input pulse.

Q45. In figure 3-26, view A, which AND gate causes FF3 to set?

Q46. Which AND gate causes FF3 to reset?

Q47. What causes the specified condition to shift position?

Q48. If the specified state is OFF, how many FFs may be off at one time?

Down Counters

Up to this point the counters that you have learned about have been up counters (with the exception of the ring counter). An up counter starts at 0 and counts to a given number. This section will discuss DOWN counters, which start at a given number and count down to 0.

Up counters are sometimes called INCREMENT counters. Increment means to increase. Down counters are called DECREMENT counters. Decrement means to decrease.

A three-stage, ripple down counter is shown in figure 3-27, view A. Notice that the PS (preset) input of the J-K FFs is used in this circuit. HIGHs are applied to all the J and K inputs. This enables the FFs to toggle on the input pulses.

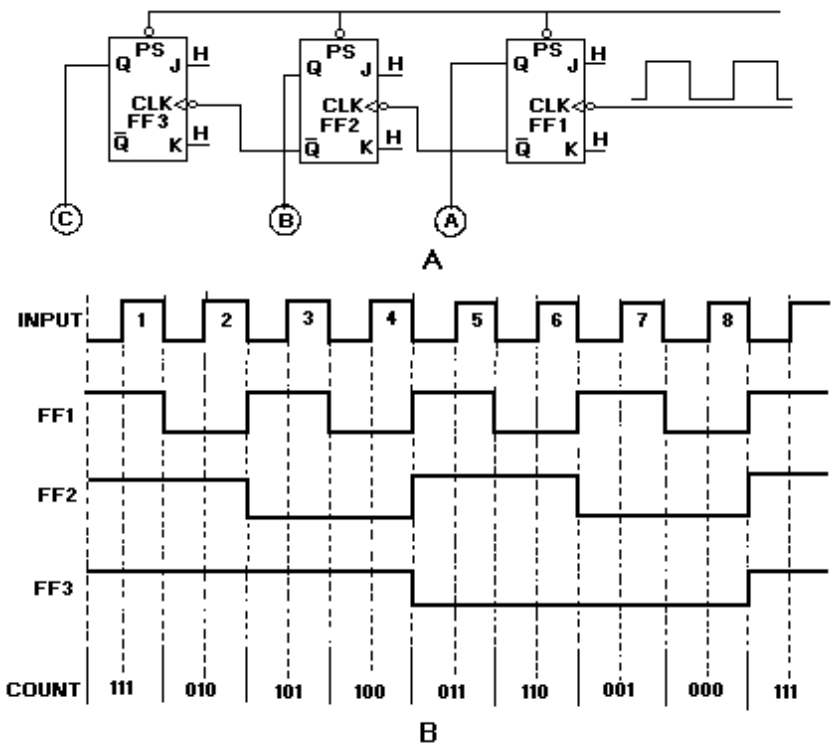


Figure 3-27. —Down counter: A. Logic diagram; B. Timing diagram.

A negative-going pulse is applied to all PS terminals to start the countdown. This causes all the FFs to set and also lights indicators A, B, and C. The beginning count is 111_2 (7_{10}). At the same time, LOWs are applied to the CLK inputs of FF2 and FF3, but they do not toggle because the PS overrides any change. All actions in the counter will take place on the negative-going portion of the input pulse. Let's go through the pulse sequences in figure 3-27, view B.

CP1 causes FF1 to toggle and output Q to go LOW. Lamp A is turned off. Notice that $\overline{Q}$ goes HIGH but no change occurs in FF2 or FF3. Lamps B and C are now on, A is off, and the indicated count is 110_2 (6_{10}).

CP2 toggles FF1 again and lights lamp A. When Q goes HIGH, $\overline{Q}$ goes LOW. This negative-going signal causes FF2 to toggle and reset. Lamp B is turned off, and a HIGH is felt at the CLK input of FF3. The indicated count is 101_2 (5_{10}); lamps A and C are on, and B is off.

At CP3, FF1 toggles and resets. Lamp A is turned off. A positive-going signal is applied to the CLK input of FF2. Lamp B remains off and C remains on. The count at this point is 100_2 (4_{10}).

CP4 toggles FF1 and causes it to set, lighting lamp A. Now FF1, output $\overline{Q}$, goes LOW causing FF2 to toggle. This causes FF2 to set and lights lamp B. Output of FF2, $\overline{Q}$, then goes LOW, which causes FF3 to reset and turn off lamp C. The indicated count is now 011_2 (3_{10}).

The next pulse, CP5, turns off lamp A but leaves B on. The count is now 010_2. CP6 turns on lamp A and turns off lamp B, for a count of 001_2. CP7 turns off lamp A. Now all the lamps are off, and the counter indicates 000.

On the negative-going signal of CP8, all FFs are set, and all the lamps are lighted. The CLK pulse toggles FF1, making output Q go HIGH. As output $\overline{Q}$ goes LOW, the negative-going signal causes FF2 to toggle. As FF2, output Q, goes HIGH, output $\overline{Q}$ goes LOW, causing FF3 to toggle and set. As each FF sets, its indicator lamp lights. The counter is now ready to again start counting down from 111_2 with the next CLK pulse.

Q49. How many FFs are required to count down from 15_{10}?

Q50. What signal causes FF2 to toggle?

REGISTERS

A register is a temporary storage device. Registers are used to store data, memory addresses, and operation codes. Registers are normally referred to by the number of stages they contain or by the number of bits they will store. For instance, an eight-stage register would be called an 8-bit register. The contents of the register is also called a WORD. The contents of an 8-bit register is an 8-bit word. The contents of a 4-bit register is a 4-bit word and so forth.

Registers are also used in the transfer of data to and from input and output devices such as teletypes, printers, and cathode-ray tubes.

Most registers are constructed of FFs and associated circuitry. They permit us to load or store data and to transfer the data at the proper time.

PARALLEL REGISTERS

Parallel registers are designed to receive or transfer all bits of data or information simultaneously.

A 4-bit parallel register is shown in figure 3-28. The data inputs are A, B, C, and D. The FFs store the data until it is needed. AND gates 5, 6, 7, and 8 are the transfer gates.

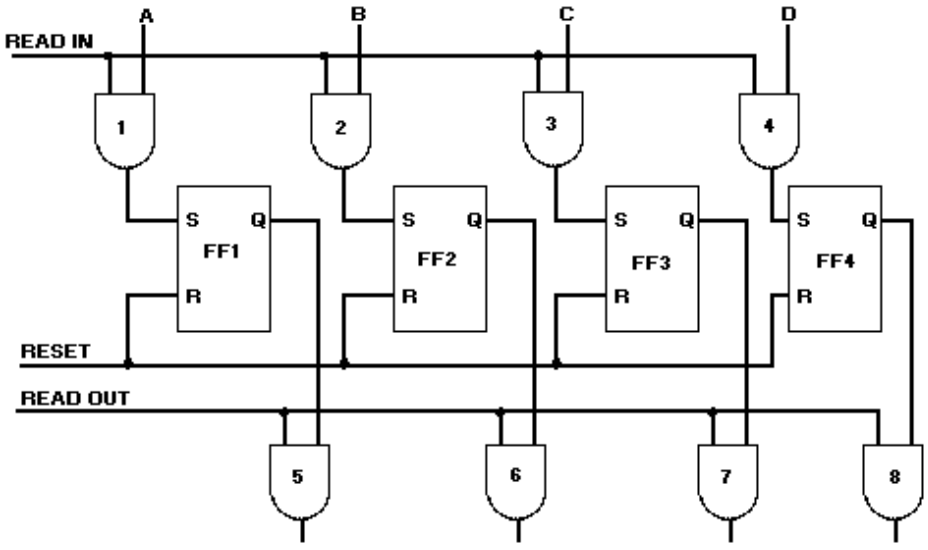


Figure 3-28. —Four-bit parallel register.

Before we go through the operation of the register, let's set some initial conditions. Assume that inputs A, B, and D are HIGH and that FF2 and FF4 are set from a previous operation. The READ IN, READ OUT, and RESET inputs are all LOW.

To begin the operation, we apply a reset pulse to the RESET input of all the FFs, clearing the Q outputs to LOWs. This step ensures against any erroneous data transfer that would occur because of the states of FF2 and FF4.

Inputs A, B, C, and D are input to gates 1, 2, 3, and 4, respectively. When the READ IN input goes HIGH, AND gates 1, 2, and 4 go HIGH, causing FF1, FF2, and FF4 to set. The output of AND gate 3 does not change since the C input is LOW. The 4-bit word, 1101, is now stored in the register. The outputs of FFs 1, 2, 3, and 4 are applied to AND gates 5, 6, 7, and 8, respectively.

When the data is required for some other operation, a positive-going pulse is applied to the READ OUT inputs of the AND gates. This HIGH, along with the HIGHs from the FFs, causes the outputs of AND gates 5, 6, and 8 to go HIGH. Since the Q output of FF3 is LOW, the output of gate 7 will be LOW. The 4-bit word, 1101, is transferred to where it is needed.

Q51. How many stages are required to store a 16-bit word?

Q52. Simultaneous transfer of data may be accomplished with what type of register?

Q53. How are erroneous transfers of data prevented?

SHIFT REGISTERS

A shift register is a register in which the contents may be shifted one or more places to the left or right. This type of register is capable of performing a variety of functions. It may be used for serial-to-parallel conversion and for scaling binary numbers.

Before we get into the operation of the shift register, let's discuss serial-to-parallel conversion, parallel-to-serial conversion, and scaling.

Serial and Parallel Transfers and Conversion

Serial and parallel are terms used to describe the method in which data or information is moved from one place to another. **SERIAL TRANSFER** means that the data is moved along a single line one bit at a time. A control pulse is required to move each bit. **PARALLEL TRANSFER** means that each bit of data is moved on its own line and that all bits transfer simultaneously as they did in the parallel register. A single control pulse is required to move all bits.

Figure 3-29 shows how both of these transfers occur. In each case, the four-bit word 1101 is being transferred to a storage device. In view A, the data moves along a single line. Each bit of the data will be stored by an individual control pulse. In view B, each bit has a separate input line. One control pulse will cause the entire word to be stored.

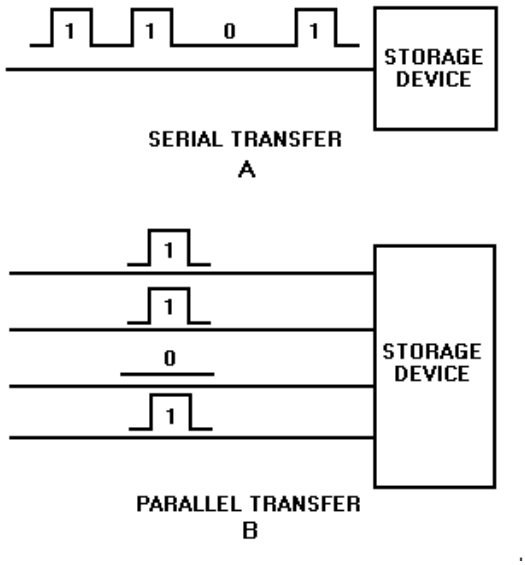


Figure 3-29. —Data transfer methods: A. Serial transfer; B. Parallel transfer.

Serial-to-parallel conversion or parallel-to-serial conversion describes the manner in which data is stored in a storage device and the manner in which that data is removed from the storage device.

Serial-to-parallel conversion means that data is transferred into the storage device or register in serial fashion and removed in parallel fashion, as in figure 3-30, view A. Parallel-to-serial conversion means the data is transferred into the storage device in parallel and removed as serial data, as shown in view B.

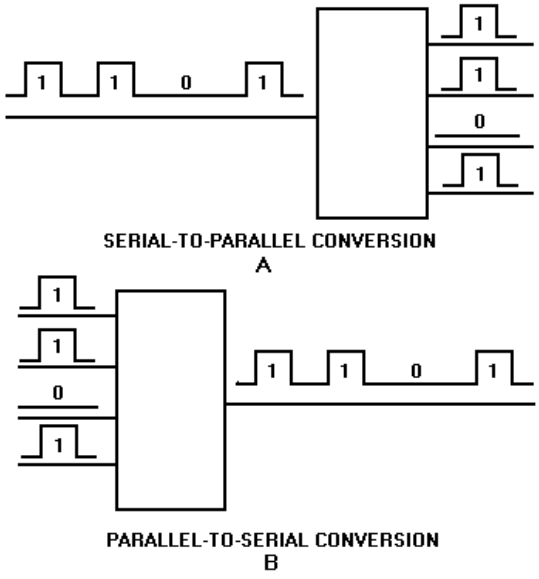


Figure 3-30. —Data conversion methods: A. Serial-to-parallel; B. Parallel-to-serial.

Serial transfer takes time. The longer the word length, the longer the transfer will take. Although parallel transfer is much faster, it requires more circuitry to transfer the data.

Scaling

SCALING means to change the magnitude of a number. Shifting binary numbers to the left increases their value, and shifting to the right decreases their value. The increase or decrease in value is based on powers of 2.

A shift of one place to the left increases the value by a power of 2, which in effect is multiplying the number by 2. To demonstrate this, let's assume that the following block diagram is a 5-bit shift register containing the binary number 01100.

0	1	1	0	0

Shifting the entire number one place to the left will put the register in the following condition:

1	1	0	0	0

The binary number 01100 has a decimal equivalent of 12. If we convert 11000_2 to decimal, we find it has a value of 24_{10}. By shifting the number one place to the left, we have multiplied it by 2. A shift of two places to the left would be the equivalent of multiplying the number by 2^2, or 4; three places by 2^3, or 8; and so forth.

Shifting a binary number to the right decreases the value of the number by a power of 2 for each place. Let's look at the same 5-bit register containing 01100_2 and shift the number to the right.

0	1	1	0	0

A shift of one place to the right will result in the register being in the following condition:

0	0	1	1	0

By comparing decimal equivalents you can see that we have decreased the value from 12_{10} to 6_{10}. We have effectively divided the number by 2. A shift of two places to the right is the equivalent of dividing the number by 2^2, or 4; three places by 2^3; or 8; and so forth.

Shift Register Operations

Figure 3-31shows a typical 4-bit shift register. This particular register is capable of left shifts only. There are provisions for serial and parallel input and serial and parallel output. Additional circuitry would be required to make right shifts possible.

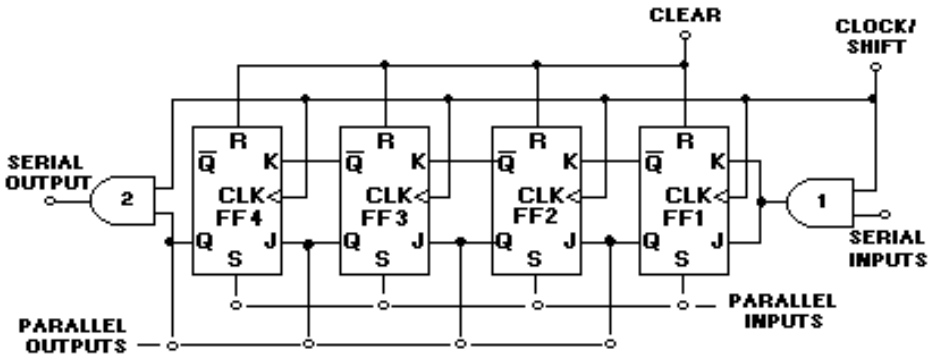


Figure 3-31. —Shift register.

Before any operation takes place, a CLEAR pulse is applied to the RESET terminal of each FF to ensure that the Q output is LOW.

The simplest modes of operation for this register are the parallel inputs and outputs. Parallel data is applied to the SET inputs of the FFs and results in either a 1 or 0 output, depending on the input. The outputs of the FFs may be sampled for parallel output. In this mode, the register functions just like the parallel register covered earlier in this section.

Parallel-to-Serial Conversion

Now let’s look at parallel-to-serial conversion. We will use the 4-bit shift register in figure 3-31 and the timing sequence in figure 3-32 to aid you in understanding the operations.

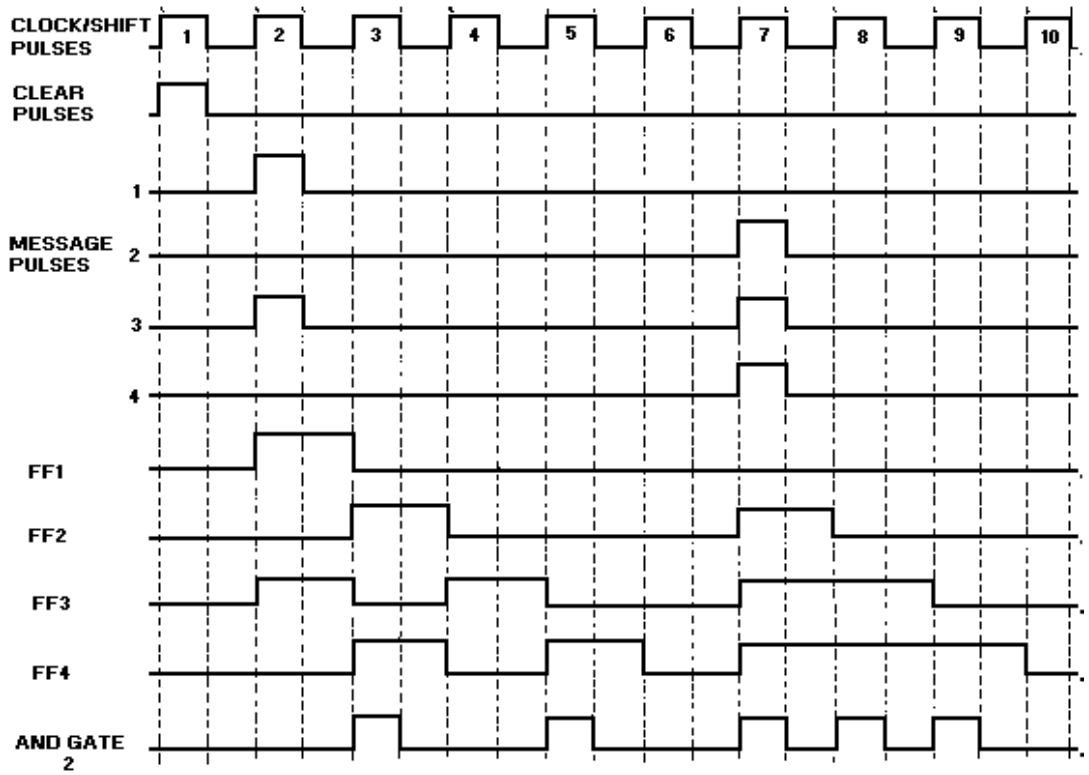


Figure 3-32. —Parallel-to-serial conversion timing diagram.

At CP1, a CLEAR pulse is applied to all the FFs, resetting the register to a count of 0. The number 0101_2 is applied to the parallel inputs at CP2, causing FF1 and FF3 to set. At this point, the J inputs of FF2 and FF4 are HIGH. AND gate 2 has a LOW output since the FF4 output is LOW. This LOW output represents the first digit of the number 0101_2 to be output in serial form. At the same time we have HIGHs on the K inputs of FF1 and FF3. (Notice the NOT symbol on FF1 at input K. With no serial input to AND gate 1, the output is LOW; therefore, the K input to FF1 is held HIGH). With these conditions CP3 causes FF1 and FF3 to reset and FF2 and FF4 to set. The HIGH output of FF4, along with CP3, causes AND gate 2 to output a HIGH. This represents the second digit of the number 0101_2.

At CP4, FF2 and FF4 reset, and FF3 sets. FF1 remains reset because of the HIGH at the K input. The output of AND gate 2 goes LOW because the output of FF4 is LOW and the third digit of the number is output on the serial line. CP5 causes FF4 to set and FF3 to reset. CP5 and the HIGH from FF4 cause AND gate 2 to output the last digit of the number on the serial line. It took a total of four CLK pulses to input the number in parallel and output it in serial.

CP6 causes FF4 to reset and effectively clears the register for the next parallel input.

Between CP7 and CP10, the number 1110_2 is input as parallel data and output as serial data.

Serial-to-Parallel Conversion

Serial input is accomplished much in the same manner as serial output. Instead of shifting the data out one bit at a time, we shift the data in one bit at a time.

To understand this conversion, you should again use figure 3-31and also the timing diagram shown in figure 3-33. In this example we will convert the number 1011_2 from serial data to parallel data.

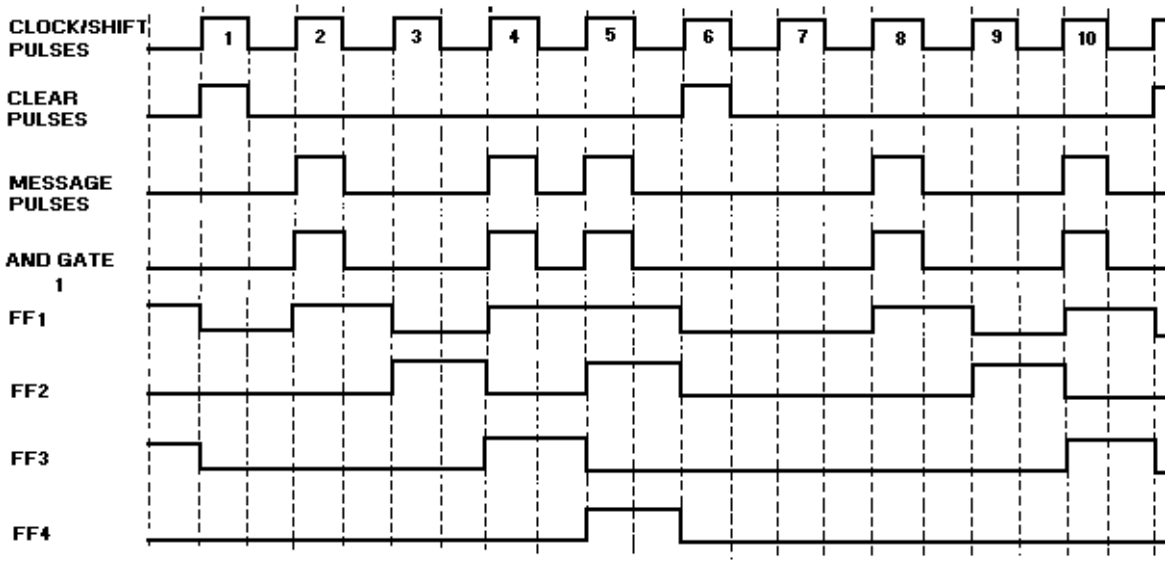


Figure 3-33. —Serial-to-parallel conversion timing diagram.

A CLEAR pulse resets all the FFs at CP1. At CP2, the most significant bit of the data is input to AND gate 1. This HIGH along with the clock pulse causes AND gate 1 to output a HIGH. The HIGH from the AND gate and the clock pulse applied to FF1 cause the FF to set. FFs 2, 3, and 4 are held reset. At this point, the MSD of the data has been shifted into the register.

The next bit of data is a 0. The output of AND gate 1 is LOW. Because of the inverter on the K input of FF1, the FF senses a HIGH at that input and resets. At the same time this is occurring, the HIGH on the J input of FF2 (from FF1) and the CLK cause FF2 to set. The two MSDs, 1 and 0, are now in the register.

CP4 causes FF3 to set and FF2 to reset. FF1 is set by the CLK pulse and the third bit of the number. The register now contains 0101_2, as a result of shifting the first three bits of data.

The remaining bit is shifted into the register by CP5. FF1 remains set, FF2 sets, FF3 resets, and FF4 sets. At this point, the serial transfer is complete. The binary word can be sampled on the parallel output lines. Once the parallel data is transferred, a CLEAR pulse resets the FFs (CP6), and the register is ready to input the next word.

Scaling Operation

Using the shift register shown in figure 3-31 for scaling a number is quite simple. The number to be scaled is loaded into the register either in serial or parallel form. Once the data is in the register, the scaling takes place in the same manner as that for shifting the data for serial output. A single clock pulse will cause each bit of data to shift one place to the left. Remember that each shift is the equivalent of increasing the value by a power of 2. The scaled data is read from the parallel outputs. Care must be taken not to over shift the data to the point that the MSDs are shifted out of the register.

Q54. Serial-to-parallel and parallel-to-serial conversions are accomplished by what type of circuit?

Q55. What type of data transfer requires the most time?

Q56. What is the main disadvantage of parallel transfer?

Q57. How many FFs would be required for an 8-bit shift register?

Q58. How many clock pulses are required to output a 4-bit number in serial form?

Q59. Two shifts to the left are equal to increasing the magnitude of a number by how much?

Q60. To increase the magnitude of a number by 2^3, you must shift the number how many times and in what direction?

LOGIC FAMILIES

Logic families are groups of logic circuits that are based on particular types of elements (resistors, transistors, and so forth). Families are identified by the manner in which the elements are connected, and, in some cases, by the types of elements used.

Logic circuits of a particular family can be interconnected without having to use additional circuitry. In other words, the output of one logic circuit can be used as the input to another logic circuit. This feature is known as compatibility. All circuits within a logic family will be compatible with the other circuits within that family.

As a technician, your responsibility will be to identify defective parts and repair or replace them as required. It will be beneficial for you to have a basic knowledge of the types of logic that are used in digital equipment.

Logic circuits are usually manufactured as integrated circuits and packaged in dual-inline packages (DIP), modified transistor outlines (TO), or flat packs. These packaging techniques are described in NEETS, Module 7, *Introduction to Solid-State Devices and Power Supplies.*

Circuitry in a package is normally shown using standard logic symbols instead of individual components such as transistors, diodes, and so forth. Figure 3-34 shows four examples of this type of packaging. The numbered blocks (1-14 and 1-16) are the pins on the package. Circuit packages are also identified by a manufacturer's part number. Similar circuits produced by different manufacturers will not carry the same identification numbers in all cases.

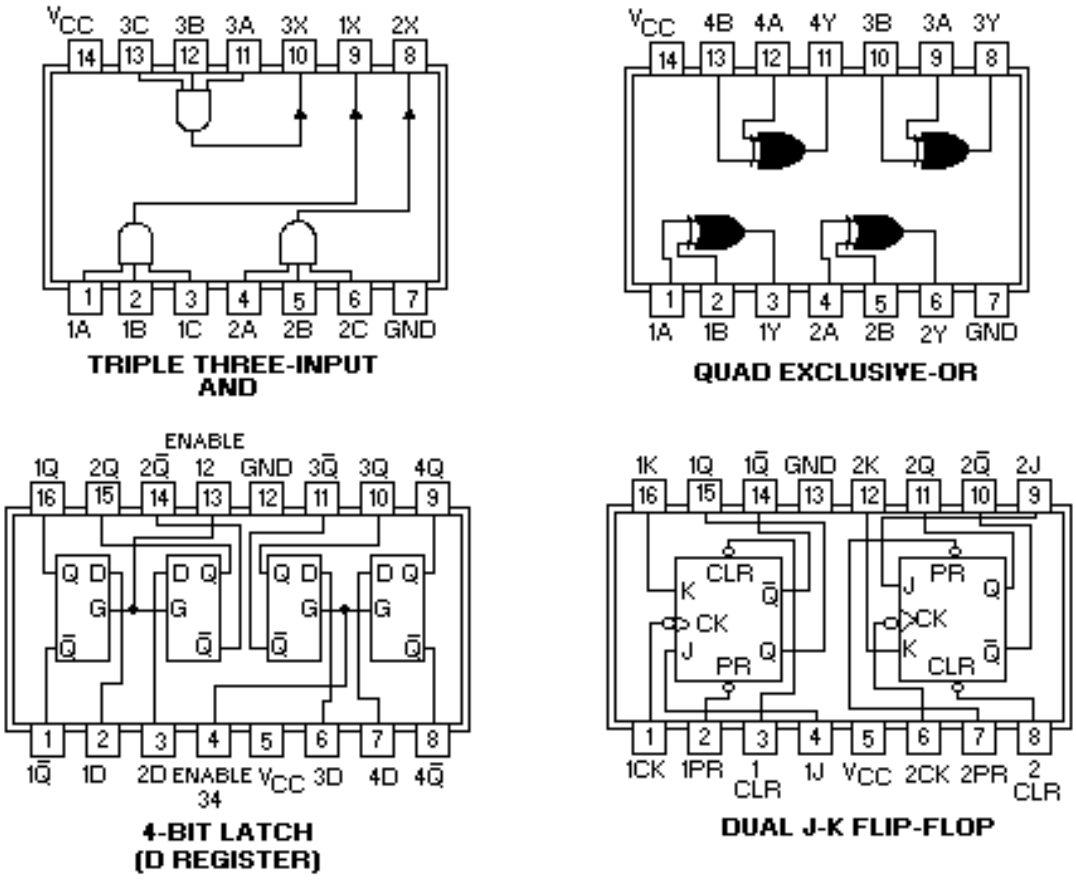


Figure 3-34. —Logic packages.

As mentioned before, logic families are identified by the elements used and the manner in which the elements are used. A brief description of some of the more common logic families follows.

RTL (RESISTOR-TRANSISTOR LOGIC)

In this type of logic, inputs are applied to resistors, and the output is produced by a transistor. RTL is normally constructed from discrete components (individual resistors and transistors). Some circuits are manufactured as integrated circuits and packaged in modified transistor outline (TO) packages, as shown in figure 3-35. An in-depth coverage of circuit packaging can be found in NEETS, Module 14, *Introduction to Microelectronics*.

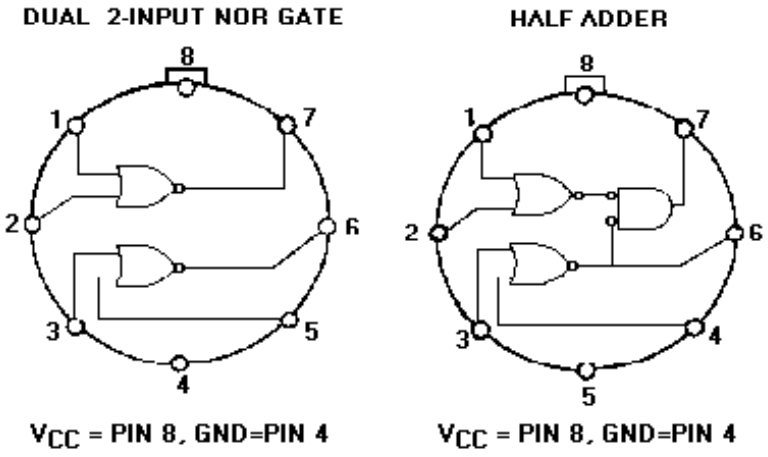


Figure 3-35. —RTL integrated circuits.

DTL (DIODE TRANSISTOR LOGIC)

Input signals are applied to diodes in this logic family. The diodes either conduct or cut off and produce the desired output from the transistor. DTL is normally found in dual-inline packages (DIP) as well as older discrete component logic.

TTL (TRANSISTOR-TRANSISTOR LOGIC)

In TTL, transistors with multiple emitters are used for the logic inputs. Additional transistors are used to produce the desired output. TTL is normally packed in DIPs and is quite common in military equipment.

CMOS (COMPLEMENTARY METAL OXIDE SEMICONDUCTORS)

The CMOS logic circuits use metal oxide semiconductors similar to field-effect transistors (FETs).

LOGIC FAMILY USE

The logic family used in a piece of equipment is determined by the design engineers. The type of logic used will be based on the requirements of the equipment and on what family best fulfills the requirements.

The use of integrated circuits enables designers to produce equipment that is very small and highly efficient when compared to other methods of construction. The block diagram shown in figure 3-36, view A, represents an 8-bit, serial-input and parallel-output shift register. This circuit is contained in a standard 14-pin DIP measuring about 0.75 inch long and 0.25 inch wide. View B shows this circuit package.

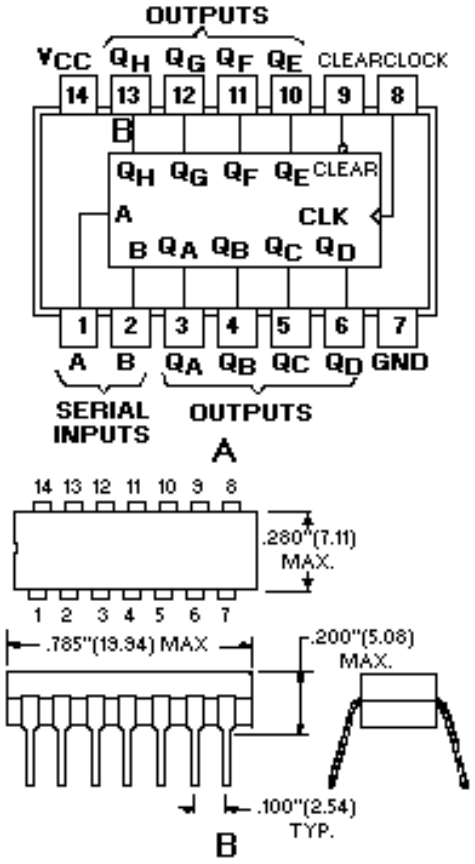


Figure 3-36. —Integrated logic circuits: A. Shift register; B. Logic package.

Q61. What are RTL, DTL, and TTL examples of?

Q62. What type of logic family uses diodes in the input?

Q63. What is the most common type of integrated circuit packaging found in military equipment?

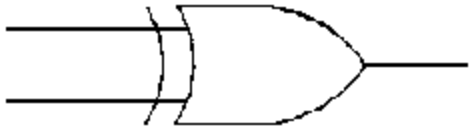
Q64. Circuits that can be interconnected without additional circuitry are known as _____ circuits.

SUMMARY

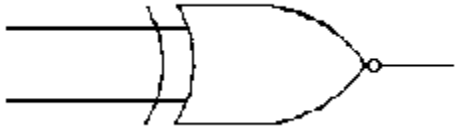
Now that you have completed this chapter, you should have a basic understanding of the more common special logic circuits. The following is a summary of the emphasized terms and points found in the "Special Logic Circuits" chapter.

SPECIAL LOGIC CIRCUITS perform arithmetic and logic operations; input, output, store and transfer information; and provide proper timing for these operations.

EXCLUSIVE OR (X-OR) circuits produce a 1 output when ONLY one input is HIGH. Can be used as a quarter adder.

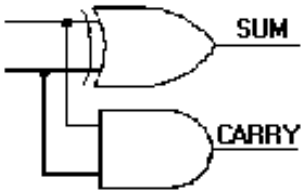


EXCLUSIVE NOR (X-NOR) circuits produce a 1 output when all inputs are 0 and when more than 1 input is 1.



QUARTER ADDER circuits produce the sum of two numbers but do not generate a carry.

HALF ADDER circuits produce the sum of two numbers and generate a carry.



FULL ADDER circuits add a carry to obtain the correct sum.

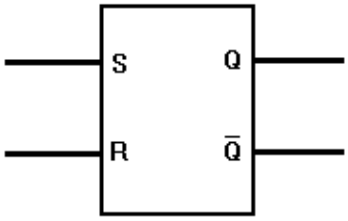
PARALLEL ADDER circuits use full adders connected in parallel to accommodate the addition of multiple-digit numbers.

STANDARD SYMBOLS depict logic circuitry with blocks, showing only inputs and outputs. One block may contain many types of gates and circuits.

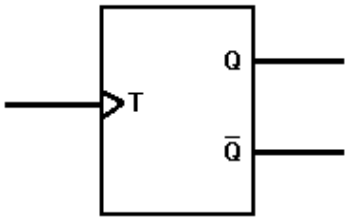
SUBTRACTION in binary is accomplished by complementing and adding.

FLIP-FLOP are bistable multivibrators used for storage, timing, arithmetic operations, and transfer of information.

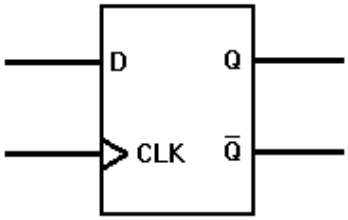
R-S FFs have the Q output of the FF HIGH in the set mode and LOW in the reset mode.



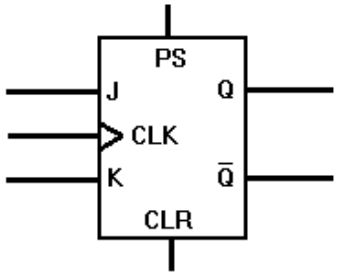
T (TOGGLE) FF$_S$ change state with each pulse applied to the input. Each T FF will divide the input by 2.



D FF is used to store data at a predetermined time.



J-K FF is the most versatile FF. J-Ks can perform the same functions as all the other FFs.



CLOCKS are circuits that generate the timing and control signals for other operations.

COUNTERS are used to count operations, quantities, or periods of time. They can be used to divide frequencies, to address information in storage, or as temporary storage.

MODULUS of a counter is the total number of counts or stable states a counter may indicate.

UP COUNTERS count from 0 to a predetermined number.

DOWN COUNTERS count from a predetermined number to 0.

RING COUNTERS are loop counters that may be used for timing operations.

REGISTERS are used as temporary storage devices as well as for transfer of information.

PARALLEL REGISTERS receive or transfer all bits of data simultaneously.

SHIFT REGISTERS are used to perform serial-to-parallel and parallel-to-serial conversion and for scaling binary numbers.

SERIAL TRANSFER causes all bits of data to be transferred on a single line.

PARALLEL TRANSFER has each data bit on its own line.

SCALING of binary numbers means to increase or decrease the magnitude of a number by a power of 2.

LOGIC FAMILIES are composed of logic circuits based on particular types of elements.

ANSWERS TO QUESTIONS Q1. THROUGH Q64.

A1. ⊕.

A2. Low (0).

A3. One or the other of the inputs must be HIGH, but not both at the same time.

A4. Exclusive NOR (X-NOR).

A5. HIGH.

A6. The half adder generates a carry.

A7. Quarter adder.

A8. Sum equals 0 with a carry of 1.

A9. Full adder.

A10. Four.

A11. $S_1 = 1$, $S_2 = 0$ and $C_2 = 1$.

A12. $C_1 = 0$.

A13. X-OR gates.

A14. Four.

A15. MSD of the sum.

A16. Add 1 portion.

A17. Subtrahend.

A18. Storing information.

A19. Six.

A20. 1 and 0, or opposite states.

A21. By cross-coupling NAND or OR gates.

A22. One.

A23. To divide the input by 2.

A24. Clock and data.

A25. Up to one clock pulse.

A26. A positive-going clock pulse.

A27. J-K flip-flop.

A28. Set, or HIGH (1).

A29. When the clock pulse goes LOW.

A30. Both J and K must be HIGH.

A31. Clear (CLR) and preset (PS or PR).

A32. The flip-flop is jammed.

A33. A timing signal.

A34. An astable or free-running multivibrator.

A35. Triggers.

A36. A multiphase clock.

A37. 32.

A38. Ripple.

A39. Toggle.

A40. Synchronous.

A41. The AND gate.

A42. 1111_2, or 15_{10}.

A43. Four.

A44. FFs 2 and 4.

A45. Two.

A46. Three.

A47. The input, or clock pulse.

A48. One.

A49. Four.

A50. $\overline{Q}$ output of FF 1 going LOW.

A51. 16.

A52. Parallel.

A53. By clearing the register.

A54. Shift register.

A55. Serial.

A56. Requires more circuitry.

A57. Eight.

A58. Four.

A59. 2^2, or four times.

A60. Three to the left.

A61. Logic families.

A62. DTL (diode transistor logic).

A63. DIPs (dual inline packages).

A64. Compatible.