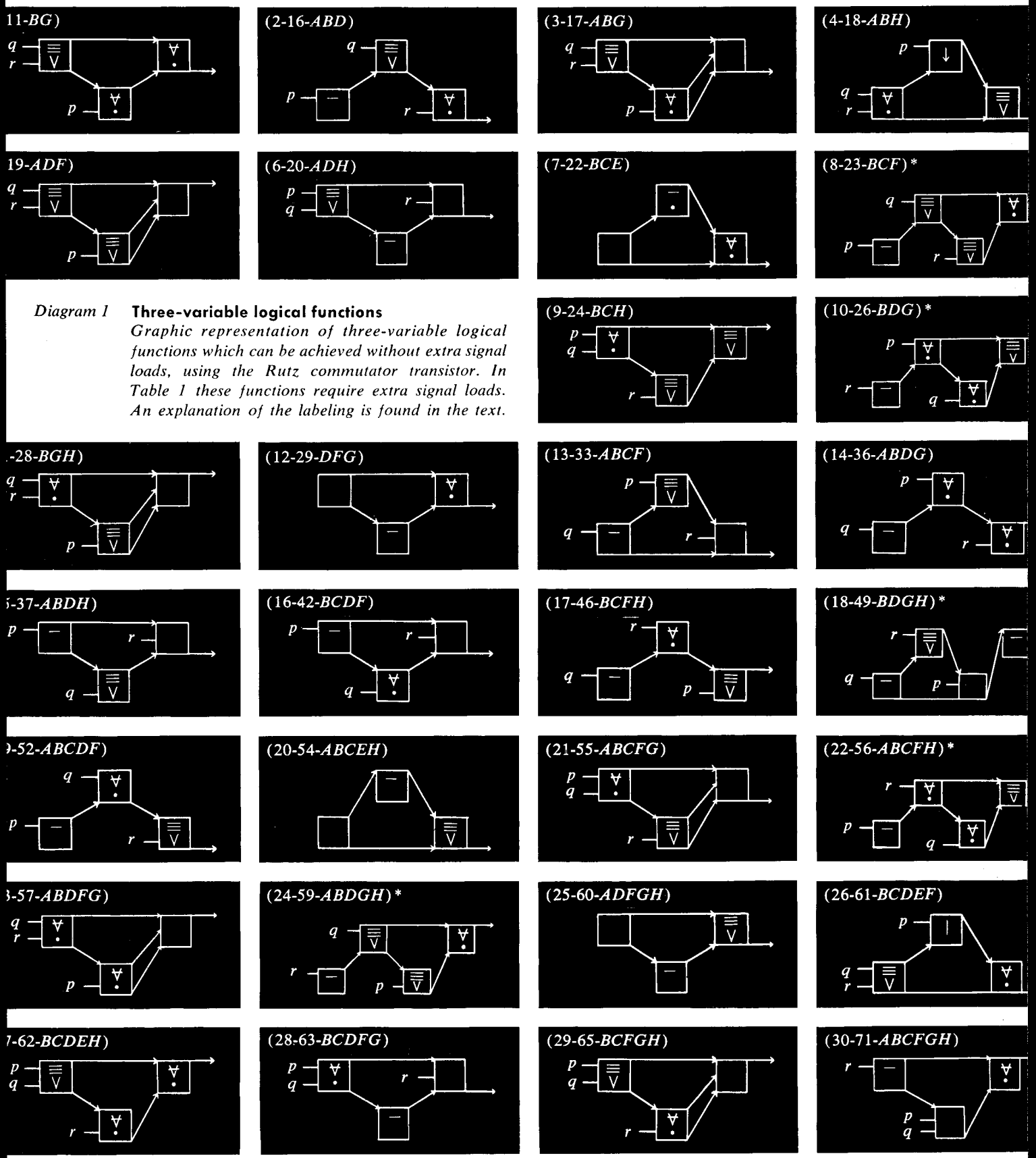


Diagram 1 **Three-variable logical functions**
Graphic representation of three-variable logical functions which can be achieved without extra signal loads, using the Rutz commutator transistor. In Table 1 these functions require extra signal loads. An explanation of the labeling is found in the text.

B. Dunham

The Multipurpose Bias Device*
Part I
The Commutator Transistor

An article on the design and development of the Rutz commutator transistor will be included in a subsequent issue of the IBM Journal of Research and Development.

Abstract: It is suggested that multipurpose devices will provide economy in both number and assortment of basic computer building blocks. A study is made of the Rutz commutator transistor largely in application to three-input, one-output logical situations. A basis is thereby provided for a more general analysis to be published at a later date.

General introduction

In 1938, Shannon indicated a correlation of truth-functional expressions with electrical circuits. Since that time much effort has been made to devise general mechanical procedures for simplifying logical formulas. One major difficulty, however, is the limited application of proposed solutions. Techniques which minimize expressions formed from one set of functions may not prove relevant if another set is assumed. As a result, the primary logical problem is to decide which groups of truth-functional elements are desired rather than to formulate rules of simplification for a given set.

The correspondent practical concern is to determine which types of physical device, correlative with logical particles, can be expected to serve as efficient building blocks for machines. Material considerations are especially pertinent. It is, for example, quite advantageous to diminish the number of devices used. On the other hand, it is important to minimize the kinds of machine element required.

If many different classes of devices are available, each type accomplishing a different truth-functional connective, the number of devices requisite for a given situation should prove relatively small. If, however, the variety of machine elements is so limited that only a few logical particles are achieved directly, the number of devices needed may prove unduly large.

The alternative we wish to suggest is that multipurpose devices be used. A *multipurpose* logic device is

*Portions of this paper were presented to the International Symposium on Theory of Switching at Harvard University, April 4, 1957.

one which can be adjusted to realize directly a diversity of truth-functional elements. Economy in both number and assortment of basic machine building blocks might then be obtained. Ease of adjustment is, of course, prerequisite.

A *bias* device is one which achieves its logical functions by on-and-off biasing of inputs. An input is said to be *biased* if it is so fixed as to be always on or always off. Since the latter method of adaptation is both direct and flexible, multipurpose bias devices are of much interest.

A full adder can be regarded as a multipurpose bias device. Consider the following table:

p	q	r	SUM	CARRY
1	1	1	1	1
0	1	1	0	1
1	0	1	0	1
0	0	1	1	0
1	1	0	0	1
0	1	0	1	0
1	0	0	1	0
0	0	0	0	0

'p', 'q', and 'r' may be thought of as denoting the three variable inputs. 'SUM' and 'CARRY' indicate the sum and carry respectively. The eight possible input states are shown symbolically by the three columns of 1's and 0's on the left. For example, the top horizontal row signifies that all three inputs are *on,* since 1 is assigned in every case. The next row down signifies that the first input is *off* and the other two on, since 0 is assigned to 'p', and 1 to 'q' and 'r'. The columns under 'SUM' and 'CARRY', which indicate the resultant outputs, are functions of the three variable inputs. As the table shows, 1 is assigned to 'SUM' if and only if one or three of the variable inputs are on. If exactly two or three of the variable inputs are on, 1 is assigned to 'CARRY'.

It should be noted that the logic of the present discussion is *2-valued,* in that 1 and 0 are the only two values assigned throughout. Such terms as *singulary, binary, ternary, quaternary,* on the other hand, are used here to denote the number of variable inputs.

Let us suppose the input represented by 'r' is so adjusted as to be always off. Then 'p' and 'q' indicate the only variable inputs. Under such conditions, as the bottom half of the table shows, 'SUM' is assigned 1 if and only if 1 is assigned to either 'p' or 'q', but not both. The binary exclusive 'or' (symbolized '$\forall$') is thus achieved. Further, as the table records, 'CARRY' is assigned 1 just in that single case where 1 as assigned to both 'p' and 'q'. The connective 'and' (symbolized '$\cdot$') is therefore realized.

Suppose, however, the input represented by 'r' is so regulated that it is always on. The top half of the table reveals that 1 is assigned to 'SUM' just in those cases where both 'p' and 'q' receive like assignments of 1 or 0. The binary function 'if-and-only-if' (symbolized '$\equiv$') is thus accomplished. On the other hand, 'CARRY' is assigned 0 just in that case where both 'p' and 'q' are assigned 0. The inclusive 'or' (symbolized '$\vee$') is therefore realized.

The singulary operator 'not' (symbolized '$-$') is obtained by biasing two of the inputs. If constant 0's are assigned to 'q' and constant 1's to 'r', 'p' will represent the only variable input. Under such conditions, as the table indicates in the third and fourth rows down, 'p' and 'SUM' will in every case receive opposite assignments of 1 and 0.

There is considerable advantage in generating '$\forall$' and '$\equiv$'. The two connectives can in many cases be substituted into an already minimized expression assembled logically from '$-$', '$\cdot$', and '$\vee$' in such a way as to reduce the number of full binary functions required; a *full* binary connective is one which cannot be reduced to a singulary function. In fact, '$\forall$' and '$\equiv$' are the only two of all the possible binary connectives which can be so used. The expression '$-(p \cdot q) \cdot (p \vee q)$', for example, can be written simply as '$p \forall q$'; and the expression '$(p \cdot q) \vee -(p \vee q)$', as '$p \equiv q$'.

An appropriate interchange of '$\forall$' and '$\equiv$' will also enable many denial signs to be removed. Any expression or part of an expression in which '$\forall$' and '$\equiv$' are the only non-singulary functions found is equivalent to some expression of equal length from which '$-$' has been eliminated. '$(\bar{p} \forall q) \equiv r$', for example, is equivalent to '$(p \equiv q) \equiv r$'.

It should further be noted that the ternary connectives 'SUM' and 'CARRY' can each be used to reduce in size many formulas assembled exclusively from singulary and binary truth-functional elements. For example, where 'p', 'q' and 'r' represent the three variable inputs, 'CARRY' is equivalent to the expression '$(q \cdot r) \forall (p \cdot (q \forall r))$'. Consequently, it can be seen that devices which achieve directly functions involving more than two variable inputs are of substantial interest.

R. F. Rutz, of these laboratories, has developed a two-collector transistor which not only operates as a full adder, but can also be adjusted to achieve directly the binary functions 'neither-nor' and 'not-both' (symbolized '$\downarrow$' and '$|$' respectively). '$\downarrow$' is assigned 1 just in that case where both component variables are assigned 0. '$|$' is assigned 0 just in that case where both component variables are assigned 1. Since all of the full binary commutative connectives are obtained ('$\cdot$', '$\vee$', '$\forall$', '$\equiv$', '$\downarrow$', '$|$'), we may suitably describe the device as an *absolute binary commutator* (see Fig. 1, p. 122).

The addition of '$\downarrow$' and '$|$' will in no case permit a reduction in the number of full binary connectives requisite for an already minimized expression. Their use will, of course, in many cases permit a reduction in the number of denial signs needed. The formula '$\bar{p} \cdot (q \vee r)$', for example, is equivalent to the expression '$p \downarrow (q \downarrow r)$'.

The binary commutator is patently a versatile logical

element. Consequently, a careful study of its truth-functional potentialities will do much to indicate the capacity of multipurpose building blocks as such. Also, a basis will be provided for comparison with other devices.

In Part I, which follows immediately, we examine the commutator, largely in application to three-input, one-output situations. We do not consider the transistor's physical performance, since a separate paper will be published by Rutz in a forthcoming issue of this journal. It should be noted, however, that the adaptation to achieve '↓' or '|' involves more than mere biasing of inputs. Load resistances, for example, may be modified.

In Parts II and III, to be published later, we shall present a brief analysis of the relative logical efficiency of different types of many-variable functions. We shall then attempt to formulate a more general philosophy of "multipurpose logic", and to suggest possible lines of future investigation.

Part I The commutator transistor

Application to total three-variable functions

From the schematic letters 'p', 'q', and 'r', an infinite number of different truth-functional formulas can be assembled. These can be broken down, however, into 256 non-overlapping groups of equivalent expressions. A convenient way of labeling these groups, which we shall call *basic* groups, is needed.

Consider the following expression and skeleton truth table:

p	$\forall$	$(r \vee (p \cdot q))$			
1	0	1	1	1	
0	1	1	0	1	*B*
1	0	1	1	0	
0	1	1	0	0	*D*
1	0	0	1	1	
0	0	0	0	1	
1	1	0	1	0	*G*
0	0	0	0	0	

Part of the table is arbitrary and part not. Starting deliberately with an assignment of 1 to each schematic letter occurrence, we have alternated 1 and 0 singly under 'p', in pairs under 'q', and in fours under 'r'. The column under '$\forall$', the main connective, shows that '$p \forall (r \vee (p \cdot q))$', taken as a whole, is assigned 1 in exactly three cases: first, where 0 is assigned to 'p', and 1 to 'q' and 'r'; second, where 0 is assigned to 'p' and 'q' and 1 to 'r'; third, where 1 is assigned to 'p', and 0 to 'q' and 'r'.

The column of letters at the right of the table, that is, 'B', 'D', and 'G', indicates the method of labelling to be used. 'p', 'q', and 'r' can be selected to represent the variables of any three-input situation. 1's and 0's can be assigned to schematic letters in the arbitrary manner just shown. Under such truth-table stipulations, an expression ϕ will be logically equivalent to '$p \forall (r \vee (p \cdot q))$' if and only if ϕ is assigned 1 in just the second, fourth, and seventh rows down. The eight truth-table rows, in order from the top, can aptly be designated by the letters 'A', 'B', 'C', 'D', 'E', 'F', 'G', and 'H'. The label 'BDG', then, will be understood to denote the whole class of expressions which, under the specific conditions, are assigned 1 in the three rows designated by 'B', 'D', and 'G'. In such a manner, 255 of the 256 basic groups can be provided with unique labels. The one remaining group, in which the resultant truth-table columns contain no 1's whatsoever, can be appropriately labelled '$— (ABCDEFGH)$'.

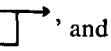
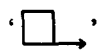
We shall now present as Table 1 an extended list of truth-functional expressions correlative with electrical circuits. As will be evident, no more than three, and in most cases one or two, Rutz binary commutators are needed to handle the purely logical ingredient of any

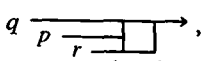
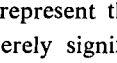
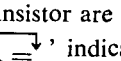
three-input, one-output situation.

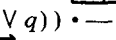
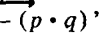
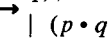
It should be emphasized that no prior inversion of signals is presupposed, since we wish every relevant logical operation to be patent. All 256 cases need not be considered, however. The 256 groups earlier described can be conveniently reclassified as eighty non-overlapping interchange groups. An *interchange group* is a collection of expressions which are said to be interchange equivalent. A formula ϕ is *interchange equivalent* to a formula ψ if and only if either of the following conditions is realized: (1) ϕ is truth-functionally equivalent to ψ; (2) there is a formula ϕ', truth-functionally equivalent to ψ, from which ϕ can be obtained by mere permutation of schematic letters. For example, the expression '$\bar{p} \cdot q \cdot r$' is truth-functionally equivalent to the expression '$p \downarrow (q \mid r)$'. From the latter, '$q \downarrow (p \mid r)$' can be obtained by permuting 'p' and 'q'. Therefore, although not logically equivalent, '$\bar{p} \cdot q \cdot r$' and '$q \downarrow (p \mid r)$' are interchange equivalent.

One addendum is necessary, however. To simplify the discussion, we assume that the formulas in question may contain vacuous equivalent parts. '$p \vee \bar{q}$', for instance, is taken as interchange equivalent to '$p \vee \bar{r}$', since the two can be represented in turn as follows: '$(p \vee \bar{q}) \cdot (r \vee \bar{r})$' and '$(p \vee \bar{r}) \cdot (q \vee \bar{q})$'.

Every interchange group is composed of one, three, or six of the 256 basic groups. Examination of any one of the latter will serve for the total interchange group of which that one is a part, since the results obtained can be adjusted by a straightforward permutation of schematic letters to the other basic groups involved. In each case, however, we shall consider only the *guide basic group*, that is, the basic group having the earliest label in alphabetic order.

In Table 1, the graphic symbols '  ' and '  ' are used in place of 'SUM' and 'CARRY', respectively. Unless otherwise shown, 'p', 'q', and 'r' are understood to denote the three inputs of an unbiased commutator.

A modified notation, such as '  ', stipulates that '$p \cdot q$', 'p', and 'r' represent the three inputs. The inscription '  ' merely signifies that both the sum and carry of a given transistor are utilized. For example, the expression '  ' indicates the application of two commutators. The sum and carry of the first, which is unbiased, serve as the two variable inputs for the second, which is biased to realize '$\equiv$'.

The number of transistors requisite for the logic of the different expressions in Table 1 can easily be determined. One need simply count the occurrences of '$-$', '$\cdot$', '$\vee$', '$\forall$', '$\equiv$', '$\downarrow$', '$\mid$', '  ', '  ', and ''. For example, the expression '$(r \equiv (p \vee q)) \cdot - (p \cdot q)$' would require five transistors; '  $\mid (p \cdot q)$', three. The column headed 'MRCE' (Minimum-Rutz-Commutator-Equivalent) records the smallest number of transistors determined for each guide basic group. It is assumed, of course, that the relatively trivial $-(ABCDEFGH)$ and $ABCDEFGH$ require no commutators.

One further point should be mentioned. The term 'three-input situation' has been understood throughout to denote any situation involving no more than three types of variable input. The same variable input may be used, therefore, a number of times. For example, the expression '$(q \cdot p) \downarrow (q \forall r)$' shows that the input symbolized by 'q' is used twice. Since inputs are also needed for 'p' and 'r', a physical realization of the expression in question would require four signal loads. A *signal load* is merely one use of a variable input. For each guide basic group, the column headed 'SLE' (Signal-Load-Equivalent) records the signal loads necessary for the simplified expression demanding the least number of transistors.

Reduction of signal loads and feedback circuits

As Table 1 reveals, three commutators are required in only twenty-six of the eighty cases. The remaining fifty-four can be achieved with less than three. No tacit inversion of signals has been presupposed, and only one kind of logical device has been assumed. We have not weighed, however, the extralogical equipment needed. It is felt that the latter, by comparison with other approaches, should not prove too considerable, since the purely logical elements involved are so few. In particular, we have not specified how the logic is accomplished in time, and have not indicated possible time delays.

Several pertinent questions, however, still remain. In setting up Table 1, our chief concern was to minimize the transistors necessary for each guide basic group. Little attention was given to signal loads. As the table shows, thirty of the eighty minimal circuits require extra signal loads. In Schematic Diagram 1, therefore, no extra signal loads are permitted; and a re-examination is made of the thirty load-redundant situations.

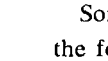
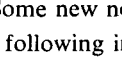
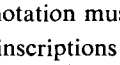
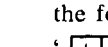
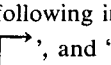
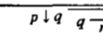
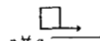
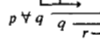
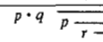
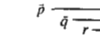
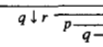
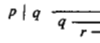
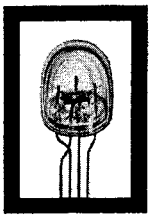
Some new notation must now be introduced. Consider the following inscriptions: '  ', '  ', '  ', '  ', and ''. The symbol '  ', designates a commutator biased to achieve '$\equiv$ and $\vee$'. '$\equiv$' is realized at the output graphically portrayed by the upper horizontal arrow. '$\vee$' is obtained at the other output. Should only one of the two outputs be required, the superfluous arrow need not, of course, be recorded. The

Table 1 **Total Three-Variable Functions**

Guide Basic Group	*First Simplification*	*↓ Simplification*	*| Simplification*	*□ Simplification*	*Interchange Equivalents*	MRCE	SLE
(1) — (*ABCDEFGH*)	$p \cdot \bar{p}$					0	0
(2) *A*	$p \cdot q \cdot r$					2	3
(3) *B*	$\bar{p} \cdot q \cdot r$		$p \downarrow (q \mid r)$		*C, E*	2	3
(4) *D*	$-(p \vee q) \cdot r$	$r \cdot (p \downarrow q)$			*F, G*	2	3
(5) *H*	$-(p \vee q \vee r)$	$p \downarrow (q \vee r)$				2	3
(6) *AB*	$q \cdot r$				*AC, AE*	1	2
(7) *AD*	$r \cdot (p \equiv q)$				*AF, AG*	2	3
(8) *AH*	$(p \equiv q) \cdot (p \equiv r)$					2	3
(9) *BC*	$r \cdot (p \forall q)$				*BE, CE*	2	3
(10) *BD*	$\bar{p} \cdot r$				*BF, CD, CG, EF, EG*	2	2
(11) *BG*	$(p \forall q) \cdot (p \forall r)$				*CF, DE*	2	4
(12) *BH*	$\bar{p} \cdot (q \equiv r)$	$p \downarrow (q \forall r)$			*CH, EH*	2	3
(13) *DF*	$\bar{p} \cdot (q \forall r)$	$p \downarrow (q \equiv r)$			*DG, FG*	2	3
(14) *DH*	$-(p \vee q)$	$p \downarrow q$			*FH, GH*	1	2
(15) *ABC*	$r \cdot (p \vee q)$				*ABE, ACE*	2	3
(16) *ABD*	$r \cdot (\bar{p} \vee q)$				*ABF, ACD, ACG, AEF, AEG*	2	4
(17) *ABG*	$(p \vee q) \cdot (q \equiv r)$				*ACF, ADE*	3	4
(18) *ABH*	$r \equiv (q \vee (p \cdot \bar{r}))$				*ACH, AEH*	3	4
(19) *ADF*	$(p \equiv (q \cdot r)) \cdot (q \vee r)$				*ADG, AFG*	3	5
(20) *ADH*	$p \equiv (q \cdot (\bar{p} \vee r))$				*AFH, AGH*	3	4
(21) *BCD*	$r \cdot -(p \cdot q)$		$r \cdot (p \mid q)$		*BEF, CEG*	2	3
(22) *BCE*	$(p \forall (q \cdot r)) \cdot (q \vee r)$					3	5
(23) *BCF*	$q \forall (p \cdot (q \vee r))$				*BCG, BDE, BEG, CDE, CEF*	3	4
(24) *BCH*	$(r \equiv (p \vee q)) \cdot -(p \cdot q)$	$(r \forall (p \vee q)) \downarrow (p \cdot q)$			*BEH, CEH*	3	5
(25) *BDF*	$\bar{p} \cdot (q \vee r)$	$p \downarrow (q \downarrow r)$			*CDG, EFG*	2	3
(26) *BDG*	$p \forall (r \vee (p \cdot q))$				*BFG, CDF, CFG, DEF, DEG*	3	4
(27) *BDH*	$\bar{p} \cdot (\bar{q} \vee r)$	$p \downarrow (q \cdot \bar{r})$			*BFH, CDH, CGH, EFH, EGH*	3	3
(28) *BGH*	$-(q \cdot p) \cdot (q \equiv r)$	$(q \cdot p) \downarrow (q \vee r)$			*CFH, DEH*	3	4
(29) *DFG*	$(r \forall (p \vee q)) \cdot -(p \cdot q)$	$(r \equiv (p \vee q)) \downarrow (p \cdot q)$				3	5
(30) *DFH*	$-(p \vee (q \cdot r))$	$p \downarrow (q \cdot r)$			*DGH, FGH*	2	3
(31) *ABCD*	r				*ABEF, ACEG*	0	1
(32) *ABCE*	$(q \cdot r) \forall (p \cdot (q \forall r))$					1	3
(33) *ABCF*	$r \equiv (p \vee (q \equiv r))$				*ABCG, ABDE, ABEG, ACDE, ACEF*	2	4
(34) *ABCH*	$r \equiv (p \vee q)$				*ABEH, ACEH*	2	3
(35) *ABDF*	$(q \cdot r) \equiv p \vee (q \equiv r))$				*ACDG, AEFG*	2	3
(36) *ABDG*	$r \forall (p \cdot \bar{q})$				*ABFG, ACDF, ACFG, ADEF, ADEG*	2	4
(37) *ABDH*	$(q \vee p) \equiv (q \cdot r)$				*ABFH, ACDH, ACGH, AEFH, AEGH*	3	4
(38) *ABGH*	$q \equiv r$				*ACFH, ADEH*	1	2
(39) *ADFG*	$p \forall q \forall r$					1	3
(40) *ADFH*	$p \equiv (q \cdot r)$				*ADGH, AFGH*	2	3
(41) *BCDE*	$r \forall (p \cdot q)$				*BCEF, BCEG*	2	3
(42) *BCDF*	$(q \vee r) \forall (p \cdot q)$				*BCDG, BDEF, BEFG, CDEG, CEFG*	3	4
(43) *BCDH*	$(p \cdot q) \forall (r \vee (p \equiv q))$				*BEFH, CEGH*	3	3
(44) *BCEH*	$(p \forall q) \equiv r$					2	3
(45) *BCFG*	$p \forall q$				*BDEG, CDEF*	1	2
(46) *BCFH*	$p \equiv (\bar{q} \cdot r)$				*BCGH, BDEH, BEGH, CDEH, CEFH*	2	4
(47) *BDFG*	$p \forall (q \vee r)$				*CDFG, DEFG*	2	3
(48) *BDFH*	$\bar{p}$				*CDGH, EFGH*	1	1
(49) *BDGH*	$p \forall (r \vee (p \equiv q))$				*BFGH, CDFH, CFGH, DEFH, DEGH*	3	4
(50) *DFGH*	$(q \cdot r) \equiv (p \cdot (q \forall r))$			— (□)		2	3
(51) *ABCDE*	$r \vee (p \cdot q)$				*ABCEF, ABCEG*	2	3
(52) *ABCDF*	$r \vee (\bar{p} \cdot q)$				*ABCDG, ABDEF, ABEFG, ACDEG, ACEFG*	2	4
(53) *ABCDH*	$r \vee -(p \vee q)$	$r \vee (p \downarrow q)$			*ABEFH, ACEGH*	2	3
(54) *ABCEH*	$(r \equiv (p \vee q)) \vee (p \cdot q)$					3	5
(55) *ABCFG*	$(p \cdot r) \vee (p \forall q)$				*ABDEG, ACDEF*	3	4
(56) *ABCFH*	$r \equiv (p \vee (q \cdot r))$				*ABCGH, ABDEH, ABEGH, ACDEH, ACEFH*	3	4
(57) *ABDFG*	$(p \forall (q \vee r)) \vee (q \cdot r)$				*ACDFG, ADEFG*	3	5
(58) *ABDFH*	$\bar{p} \vee (q \cdot r)$		p		*ACDGH, AEFGH*	2	3
(59) *ABDGH*	$q \equiv (r \cdot (p \vee q))$				*ABFGH, ACDFH, ACFGH, ADEFH, ADEGH*	3	4
(60) *ADFGH*	$(p \equiv (q \cdot r)) \vee -(q \vee r)$	$(p \equiv (q \cdot r)) \vee (q \downarrow r)$				3	5
(61) *BCDEF*	$(\bar{q} \cdot p) \vee (q \forall r)$				*BCDEG, BCEFG*	3	4
(62) *BCDEH*	$(r \forall (p \cdot q)) \vee -(p \vee q)$	$(r \forall (p \cdot q)) \vee (p \downarrow q)$			*BCEFH, BCEGH*	3	5
(63) *BCDFG*	$(\bar{p} \cdot r) \vee (p \forall q)$				*BDEFG, CDEFG*	3	4
(64) *BCDFH*	$\bar{p} \vee (\bar{q} \cdot r)$		$p \mid (q \vee \bar{r})$		*BCDGH, BDEFH, BEFGH, CDEGH, CEFGH*	3	3
(65) *BCFGH*	$-(p \vee r) \vee (p \forall q)$	$(p \downarrow r) \vee (p \forall q)$			*BDEGH, CDEFH*	3	4
(66) *BDFGH*	$-(p \cdot (p \vee r))$		$p \mid (q \vee r)$		*CDFGH, DEFGH*	2	3
(67) *ABCDEF*	$q \vee r$				*ABCDEG, ABCEFG*	1	2
(68) *ABCDEH*	$r \vee (p \equiv q)$				*ABCEFH, ABCEGH*	2	3
(69) *ABCDFG*	$r \vee (p \forall q)$				*ABDEFG, ACDEFG*	2	3
(70) *ABCDFH*	$\bar{p} \vee r$				*ABCDGH, ABDEFH, ABEFGH*	2	3



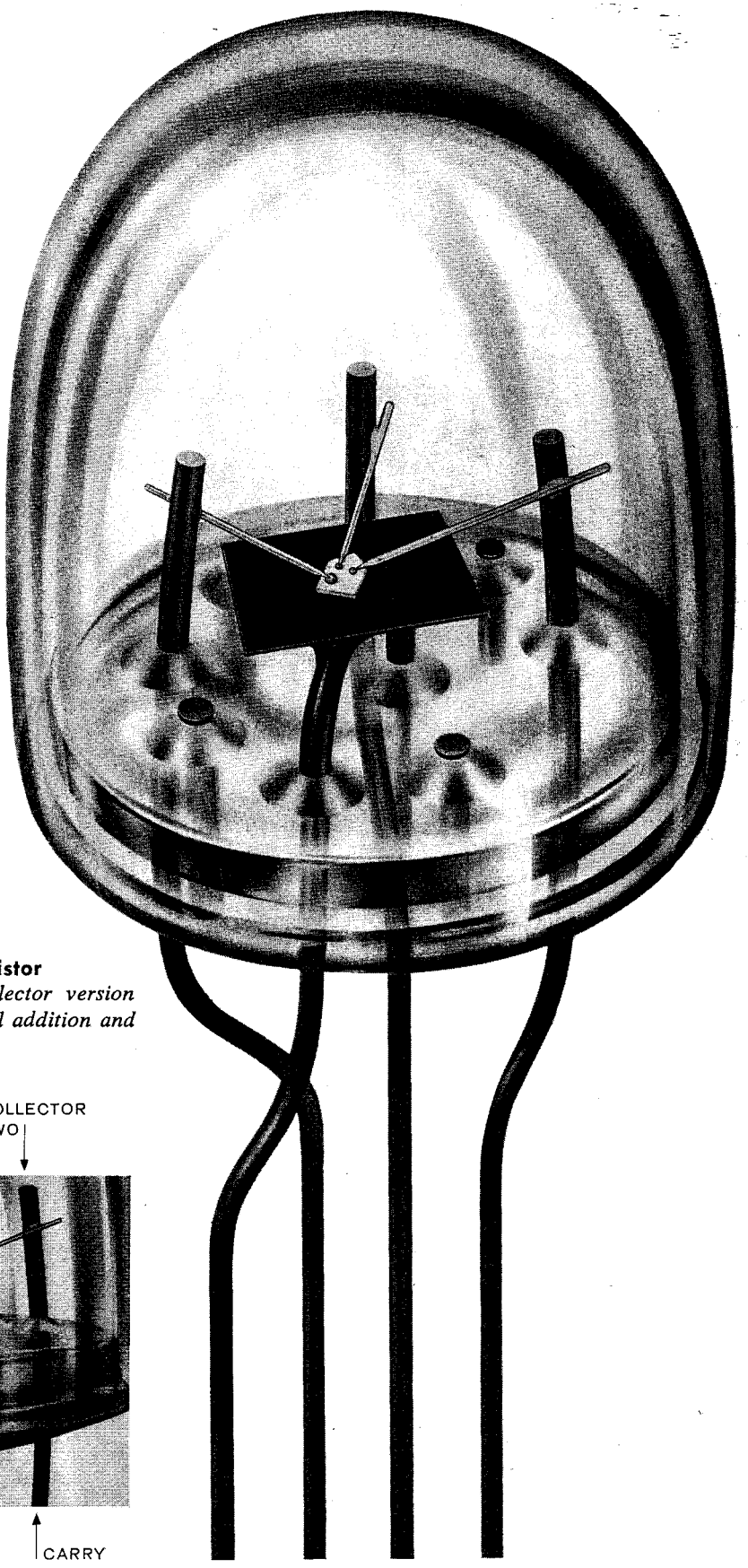
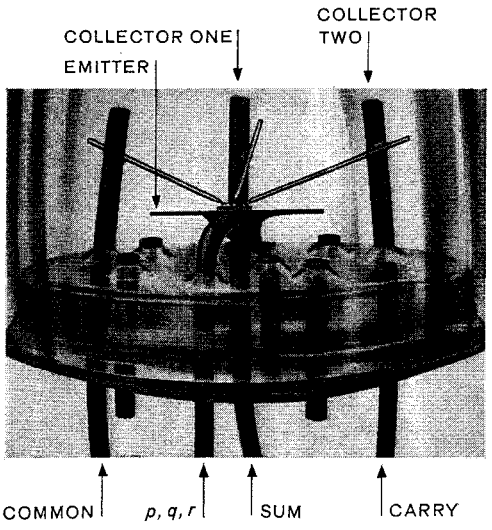
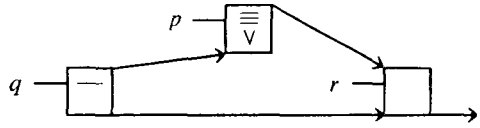


Figure 1 **The Rutz commutator transistor**
Shown here is a "hook" collector version which will perform binary full addition and other logical operations.



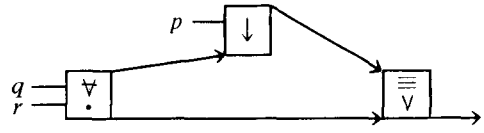
symbol '  ' indicates that '$\forall$' is obtained at the upper-arrow output; '$\cdot$' is realized at the lower. '  ', on the other hand, delineates a transistor biased to achieve '$-$'. For example, where 'p' represents the sole variable input, '$\bar{p}$' is realized at the upper-arrow output; and the 'p' itself, at the lower. '  ' and '  ' as the two notations show, designate transistors adapted to achieve '$\downarrow$' and '$|$' respectively.

Consider the following graphic expression:



From the discussion just concluded, it can be seen that '$\bar{q} \equiv p$', 'r', and 'q' represent the three variable inputs of the commutator denoted by '  '. Simple truth-table analysis will reveal that, for such components, the ternary connective '  ' generates an expression with the label '*ABCF*'.

Next, let us examine a different formula:



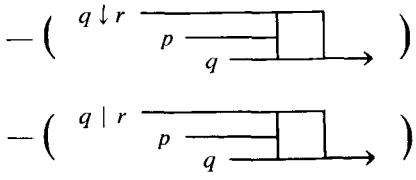
It is clear that '$q \cdot r$' and '$p \downarrow (q \forall r)$' represent the two variable inputs of the transistor symbolized '  '. Since '$\vee$' functions as the main connective, an expression with the label '*ABH*' is obtained.

One final point should be touched. The thirty expressions which comprise Schematic Diagram 1 are numbered in consecutive order by the numeral preceding the first dash of their respective headings. The numeral and label following the first dash indicate which guide basic group of Table 1 is under consideration.

It can be seen, that, if no redundant signal loads are allowed, three commutators are still sufficient for all but the five starred cases: that is, 8-23-*BCF*, 10-26-*BDG*, 18-49-*BDGH*, 22-56-*ABCFH*, and 24-59-*ABDGH*. These five, insofar as we have determined, require four transistors.

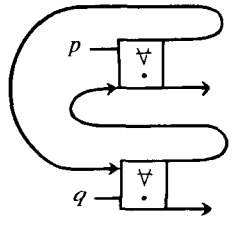
An additional problem of some interest concerns the applicability of Table 1 and Schematic Diagram 1 to full adders of a different type from the Rutz binary commutator. It is clear from the Introduction that the latter device, although quite simple, is logically more versatile than the usual full adder in that it can be adjusted to achieve '$\downarrow$' and '$|$'. Every expression in Table 1 or Schematic Diagram 1, however, which does not contain '$\downarrow$' or '$|$' can be accomplished along the lines already indicated through the use of any kind of full adder.

A direct examination of the two tables reveals that three full adders as such, not necessarily commutators, are sufficient to handle seventy-eight of the eighty guide basic groups. In fact, less than three are adequate in thirty-five cases. The two remaining groups, *BDH* and *BCDFH*, can be realized with four full adders. It might be noted, however, that, if either one of the connectives '$\downarrow$' or '$|$' were added, the two groups in question could be achieved with only three physical devices. Table 1 records '$p \downarrow (q \cdot \bar{r})$' for *BDH*, and '$p \mid (q \vee \bar{r})$' for *BCDFH*. The two groups can also be accomplished respectively by the unlisted expressions which follow:



A further question of some importance concerns the use of commutator hookups involving feedback. These often possess unique logical properties.

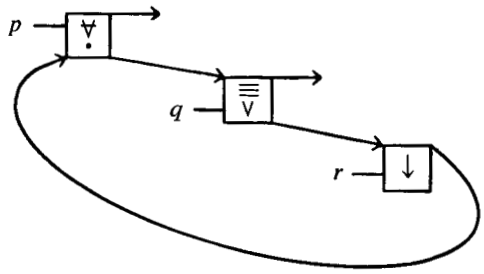
Consider the following expression:

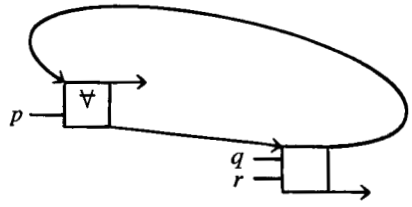


If 0 is assigned to 'p' and 1 to 'q', an oscillatory situation arises. The graphic arrow '  ', symbolizing both an output and an input, cannot consistently be allotted either 1 or 0. If 1 is assigned, simple reflection shows that 0 is generated at the arrow in question. If 0 is assigned, 1 is obtained. There is thus a fluctuation between 1 and 0, so that any physical counterpart of the above expression would presumably oscillate.

A great many commutator feedback circuits are of an oscillatory nature. The input conditions which lead to

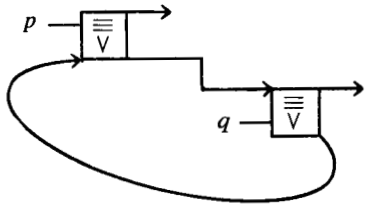
oscillation vary from hookup to hookup. Consider, for example, the two graphic formulas below:





The first expression involves fluctuation just in that case where 1 is assigned to 'p' and 0 to 'q' and 'r'. For the second expression, two input states entail fluctuation: first, where 0 is assigned to 'q', and 1 to 'p' and 'r'; second, where 0 is assigned to 'r', and 1 to 'p' and 'q'.

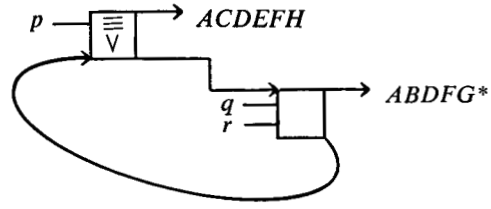
Another somewhat distinctive characteristic of feedback hookups can be seen from the following non-oscillatory expression:

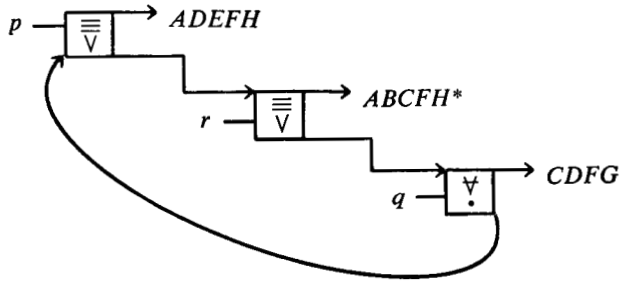


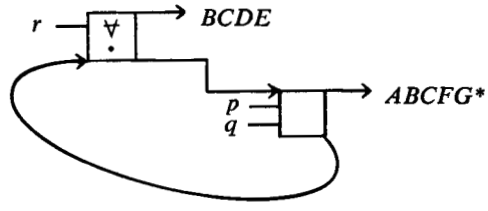
If either 'p' or 'q' is assigned 1, clearly both of the connecting output-input arrows, that is, '↻' and '↳', must be allotted 1. Should 'p' and 'q' be assigned 0, however, an unusual circumstance arises. The two arrows in question can with full consistency receive like assignments of 1 or 0; 1 will tend to perpetuate 1, and 0, 0. Since the two output arrows (inscribed '$\rightarrow$') are in part a function of '↻' and '↳', this formula, as against the formulas of earlier sections, is logically ambiguous.

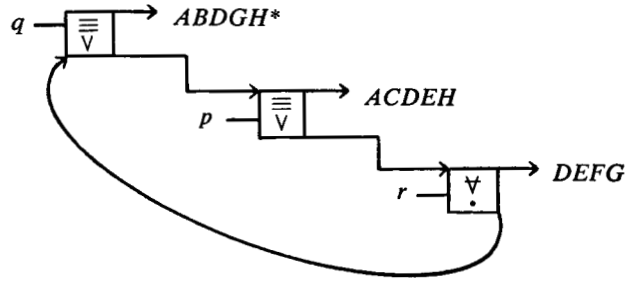
Let us imagine, however, a physical realization of the foregoing expression. The circuit may be so fixed that, before each new group of signals, all of the non-feedback inputs, including biases, are cut off, that is, reset-to-zero. The two wires, graphically represented by '↻' and '↳', will also cut off, and stay off for the troublesome input condition correlative with the assignment of 0 to 'p' and 'q'. As a result, the suggested circuit can receive a definitive truth-table representation.

There are many such ambiguous feedback expressions, some oscillatory and some not. As the circuit application varies, however, the need for zero-reset may or may not prove disadvantageous. The formulas listed below are typical. We have indicated, after each output arrow, the basic group obtained on the assumption of zero-reset. The starred cases are of most interest.







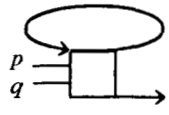


It should be remembered that in Table 1 three commutators, as against two, were needed to achieve either *ABDFG* or *ABCFG*. In Diagram 1, where no extra signal loads were admitted, four commutators were necessary for either *ABCFH* or *ABDGH*.

Ambiguous feedback circuits without zero-reset should not be disregarded, however. Such hookups are indeterminate, but only insofar as their immediate condition is a function of more than the concurrent non-feedback inputs. The prior logical state of the circuit is also relevant. As a result, memory is involved.

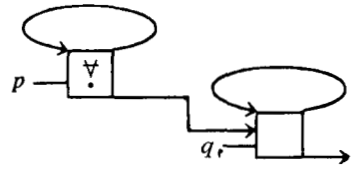
Let us suppose, for example, that the eight feedback formulas already encountered in this section represent eight circuits so operated that input signals follow one another without break. If a list of the temporally ordered signals correlative with assignments of 1 or 0 to '*p*', '*q*', and '*r*' were provided, it is possible that we might for each step determine the progressive logical states of the hookup. In some cases, a signal might be trapped in the feedback lines at one time step which would influence the operation of the circuit at another. The added property of memory might then be used to advantage.

A somewhat analogous possibility can be seen from the following:



If 0 is assigned to '*p*' and 1 to '*q*', the feedback arrow '↻' must fluctuate between 1 and 0. Let us assume however, a physical realization of the above. The circuit as a whole might be so adjusted that a properly shaped input signal could activate the line symbolized '↻' and terminate before oscillation begins. The line in question would then contain a self-perpetuating on signal. This signal, combined with a later on signal at either of the two external inputs would generate a positive signal at the output symbolized '⟶'. Such a hookup, if it could actually be made to work, would function somewhat like a flip-flop. The use of two variable inputs, as against one, however, might permit a more extended application.

Consider, for example, the circuit indicated below:



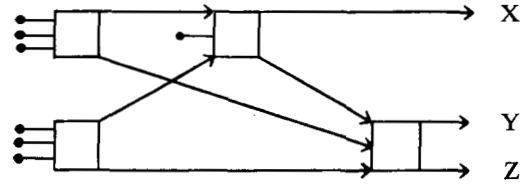
Let us suppose that the positive input signals are regulated in the manner just described and that they occupy uniform time steps. For each time step, the inputs represented by '*p*' and '*q*' might both receive a positive signal. Positive signals would then be generated at the output symbolized '⟶' three out of every four time steps.

One further topic will be mentioned briefly in this section. The Rutz commutator is a single-emitter, amplitude-sensitive device. It has been described as having three inputs; and, in fact, three separate input wires are directed into the emitter when it is used as a commutator. The action of the transistor is a function of the amplitude of the input signal. Consider the table below, which is a variation of the table shown in the Introduction:

p	*q*	*r*	CARRY	SUM
0	0	0	0	0
0	0	1		
0	1	0	0	1
1	0	0		
0	1	1		
1	0	1	1	0
1	1	0		
1	1	1	1	1

It can be seen that the 'SUM' and 'CARRY' outputs, taken together, represent the first four binary numbers as a result of the superposition of the amplitudes of the input signals. Note that the three cases where only one input is on, which are distinguishable from a logical point of view, are lumped together as giving the same binary number output. This is also the case where exactly two inputs are on.

Consider, then the following expression:



The output denoted by 'X' will be on if and only if exactly one, three, five, or seven of the external inputs (symbolized '•—') are on. The output denoted by 'Y' will be on if and only if exactly two, three, six, or seven

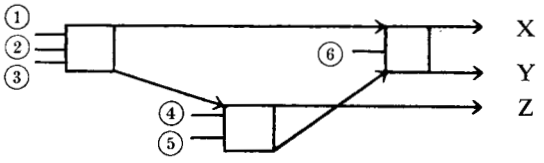
of the inputs are on. The output denoted by 'Z' will be on if and only if four or more of the inputs are on. This information is recorded in the table below:

Inputs On	Z	Y	X
None	0	0	0
One	0	0	1
Two	0	1	0
Three	0	1	1
Four	1	0	0
Five	1	0	1
Six	1	1	0
Seven	1	1	1

It can be seen that the first eight binary numbers are accomplished.

"Black-box" arrangements

Another possible use of the Rutz transistor is in the fabrication of larger building blocks which are themselves multipurpose elements. Consider the arrangement suggested below:



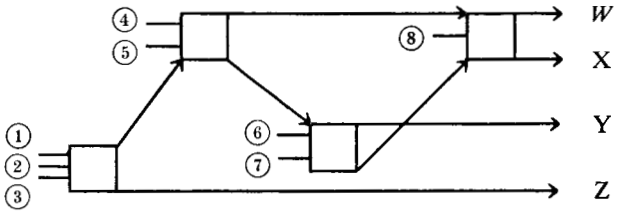
The three symbolized full adders can be placed inside an imaginary "black box" with six wires entering and three coming out. An apt selection of input wires to carry biases and variable inputs will enable the box in question to accomplish numerous logical operations. In fact, as Table 2 will show, more than half of the eighty guide basic groups of Table 1 can be achieved without extra signal loads.

In Table 2, the column headed 'Code Equivalent' requires some explanation. The six input wires are symbolized and ordered by the circled numerals above. The code designation '*p q* 1 *r* 0 0 Y', opposite the label '*A*', signifies that guide basic group *A* is obtained at the output denoted by 'Y'. The input conditions are also specified. '*p*', '*q*', and '*r*' represent the variable inputs at the first, second, and fourth wires respectively. The third wire receives a positive bias; the fifth and sixth wires, a negative one.

From Table 2 it can be seen that the fixed combination just shown (typical of many such arrangements) is adequate without extra signal loads for forty-five of the eighty guide basic groups. Thirty-three of the groups obtained represent full ternary functions.

It should be noted, however, that the adjustments to achieve the various logical functions are external. Consequently, the Rutz transistor operates merely as a full adder, and is not alternatively adapted to realize '↓' or '|'.

Next, let us examine a different expression shown graphically below:



The hookup indicated is adequate for all eighty guide basic groups, if extra signal loads are permitted in twenty-two cases. Table 3 will provide the particulars.

The column headed 'Code Equivalent' is, of course, similar to that of Table 2. The twenty-two special cases are starred. We have not listed the concurrent outputs, but these can be easily determined by the reader.

It is probable that even more effective combinations of three or four Rutz transistors can be worked out. Nevertheless, the two examples taken do indicate the broad adaptability of such "black-box" arrangements. We have not listed the quaternary nor quinary connectives patently obtainable, but we have illustrated in some detail the cumulative way in which many-variable functions include lesser ones. It is the latter logical phenomenon which underlies the great versatility of multipurpose bias devices.

In closing Part I, the author wishes to thank R. F. Rutz for his generous assistance. A. Cobham, H. Fleisher, M. K. Haynes, L. P. Hunter, R. W. Landauer, and J. A. Swanson have made many helpful suggestions.

Table 2 **Generation of Functions by Arrangement No. 1**

Guide Basic Group	*Code Equivalent*	*Concurrent Outputs*
(1) — (*ABCDEFGH*)	*p q* 0 *r* 0 0 Y	X *ABCFG*, Z *BCDE*
(2) *A*	*p q* 1 *r* 0 0 Y	X *BCDEH*, Z *DEFG*
(3) *B*	*p* 1 0 *q r* 0 Y	X *ACDEFH*, Z *ADFG*
(6) *AB*	*q* 0 0 0 0 *r* Y	X *CDEF*, Z — (*ABCDEFGH*)
(7) *AD*	*p q* 1 0 0 *r* Y	X *BCEH*, Z *ABCEFG*
(8) *AH*	*p q r* 1 0 1 X	Y *ABCDEFG*, Z *DFGH*
(9) *BC*	*p q* 0 0 0 *r* Y	X *ADFG*, Z *AE*
(10) *BD*	*p* 1 0 0 0 *r* Y	X *ACFH*, Z *ACEG*
(11) *BG*	*p* 1 0 *q r* 1 X	Y *ABCDEFH*, Z *ADFG*
(14) *DH*	1 1 1 *p q* 0 X	Y *ABCEFG*, Z *ADEH*
(15) *ABC*	*p q* 0 1 0 *r* Y	X *DEFG*, Z *BCDFGH*
(16) *ABD*	*p* 1 0 *q* 0 *r* Y	X *CEFH*, Z *BCFG*
(17) *ABG*	*q r* 1 *p* 1 1 X	Y *ABCDEFGH*, Z *ACEH*
(19) *ADF*	*q r* 1 *p* 0 1 X	Y *ABCEGH*, Z *BDFG*
(24) *BCH*	*p q* 0 *r* 1 1 X	Y *ABCDEFG*, Z *AFGH*
(28) *BGH*	*q r* 0 *p* 0 1 X	Y *ACDEF*, Z *BCEG*
(31) *ABCD*	*r* 1 0 0 0 0 Z	X *EFGH*, Y — (*ABCDEFGH*)
(32) *ABCE*	*p* 0 0 *q* 1 *r* Y	X *ADFG*, Z *CDGH*
(34) *ABCH*	*p q* 1 *r* 1 1 Z	X *ADE*, Y *ABCDEFGH*
(38) *ABGH*	*q* 1 0 *r* 1 *p* Z	X *BCFH*, Y *ACDEG*
(39) *ADFG*	*p* 1 0 *q r* 1 Z	X *BG*, Y *ABCDEFH*
(40) *ADFH*	*q r* 0 *p* 1 1 Z	X *CEH*, Y *ABCDEFG*
(41) *BCDE*	*p q* 0 *r* 0 0 Z	X *ABCFG*, Y — (*ABCDEFGH*)
(44) *BCEH*	*p q r* 0 0 1 X	Y *ADFG*, Z *ABCE*
(45) *BCFG*	*p* 1 0 *q* 0 1 Z	X *CG*, Y *ABDEFH*
(46) *BCFH*	*q* 1 0 *r* 1 *p* X	Y *ACDEG*, Z *ABGH*
(47) *BDFG*	*q r* 1 *p* 0 1 Z	X *ADF*, Y *ABCEGH*
(48) *BDFH*	*p* 1 0 0 0 0 X	Y — (*ABCDEFGH*), Z *ACEG*
(50) *DFGH*	*p q r* 1 0 1 Z	X *AH*, Y *ABCDEFG*
(51) *ABCDE*	*r* 0 0 *p q* 1 Y	X *AFGH*, Z *BCFG*
(52) *ABCDF*	*p* 1 0 *q* 1 *r* Y	X *BEGH*, Z *ADEH*
(55) *ABCFG*	*p q* 0 *r* 0 1 Y	X *DEH*, Z *BCDE*
(57) *ABDFG*	*q r* 0 *p* 1 0 X	Y *CE*, Z *ADFH*
(62) *BCDEH*	*p q* 1 *r* 0 0 X	Y *A*, Z *DEFG*
(65) *BCFGH*	*p q* 1 *r* 1 0 X	Y *ADE*, Z *ABCH*
(67) *ABCDEF*	*q* 1 1 1 1 *r* Y	X *ABGH*, Z *ABCDEFGH*
(68) *ABCDEH*	*p q* 1 *r* 0 1 Y	X *AFG*, Z *DEFG*
(69) *ABCDFG*	*p q* 0 1 1 *r* Y	X *BCEH*, Z *AE*
(70) *ABCDFH*	*p* 1 0 1 1 *r* Y	X *BDEG*, Z *ACEG*
(71) *ABCFGH*	*r* 1 0 *p q* 0 X	Y *E*, Z *ADFG*
(73) *BCDEFG*	*p q r* 1 0 0 X	Y *A*, Z *DFGH*
(75) *BCDFGH*	*p q* 0 1 0 1 Z	X *DH*, Y *ABCEFG*
(76) *ABCDEFG*	*p q* 0 *r* 1 1 Y	X *BCH*, Z *AFGH*
(77) *ABCDEFH*	*p* 1 0 *q r* 1 Y	X *BG*, Z *ADFG*
(80) *ABCDEFGH*	*p q* 1 *r* 1 1 Y	X *ADE*, Z *ABCH*

Table 3 **Generation of Functions by Arrangement No. 2**

Guide Basic Group	*Code Equivalent*
(1) — (*ABCDEFGH*)	1 0 0 *p q r* 0 0 Z
(2) *A*	1 0 0 *p q r* 0 0 X
(3) *B*	0 0 0 *p* 1 *q r* 0 X
*(4) *D*	*p q r r* 0 *p* 1 *q W*
*(5) *H*	*p q* 0 *p q r* 1 1 *W*
(6) *AB*	0 0 0 *q* 0 *p* 0 *r* X
(7) *AD*	*p q* 1 *r* 0 0 0 0 Y
(8) *AH*	*p* 0 0 *q r* 1 0 1 *W*
(9) *BC*	0 0 0 *p q r* 1 0 X
(10) *BD*	*p* 1 0 *r* 0 0 0 0 Y
(11) *BG*	0 0 0 *p* 1 *q r* 1 *W*
(12) *BH*	*q r* 0 *p* 1 1 0 0 Y
(13) *DF*	*q r* 1 *p* 1 1 0 0 Y
(14) *DH*	1 0 0 *p q* 1 0 0 Y
(15) *ABC*	0 0 0 *p q* 1 0 *r* X
(16) *ABD*	0 0 0 *p* 1 *q* 0 *r* X
(17) *ABG*	1 0 0 *q r p* 1 1 *W*
*(18) *ABH*	1 0 0 *q r p* 0 *p W*
(19) *ADF*	1 0 0 *q r p* 0 1 *W*
*(20) *ADH*	*r* 0 0 *p q r* 1 1 *W*
*(21) *BCD*	0 0 0 *q r p* 0 *q W*
*(22) *BCE*	*p* 1 0 *q r p* 1 0 X
(23) *BCF*	*p* 1 0 *q* 1 *r* 1 0 X
(24) *BCH*	0 0 0 *p q r* 1 1 *W*
(25) *BDF*	*p* 1 0 1 1 *q r* 0 X
(26) *BDG*	*r* 1 0 *p* 0 *q* 0 1 *W*
*(27) *BDH*	*q* 1 0 *p* 1 *r* 1 *q W*
(28) *BGH*	0 0 0 *q r p* 0 1 *W*
*(29) *DFG*	0 0 0 *p q r* 0 *r W*
*(30) *DFH*	*p* 1 0 *r* 0 *q* 0 *r W*
(31) *ABCD*	0 0 0 *p* 0 *r* 0 *q* Y
(32) *ABCE*	*p q r* 0 0 0 0 0 Z
*(33) *ABCF*	*q* 1 0 *p* 1 *r* 0 *q* X
(34) *ABCH*	1 0 0 *p q r* 1 0 Y
(35) *ABDF*	*p* 1 0 *q r* 0 0 0 Y
(36) *ABDG*	*p* 1 0 *q* 1 *r* 1 0 Y
*(37) *ABDH*	*p q r q* 0 *p* 1 *r* Y
(38) *ABGH*	*q* 1 0 *r* 0 0 0 0 *W*
(39) *ADFG*	0 0 0 *p* 1 *q r* 0 Y
(40) *ADFH*	0 0 0 *q r p* 1 0 Y

Guide Basic Group	*Code Equivalent*
(41) *BCDE*	0 0 0 *p q r* 0 0 Y
*(42) *BCDF*	*p q r q* 0 *r* 0 1 Y
(43) *BCDH*	*r* 1 0 *p q* 1 0 0 Y
(44) *BCEH*	*p q* 1 *r* 0 0 0 0 *W*
(45) *BCFG*	0 0 0 *p* 0 *r* 0 *q W*
(46) *BCFH*	0 0 0 *r* 1 *q* 0 *p W*
(47) *BDFG*	1 0 0 *q r p* 0 0 Y
(48) *BDFH*	*p* 0 0 1 0 0 0 0 *W*
*(49) *BDGH*	*p q r* 1 0 *r* 0 *p W*
(50) *DFGH*	*p* 0 0 *q r* 1 0 1 Y
(51) *ABCDE*	0 0 0 1 0 *p q r* X
(52) *ABCDF*	0 0 0 *p* 1 *q* 1 *r* X
*(53) *ABCDH*	1 0 0 *q r p* 1 *q W*
*(54) *ABCEH*	*p* 1 0 *q r p* 0 0 *W*
(55) *ABCFG*	0 0 0 *p q r* 0 1 X
(56) *ABCFH*	*p* 1 0 *r* 0 *q* 0 1 X
(57) *ABDFG*	0 0 0 *q r p* 1 0 *W*
(58) *ABDFH*	*p* 1 0 0 0 *q r* 1 X
(59) *ABDGH*	*r* 1 0 *q* 1 *p* 1 0 *W*
*(60) *ADFGH*	*p* 1 0 *q r p* 1 0 *W*
*(61) *BCDEF*	0 0 0 *q r p* 1 *p W*
(62) *BCDEH*	1 0 0 *p q r* 0 0 *W*
*(63) *BCDFG*	*r* 0 0 *p q r* 0 0 *W*
*(64) *BCDFH*	*q* 1 0 *p* 0 *r* 0 *q W*
(65) *BCFGH*	1 0 0 *p q r* 1 0 *W*
*(66) *BDFGH*	*p* 1 0 *q* 1 *r* 1 *q W*
(67) *ABCDEF*	*q r* 1 0 0 0 0 0 Z
(68) *ABCDEH*	1 0 0 *p q r* 0 1 X
(69) *ABCDFG*	*p q* 0 *r* 1 0 0 0 Y
(70) *ABCDFH*	*p* 1 0 *r* 1 0 0 0 Y
(71) *ABCFGH*	0 0 0 *r* 1 *p q* 0 *W*
(72) *ABDFGH*	*q r* 0 *p* 1 1 0 0 *W*
(73) *BCDEFG*	*p* 0 0 *q r* 1 0 0 *W*
(74) *BCDEFH*	*q r* 1 *p* 1 1 0 0 *W*
(75) *BCDFGH*	1 0 0 *p q* 1 0 0 *W*
(76) *ABCDEFG*	0 0 0 *p q r* 1 1 X
(77) *ABCDEFH*	0 0 0 *p* 1 *q r* 1 X
*(78) *ABCDFGH*	*p q r r* 1 *p* 0 *q W*
*(79) *BCDEFGH*	*p q* 1 *p q r* 0 0 *W*
(80) *ABCDEFGH*	1 1 0 *p q r* 0 0 Z

Bibliography

The first chapter of W. V. Quine's *Mathematical Logic* (revised edition, Harvard University Press, 1951) will provide to the reader unacquainted with logic all the information necessary to understand the foregoing discussion. Some of the more interesting works which bear upon the application of elementary logic to the design of circuits are the following:

1. C. E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits," *Transactions of the AIEE,* **57,** 713 (1948).
2. Staff of the Computation Laboratory, *Synthesis of Electronic Computing and Control Circuits,* Harvard University Press, 1951.
3. Keister, Ritchie, and Washburn, *The Design of Switching Circuits,* D. Van Nostrand Co., 1951.
4. Goodell, Sobociński, and others, a series of papers dealing with typical logical particles which can be assembled machine-wise to generate the whole class of truth-functional elements, *The Journal of Computing Systems,* **1,** nos. 1-4 (1953-4).

Received December 28, 1956