signal was perfect, there was a deformation of the waveform. The difference in wave height and overshoot is actually caused by the counter-electromotive force from the speaker, the result of greater resistance and weaker damping of the long cable. Fundamentally, even very low distortion, which cannot be measured, can be detected by our sense of hearing."

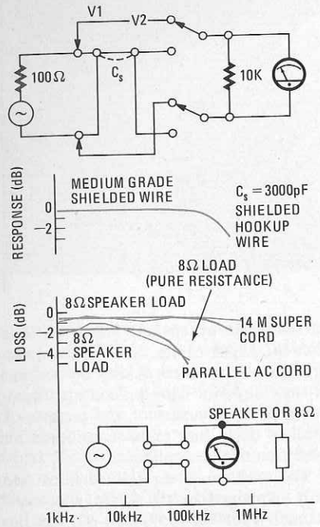
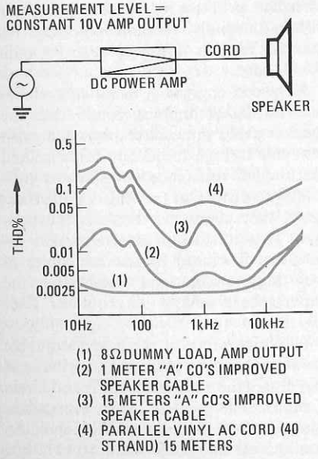


FIG. 2—SPEAKER-CABLE IMPEDANCE has a significant effect on high-frequency response.



(1) 8Ω DUMMY LOAD, AMP OUTPUT
(2) 1 METER "A" CO'S IMPROVED SPEAKER CABLE
(3) 15 METERS "A" CO'S IMPROVED SPEAKER CABLE
(4) PARALLEL VINYL AC CORD (40 STRAND) 15 METERS

FIG. 3—SPEAKER CABLE CAN CAUSE surprisingly large distortion at the speaker inputs.

The most disturbing finding was that the amplifier should ideally be located much closer to the speakers—this is hardly reassuring for the millions who can't or find it inconvenient to do so. The farther the speaker is from the amplifier the slower the slew rate.

Obviously, to do away with the speaker cable is impractical. Kenwood found however that their specially developed cable (not sold in the U. S.) could be used up to one meter (3.25 feet) from the speaker system, with virtually no effect on tonal quality. On the other hand, the audio cable between power amplifier and control amplifier can be long since it is merely a signal transmission line (see Table 2). I know some people who have suffered from RF problems, which they claimed was only eliminated by using specially constructed (expensive) audio cables like Verion, who might suggest that audio cables are important too. But that's another subject.

TABLE 2—Comparison of Audio Cable and Speaker Cord

	Audio cable	Speaker cord
Role	Voltage transmission	Power transmission
Transmitting impedance	10 ohms ~ 1000 ohms	Almost zero
Load conditions	20 ohms ~ 100,000 ohms. Changes somewhat with frequency and is constant without regard to signal level; slight input capacitance only; no reactance.	Indication 4~14 ohms. Changes considerably with frequency; changes with signal level; reactance component is large and complex; counterelectromotive force produced by speaker.
Effect on performance	Small	Large
Effect on tonal quality	Small	Large
Change with length	Small	Large
Extraneous induction	Easy	Difficult
Effect of cord characteristics	Since the load conditions other than component C are large, effect is small.	Components L, C and R have a large effect.

Note: In the past the quality of the system was not improved while pursuing the characteristics on the amplifier side because the dynamic characteristics with the speaker connected such as these were not considered. As shown in Fig. 7, the deterioration in distortion when the speaker is connected exceeded our imagination.

The Kenwood solution is obviously directed at the perfectionist audiophile (assuming you agree with their analysis of the problem). There are alternative solutions if you agree that the problem of transmission losses does exist and *can be heard*. If you don't like the price of the super cables, for less cost you can use the heaviest zip cord you can find, if your amplifier is not bothered by capacitance problems. Don't be afraid of reducing the size of the wire at the speaker terminals (by no more than half) so it will fit into spring-loaded speaker or amplifier terminals. But be careful, since the total resistance of the length of the wire must be considered.

If you believe that there is a problem of self-inductance, you'll usually find that the super cables are sold on a money-back basis. So if you don't hear a difference you can return them. Most of the special cables are sold by specific lengths and several have special tip ends that allow you to make excellent uniform connections.

For example, Disc Washers' *Smog Lifters*, shown in Fig. 4, have special plastic Y-finished tip ends that resist potential shorting, these cables sell for $1.40-per-foot.

The M & K *Mogami* cable (sold for $1.50-per-foot) must be debraided, and should not be tinned if splicing is necessary. (See Fig. 5.) A firm connection between the wire ends to be joined should be made before soldering. You are also warned to avoid speaker switches,

continued on page 93

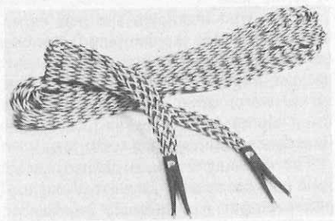


FIG. 4—DISC WASHERS' *Smog Lifter* cables have special plastic tips.

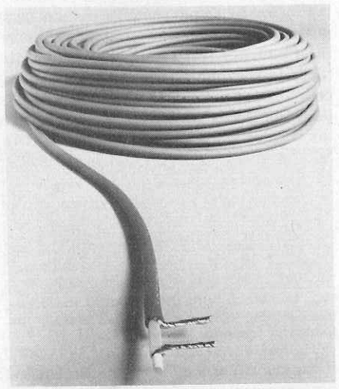


FIG. 5—M & K *MOGAMI* cable must be debraided; no tinning before splicing.

How To Design Digital Circuits

Part 1—With digital circuitry becoming an increasingly important factor in our everyday lives, it's time that we learn how to design logic circuits. Get in on the start of this series as the author discusses digital logic design—beginning with Boolean algebra and Karnaugh maps.

JERRY WOOLSEY

TODAY'S ELECTRONICS HOBBYIST HAS available to him a previously undreamed-of assortment of hardware for his projects. Whereas 15 or 20 years ago electronics magazines ran construction articles on simple two- or three-tube circuits, using point-to-point wiring, the projects of today consist of computer CPU boards and computer terminals on complicated double-sided PC boards. Digital circuits are now appearing in almost everything electronic, including "linear" applications such as tuners, TV sets and synthesizers.

To enjoy fully the electronic technology of today, a hobbyist needs to know not only how to bias transistors and match impedances, but also how to analyze and design digital circuits. Although most experimenters can do this using brute-force methods, *there are some fairly simple methods for reducing the number of gates in*, and hence the complexity of, a digital circuit.

Digital electronics is the realization of Boolean algebra, and some knowledge of it is required to design a digital circuit. Since the subject of Boolean algebra has been covered in magazine articles as well as in many textbooks, it is assumed the reader has a fair knowledge of it, and is able to write his desired function in both equation and truth-table form. In this article, we will see how to apply the fundamentals of Boolean algebra to construct both parallel and series circuits from a truth table, and then *reduce the gates to the minimum needed*. Throughout this article, the AND function will be implied between two variables if no operator is given between them, i.e., $x \cdot y$ will be written simply as xy.

Combinational switching circuits

A *combinational switching circuit* is a digital circuit whose output at any time is dependent only on its input at that time, regardless of any previous input or output. Thus, no "memory" circuits are included. (A flip-flop is considered a memory circuit.) The first part of this article is concerned only with these circuits.

A Boolean equation, no matter what form, can always be reduced or expanded to give an equation in either a sum-of-products (S-P) or product-of-sums (P-S) form. In the S-P form, the equation is an OR (sum) of several AND (product) groups. In the P-S form, the equation is an AND of several OR groups. As an example, take the following equation:

$$a = x(y + z) + y \qquad 1)$$

This can be expanded by multiplying through the x to get

$$a = xy + xz + y \text{ (S-P)} \qquad 2)$$

which is in S-P form, or may also be written in P-S form as

$$a = (x + y + z)(x + y + \bar{z})(\bar{x} + y + z) \text{ (P-S)} \qquad 3)$$

A primitive implementation of equation 1 is shown in Fig. 1, using the S-P form of the equation. So we pick up the IC data book and notice one peculiar thing: almost all the gates available are NAND, with several AND, but few OR and NOR types. Why should this be so when most functions are written as strictly AND and OR? To understand why, we can apply DeMorgan's theorem to equation 2. This theorem states that if we invert the individual members of one side of an equation, then change the signs between members from AND to OR and vice-versa, then invert the entire side of the equation, the equation is still true. To illustrate, let's apply this theorem to equation 2. First, we invert the individual members to obtain

$$a = \overline{xy} + \overline{xz} + \bar{y}$$

Now we change the signs between members and get

$$a = \overline{xy} \cdot \overline{xz} \cdot \bar{y}$$

Finally inverting the entire string, we get

$$a = \overline{\overline{xy} \cdot \overline{xz} \cdot \bar{y}}$$

This says that to obtain the result a, we NAND x and y, NAND x and z, INVERT y, and NAND the three results. Figure 2 shows the logic circuit. Thus, the NAND gates can perform the AND and OR functions. When a group of NAND gates feed another NAND gate, the first gates perform an AND function, and the gate they feed performs the OR function on each AND'ed group. We thus only need to keep a supply of NAND gates to realize any equation in S-P form. In this case, an extra inverter is needed, but inverted variables are often already available from another output, and even if not, this could be performed by a NAND, keeping the three input gates on one package.

It should be noted that the P-S form can be implemented in circuit form by using NOR gates, the first input gates

perform the OR function, and the second set of gates perform the AND function. However, P-S forms can always be expanded to S-P forms, so the remainder of the article will deal only with S-P forms.

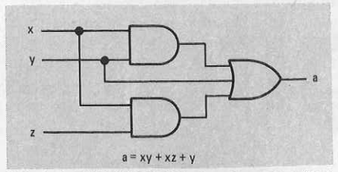


FIG. 1—LOGIC CIRCUIT that performs the Boolean algebra expression shown.

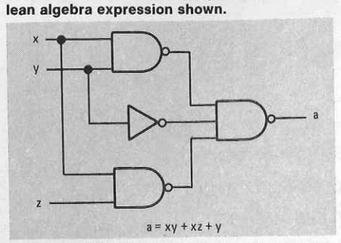


FIG. 2—NAND GATES can be used to perform the same function as shown in Fig. 1.

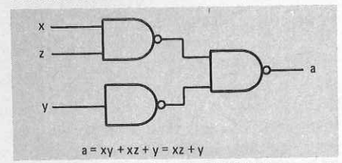


FIG. 3—SIMPLIFIED LOGIC CIRCUIT is equivalent to circuit shown in Fig. 2.

Using the theorems of Boolean algebra, it can be seen that equation 2 can be reduced to:

$$a = xz + y$$

This means that the circuit shown in Fig. 3 is equivalent to the one shown in Fig. 2. Obviously, this is much simpler, and can be done on only one IC.

Reduction of mathematical expressions by Boolean algebra theorems is tedious and often hit-and-miss. To alleviate the problem, we turn to a tool that eliminates much of the work.

The Karnaugh map

The Karnaugh map is simply a rearranged truth table that can readily give valuable information for circuit design and reduction. There are 2^n boxes in the map, where n is the number of inputs to the circuit. Each row and column is numbered in binary, and the value in each box is the output of the circuit when the coordinates of the box are the input. The numbering of the columns starts at the left at zero, and is arranged such that the number of the next column to the right differs in only one bit position. Thus, 00 is followed by 01, which is followed by 11, which is followed by 10. The rows are numbered similarly. Figure 4 illustrates the numbering and the corresponding decimal coordinates of the boxes for functions with two, three and four inputs. Beyond four inputs, the Karnaugh map becomes too cumbersome, so other methods have been designed for these situations.

As an example, suppose a three-input circuit with inputs x, y and z were to produce a logic-1 output when x = 0 and y = z = 1. Then we would enter a 1 into the box numbered 3 in Fig. 4-b (xyz = 011 = 3_{10}). If a zero were to be produced when x = y = 1 and z = 0, the box numbered 6 would contain a zero. In certain cases, such as BCD circuits, some input combinations are meaningless (1010 is not a BCD number). In these instances, we enter a "d" in the appropriate box to indicate a "don't-care" condition. This tells us the output may be either 0 or 1 with the given input, since that input would never occur. This may be used to further aid in circuit reduction.

Figure 5-a shows the truth table for equation 1, and Fig. 5-b shows the Karnaugh map derived from it.

Now we come to the interesting property of the map. By definition of the structure of the map, any two adjacent boxes (horizontally or vertically, but not diagonally) differ in coordinates, i.e., in input conditions, by only one bit. For example, the boxes where (x = 0, y = z = 1) and (x = z = 0, y = 1) are adjacent, and differ in coordinates in only the z-input. This property also holds when "wrapped-around," i.e., the top right-hand box (x = z = 0, y = 1) is adjacent to the top left-hand box (x = y = z = 0), since they differ only in the y-bit of the coordinate. The same holds true for vertical wrap-around. Two of these adjacent boxes are said to form a 1-cube, since there are 2^1 boxes in the cube.

Now refer to Fig. 5-b. If we take two adjacent boxes that contain a 1, for example (x = z = 1, y = 0) and (x = y = z = 1), we find that the output of the circuit must be a 1 whenever x = z = 1, independent of the value of y. That is, whenever xz is true, the equation is true. Similarly, box (x = 0, y = z = 1) and box (x = z = 0, y = 1) are adjacent and contain ones, so we see that the output is true whenever x = 0 and y = 1, independent of z. Thus, $\bar{x}y$ being true will cause the equation to be true, or a 1 to be output. Taking all combinations of two adjacent boxes, both of which contain a 1, the following equation is derived

$$a = xz + \bar{x}y + xy + yz + y\bar{z}$$

which is equivalent to equation 1, but is obviously not reduced. The reason for this is that several conditions for an output have been duplicated by more than one term of the equation. For example, if x = y = z = 1 is entered, the three terms xy, xz and yz will all be true, causing the output to be true, but it is only necessary to have one term true to cause the output to be true. Thus, we have redundant members in the equation. If we take only three adjacent sets of boxes to cover all the 1-outputs, we can obtain the equation

$$a = xz + \bar{x}y + xy$$

Now all the boxes containing a 1 have been covered by at least one of the terms of the equation, which means that the equation is a true representation of the truth table. But the equation is still not completely simplified. If we look at the 1-cube (two adjacent boxes) consisting of (x = 0, y = z = 1) and (x = z = 0, y = 1), and the 1-cube consisting of (x = y = z = 1) and (x = y = 1, z = 0), we see that the output of the function is always 1 whenever y = 1, regardless of the value of x or z. Thus, we have formed a 2-cube (2^2 boxes), and have found that y = 1 satisfies the conditions for generating a logic-1 output for each of the four boxes. Note that, in looking at the coordinates of each of these boxes, y is the only coordinate that does not change, and is always 1. We need now to cover only one more box where a logic-1 output is to be generated, box (x = z = 1, y = 0). To do this, we could simply say we need $x\bar{y}z$ to be true for the output to be a logic 1, but we have another adjacent 1-labeled box (x = y = z = 1) and if we use this to form a 1-cube, that term of the equation reduces to xz, since a 1-output is independent of y if x and z are 1. Thus, we obtain

$$a = y + xz$$

as our final equation, and implement it as shown in Fig. 3.

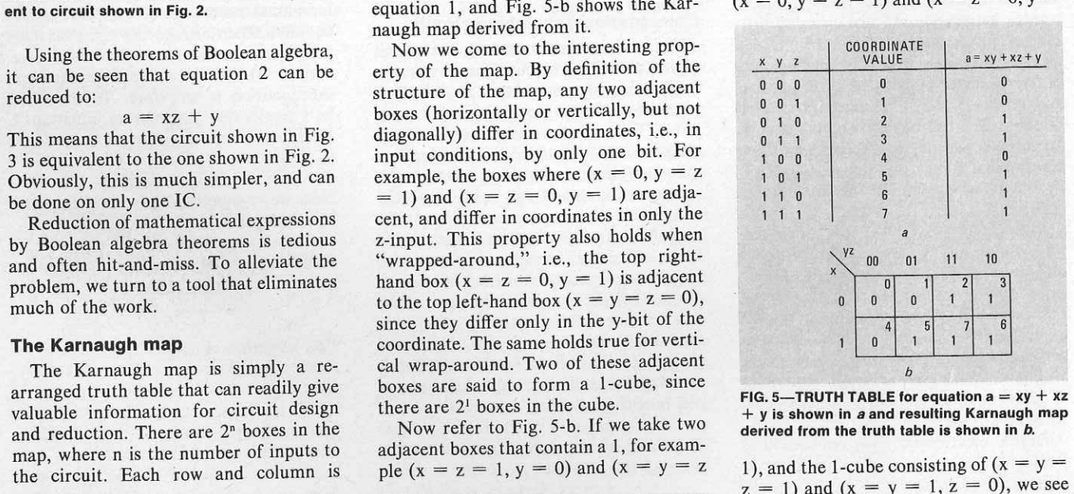


FIG. 4—KARNAUGH MAPS are used to simplify logic circuits. A Karnaugh map for a 2-input circuit is shown in *a*, 3-input circuit is shown in *b* and a 4-input circuit is shown in *c*.

x	y	z	COORDINATE VALUE	a = xy + xz + y
0	0	0	0	0
0	0	1	1	0
0	1	0	2	1
0	1	1	3	1
1	0	0	4	0
1	0	1	5	1
1	1	0	6	1
1	1	1	7	1

a

x \ yz	00	01	11	10
0	0	0	1	1
1	0	1	1	1

b

FIG. 5—TRUTH TABLE for equation a = xy + xz + y is shown in *a* and resulting Karnaugh map derived from the truth table is shown in *b*.

When "d" (don't-care) outputs are specified, these are included as 1-outputs if it enables us to make larger cubes with other 1-outputs, hence simplifying the equation, or as 0-outputs if they are not used in making larger cubes.

Even larger cubes may be found in four-input functions. A 1-cube is two adjacent boxes containing either a 1 or "d"; a 2-cube is two adjacent 1-cubes (i.e., a 2 × 2 box or 4 × 1 horizontal or vertical row); and a 3-cube is two adjacent 2-cubes (i.e., a 4 × 2 horizontal or vertical box). If a map consists only of 1- and d-labeled boxes, the function is always true, or a constant 1.

The step-by-step procedure for circuit reduction, then, is as follows:

1) Draw the truth table, and fill in the boxes of the Karnaugh map with a 1, 0 or d, using the inputs as coordinates and the outputs as box entries. For example, see Figs. 6-a and 6-b.
2) Examine the map for any 3-cubes, i.e., a 4 × 2 box containing no zeroes. Don't forget to check for possible wrap-around. In Fig. 6, a 3-cube is formed by decimal coordinate boxes 0, 1, 2, 3, 4, 5, 6 and 7 (see Fig. 4-c). The coordinates abcd of these boxes are examined, and it is found that b, c and d take on all values (0 and 1) while a is always 0. Thus, when a = 0, the output is always 1 independent of b, c and d, so one term of the final equation is simply $\bar{a}$. Place a check in each of the boxes of this 3-cube to indicate that they have been covered. When the coordinates of any of these boxes are input, the output will be 1 simply because $\bar{a}$ is 1. The map now appears as in Fig. 7-a.

a	b	c	d	OUTPUT	DECIMAL
0	0	0	0	1	0
0	0	0	1	1	1
0	0	1	0	d	2
0	0	1	1	1	3
0	1	0	0	d	4
0	1	0	1	1	5
0	1	1	0	1	6
0	1	1	1	d	7
1	0	0	0	1	8
1	0	0	1	0	9
1	0	1	0	1	10
1	0	1	1	0	11
1	1	0	0	0	12
1	1	0	1	d	13
1	1	1	0	0	14
1	1	1	1	1	15

a

ab \ cd	00	01	11	10
00	1	1	1	d
01	d	1	d	1
11	0	d	1	0
10	1	0	0	1

b

FIG. 6—TO SIMPLIFY a logic circuit, first draw a truth table that represents the circuit function as shown in *a*. Next, a Karnaugh map is derived from the truth table as shown in *b*.

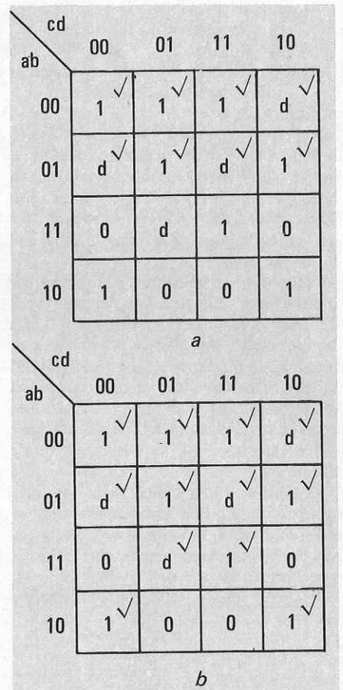


FIG. 7—KARNAUGH MAPS are reduced by first looking for a 3-cube (a 4 × 2 box containing no zeroes). When a 3-cube is found, check the individual boxes as shown in *a*. Next, look for 2-cubes and check these boxes as shown in *b*.

3) Examine the map for any 2-cubes, that is, any 2 × 2 or 4 × 1 box containing no zeroes, and at least one 1-labeled box not yet checked. In our example, decimal boxes 5, 7, 13 and 15 form such a cube. Examining the binary coordinates of these boxes, we find that b and d are always 1, while a and c may take on any value. Thus, our second term of the equation is bd, giving f = $\bar{a}$ + bd so far. Check off the boxes covered by the second term. Check for more 2-cubes. In the example, we have another 2-cube that is not so obvious, due to wrap-around. This is the cube containing decimal boxes 0, 2, 8 and 10. From the binary coordinates, we find b and d are always 0, while a and c can take on any value. Thus our next term for an output of 1 is $\bar{b}\,\bar{d}$ = 1, and our function is now f = $\bar{a}$ + bd + $\bar{b}\,\bar{d}$. Check off the boxes covered. The map now looks like Fig. 7-b.
4) Examine the map for any 1-cubes, i.e., two adjacent boxes each containing a 1 or a d that also contains one 1-labeled box not checked. Write down the nonvarying coordinates as a term of the function, and check for more 1-cubes. No 1-cubes remain in the example.
5) If any 1-labeled boxes remain unchecked, write down their coordinates as a term of the function. For example, if the box with coordinates (a = 0, b = c = d = 1) contained a 1 but was not yet checked off, we would write the coordinates as $\bar{a}$bcd and insert it as a term in the equation. At the completion of this step, all boxes with a 1 in them should be checked off.
6) By inspection, make sure no cube is completely covered by other cubes. Each cube, no matter what size, must contain at least one 1-labeled box not contained in any other cube. If it does not, discard its corresponding term from the equation.
7) OR all the terms derived above to get the final reduced function. In our example we get:
$$f = \bar{a} + bd + \bar{b}\,\bar{d}$$
8) Feed each term into a NAND gate, and feed the outputs of the NAND gates to another NAND gate. The circuit is complete. See Fig. 8-a.

In searching for cubes to cover an unchecked 1-labeled box, the largest possible cube should be chosen, even if it covers other boxes already checked, so that the number of inputs to each gate is minimized.

Note that Fig. 8-a has $\bar{a}$ as an input that simply gets inverted before going to the output gate. Instead of this, it would be simpler to feed a directly to the output gate, as in Fig. 8-b. Also note that the

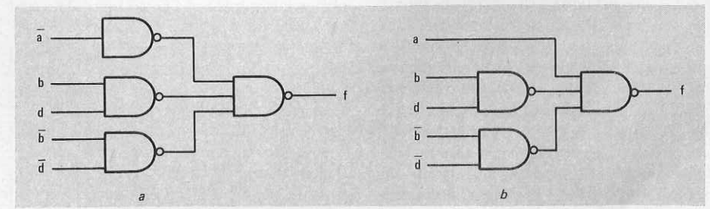


FIG. 8—LOGIC CIRCUIT is derived from reduced Karnaugh map and is shown in *a*. Inverter can be eliminated as shown in *b*.

output of the gate fed by bd cannot be used as input $\bar{b}\,\bar{d}$ to the gate below it since $\bar{b}\,\bar{d} \neq \overline{bd}$.

As was noted earlier, the Karnaugh map method is too difficult beyond four inputs. The designer has to start considering mirror-images, and mistakes are easily made. It also does not reduce the circuit fully if multiple outputs are desired, as in a BCD to seven-segment decoder. Fortunately, there is another fairly simple method to use in these cases.

Quine-McCluskey method

The Quine-McCluskey method works on the same principle as the Karnaugh map, but is performed in tabular form. As an example of this method, we will construct a circuit to produce a 1-output, called f, whenever a 2-bit number A, whose bits are designated as a_1 and a_2, is larger than a 2-bit number B, whose bits are b_1 and b_2. The truth table for this function is given in Fig. 9.

a_1	a_2	b_1	b_2	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

FIG. 9—QUINE-McCLUSKEY METHOD is used when a circuit with four or more inputs must be designed. First a truth table describing the circuit function is generated as shown.

NO. OF 1-BITS	INPUT (DECIMAL EQUIVALENT OF $a_1\,a_2\,b_1\,b_2$)
1	4 8
2	9 12
3	13 14

FIG. 10—ADJACENT BOXES and cubes are determined in a table generated from the truth table shown in Fig. 9.

The Karnaugh map obviated adjacent boxes and cubes. In the Q-M method, we write down a table to help show adjacency (see Fig. 10). The decimal value of each set of inputs that will generate either a 1- or don't-care output is listed in ascending order in groups according to the number of 1-bits in the input. For example, A = a_1a_2 = 10 and B = b_1b_2 = 01 produces a 1-output, and the number of 1-bits in A and B is two, so a 9 ($a_1a_2b_1b_2$ = 1001) is placed in the 2-bits group. Within each group, the decimal inputs are listed in ascending order. It can now be seen that two inputs producing a 1-output are adjacent, in the same sense as in the Karnaugh map, if three conditions are met:

1) The number of 1-bits of each input differs by exactly one.
2) The decimal input with the smaller number of 1-bits must be smaller than the input with the larger number of 1-bits.
3) The difference of the two decimal inputs must be a power of two.

According to condition 1, inputs listed in the 3-bit group can only be adjacent to inputs in the 2-bit or 4-bit groups; inputs in the 1-bit group can only be adjacent to inputs in the 0-bit or 2-bit groups, etc. This is consistent with the definition of adjacency being a difference in only one bit of the input.

According to condition 2, the decimal input 4 may be adjacent to decimal input 12, since 4, having fewer 1-bits than 12, is smaller than 12. If, for example, the decimal input 3 produced a 1-output, it would be placed in the 2-bits group, but could not be adjacent to 4, since it has more 1-bits but is less than 4. This would cause more than one bit in the input to be different.

Condition 3 is obvious, since only one input bit may differ for adjacency. If the difference of the two numbers is not a power of two, more than one bit differs.

NO. OF 1-BITS	INPUT	1-CUBES
1	4 ✓ 8 ✓	4,12 (8) 8,9 (1) 8,12 (4)
2	9 ✓ 12 ✓	9,13 (4) 12,13 (1) 12,14 (2)
3	13 ✓ 14 ✓	

FIG. 11—SIMPLIFICATION starts by listing the 1-cubes.

We now use these rules to make a third column, consisting of a list of adjacent boxes, or 1-cubes. We take the first input number in the table, 4, and check it for adjacency with the entries in the next bit group. Box 4 is not adjacent to 9, since the difference, 5, is not a power of two. Box 4 is, however, adjacent to 12, since the difference is 8, and 4 is less than 12. Thus, we enter into the third column the numbers 4 and 12 together, with their difference in parentheses (see Fig. 11). Since the inputs 4 and 12 have been covered by a higher cube, we place a check next to them in the column labeled input.

The input 4 cannot be adjacent to any other bit group, so we look at input 8. Box 8 is adjacent to 9, since the difference is a power of two, so we enter the numbers and difference as a 1-cube and place a check next to the 8 and 9 inputs to indicate they have been covered by a higher cube. Box 8 is also adjacent to box 12, so we repeat the process for them. Since we are through checking the 1-bit group, we place a line under 8,12(4) in the 1-cube column and start checking the 2-bit group. Box 9 is adjacent to box 13 but not to box 14. Box 12 is also adjacent to box 13, as well as 14, and we are finished creating 1-cubes. The 1-cube column now contains a list of all the possible 1-cubes that could be extracted from the Karnaugh map. Since all the inputs in the input column have been checked off, they are all contained in higher cubes.

We now have two groups of 1-cubes, and use these to form 2-cubes. The same conditions hold for forming adjacent cubes, except now the numbers in parentheses must also match. Looking at the first 1-cube entry, 4,12(8), we see that it is not adjacent to any 1-cube in the second group, since none have an 8 in parentheses. Going to 8,9(1), we find an entry, 12,13(1) in the second group that has the same number in parentheses. Since the difference of 8 and 12 (or 9 and 13) is also a power of two, and 8 is less than 9 and in a lower group, we enter this in the next column as a 2-cube, and indicate both the first and second differences in parentheses. The two entries that formed this cube are checked, since they are covered by the higher cube (see Fig. 12).

Another entry, 8,12(4), is adjacent to the entry 9,13(4), so it is entered as a 2-cube and the separate 1-cubes are checked. However, this is identical to the previous 2-cube and is thus stricken. No further adjacency is found, and there are no more groups to check for adjacency, so the checking of the 1-cube column is complete.

We now go to the next column and continue until no adjacencies are found. The same rules are followed in each column, checking each entry against each

continued on page 92

NO. OF 1-BITS	INPUT	1-CUBES	2-CUBES
1	4 ✓ 8 ✓	4,12 (8) 8,9 (1) ✓ 8,12 (4) ✓	8,9,12,13 (1,4)
2	9 ✓ 12 ✓	9,13 (4) ✓ 12,13 (1) ✓ 12,14 (2)	
3	13 ✓ 14 ✓		

FIG. 12—ALL 2-CUBES are listed. Second 2-cube is crossed off since it is covered by first entry.

BUILD THIS

REMOTE TELEPHONE EAR–

Listen via Long Distance

This device—the fourth in a series of phone gadgets—lets you monitor sounds in your home or office when you call your telephone from a remote location.

JULES H. GILDER

IN THE APRIL AND MAY 1977 AND MAY 1978 issues, we showed how to construct add-on telephone accessories that let you turn on and turn off various household appliances by remote control, build a hands-off telephone amplifier and assemble an autodialer and cassette interface that dialed authorities or neighbors in case of a fire or intruders in your home.

If you were interested in these items, you'll flip over the Remote Ear that lets you dial your home phone and then listen for the sound of running water or a radio that was inadvertently left on. Or maybe you just want to check your house and see that everything is quiet and no one has broken into it.

The Remote Ear is an adaptation of the Teleswitch circuit (April 1977). It automatically connects a microphone and amplifier to the telephone so that you can monitor a remote location. As you will quickly see from the schematic, the Remote Ear uses the same type of signal detectors as the Teleswitch. However, instead of having controlled outlets to turn devices on and off, the Remote Ear has a small three-transistor amplifier connected to it.

This amplifier is identical to the one described in the Speakerphone circuit (May 1977). Its signal is very clear and audible. The output of the amplifier, which is located in the area that you want to monitor, is fed to a small speaker that is acoustically coupled to the telephone mouthpiece.

Since it is unlikely that remote listening will be done for long periods of time, the Remote Ear has a built-in timer that allows you to listen for about three minutes. Longer or shorter listening times can be set by adjusting the timing resistor in the emitter circuit of unijunction transistor Q2.

About the circuit

Sound switch 1 and the unijunction timer that is associated with it (Q1) are the same as were used in the Teleswitch. Sound switch 2, however, is slightly modified. Instead of having one relay connected to the 2N3904 collector, there are two relays, RY2 and RY4.

The operation of the Remote Ear involves several steps. When the telephone rings the first time, sound switch 1 triggers and causes RY1 to close. This applies power to sound switch 2 and to the two timing circuits consisting of unijunction transistors Q1 and Q2.

If the phone rings more than once, within 20 seconds sound switch 2 triggers and RY 2 closes. Contact RY2-1 disconnects the power from the first unijunction transistor timing circuit and from sound switch 1. This prevents the Remote Ear from being activated and makes it necessary to wait three minutes before the next attempt.

If, however, the phone rings only once, there is enough time for a charge to build up on C1 and for Q1 to trigger, activating RY3. When RY3 is activated it switches the RY2 coil out of the control circuit of sound switch 2 and replaces it with the RY4 coil.

The first time the telephone rings only once it arms the circuit. The next time the telephone rings it turns on the listening circuitry. This is done by sound switch 2 activating RY4, which, in turn, controls the amplifier and the answering solenoid.

Relay RY4 latches closed and is held in that position until a reset pulse from unijunction timer Q2 turns off the 2N3904 controlling RY2 and RY4 and unlatches SCR2.

The telephone is actually answered by a solenoid that pulls up when RY4 closes. This releases the cradle switch and answers the phone. The handset of the telephone is placed on the table alongside the telephone. The loudspeaker connected to the output of the amplifier is held next to the mouthpiece (rubber bands can be used). Thus, the sound picked up by the crystal microphone is amplified and acoustically coupled to the telephone.

After three minutes, or whatever time period you selected has elapsed, a reset pulse is generated and the bases of the control 2N3904's are brought to ground potential, turning these transistors off and unlatching the SCR's. The unit is now ready for its next monitoring period.

Construction

This project is constructed from four modular circuits. The first two circuits are sound switches identical to those built in the Teleswitch (April 1977). After the sound switches are built, they should be mounted in a metal chassis that is large enough to be placed under the telephone. A 5 × 9 × 2-inch aluminum chassis was used for the prototype. A 1/8-inch hole should be drilled where each of the crystal microphones is mounted so that sound will reach them more easily.

After the sound switch modules are mounted, assemble the control module using the circuit shown in the schematic. The circuit can be fabricated by wiring