Logic design — 11

Design with m.s.i. — multiplexers and demultiplexers

by B. Holdsworth* and **D. Zissos†**

*Chelsea College, University of London †Dept. of Computing Science, University of Calgary, Canada

The introduction of m.s.i. circuits is tending to result in the replacement of the old methods of logic design. Traditionally, the design engineer has developed a logic function as the solution to a particular problem. This function has then been minimized using the methods described earlier in this series and has been implemented using s.s.i. circuits. However when implementing logic functions with m.s.i. circuits such as the multiplexer, the Boolean function is used in its canonical form (i.e. each term in the Boolean function contains all the variables in the true or complemented form), and is implemented directly without minimization.

THE COST of a digital system is approximately proportional to the number of i.cs in the system, hence, to reduce the cost, the number of packages used should be minimized. The logic designer should therefore be looking for the replacement of a large number of s.s.i. circuits by one or more m.s.i. packages. It is frequently better to use a standard m.s.i. package even if this introduces redundant or unused gates rather than to design with s.s.i. circuits.

Data selector or multiplexer

The multiplexer selects one out of n lines where n is usually 4, 8 or 16. A block diagram of a data selector having 4 input lines, D_0, D_1, D_2 and D_3 and 2 output lines f and $\bar{f}$ is shown in Fig. 1(a). The device also has 2 control lines X and Y and may have an "enable" line E. The selector may be regarded as a single-pole switch which selects 1 out of 4 lines as shown in Fig. 1(b). The implementation of the multiplexer using gates is shown in Fig. 1(c).

In essence the circuit is an AND-OR-INVERT gate having complementary outputs. The Boolean function which represents the output of this circuit is: $f = \bar{X}\bar{Y}D_0 + \bar{X}YD_1 + X\bar{Y}D_2 + XYD_3$. Data lines can be selected by applying the appropriate binary coded signal to the control lines X and Y: when the control signal $\bar{X}\bar{Y} = 1$ the output of the circuit is D_0, and so on. Some multiplexers are provided with an input enable line as shown in Fig. 1(c). When the input to this line is logical 0 the four AND gates are enabled.

The number of data lines to be selected can be increased either by choosing a multiplexer with a larger number of data lines or alternatively by combining multiplexers. A combination of two data-selectors, which allows the selection of 1 out of 8 lines, is shown in Fig. 2, the enable signal in this case being used as an additional control signal. The data lines are sequentially selected with the aid of a binary counter, the control signals X and Y being clocked through the sequence 00, 01, 10 and 11, thus accessing the data

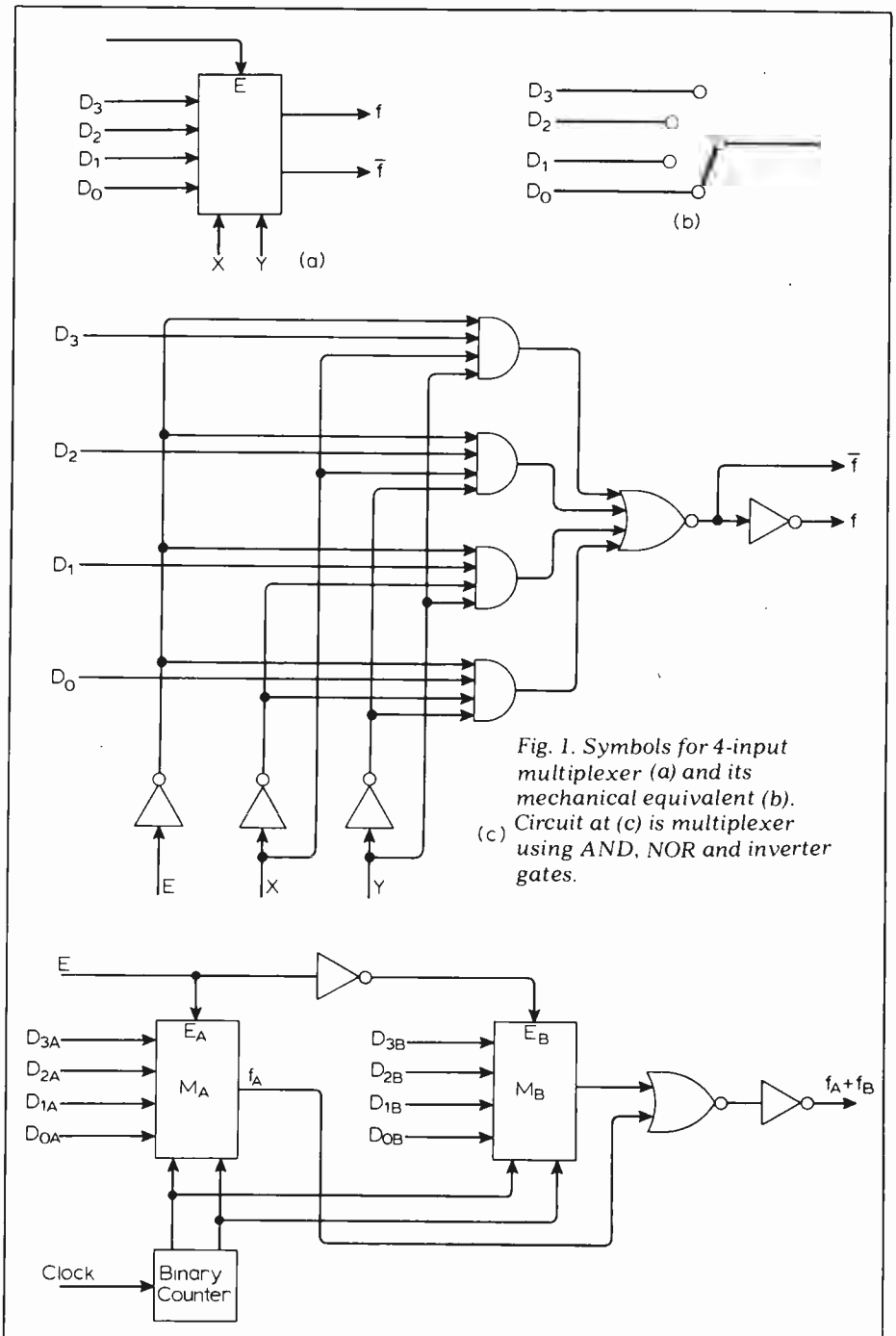


Fig. 1. Symbols for 4-input multiplexer (a) and its mechanical equivalent (b). Circuit at (c) is multiplexer using AND, NOR and inverter gates.

Fig. 2. Multiplexers can be combined to provide more inputs.

lines in the order D_0, D_1, —D_7. A truth table for the circuit is shown in Table 1. This principle can be extended to allow the selection of a larger number of data lines. For example, the selection of 1 out of 64 lines can be achieved using nine 8-input multiplexers, as shown in Fig. 3, arranged in two levels of multiplexing.

An alternative way of looking at the multiplexer is to regard it as a device which converts parallel information into serial information. For example, in the arrangement shown in Fig. 2(a), the two multiplexers M_A and M_B can be presented with an 8-bit word on the 8 input lines in parallel form, and this can be taken off in serial form by using the sequential accessing technique.

Multiplexer as logic function generator

The equation for a multiplexer having four input lines is:

$f = \bar{A}\bar{B}D_0 + \bar{A}BD_1 + A\bar{B}D_2 + ABD_3$,

where the Boolean variables A and B are used as the signals for the control lines X and Y. Hence A and B can be factored out of any function of n variables, and the residue functions of n 2 variables can then be applied to the data lines. For example if n=3, four signals of one variable can be applied to each of the data lines. Assuming that the third variable is C the possible signals that can be applied to these lines are C, $\bar{C}$, 0 and 1. In all there are $4^4 = 256$ possible combinations of four input signals which can be applied to the 4-input lines; a multiplexer with 4 input lines can generate any of the 256 possible Boolean functions of 3 variables.

For the 4-input multiplexer there are three possible choices for the control variables – AB, AC and BC. These various combinations of the control variables can be associated with individual data lines as indicated in Fig. 4. For example, with control variables A and B, the input line D_0 is associated with those cells marked A=0 and B=0, that is the two top left-hand cells on the K-map of Fig. 4(a). In effect, the K-map for 3-variables has now been split into four 2-cell, 1-variable K-maps, each of these 2-cell maps being associated with a data line.

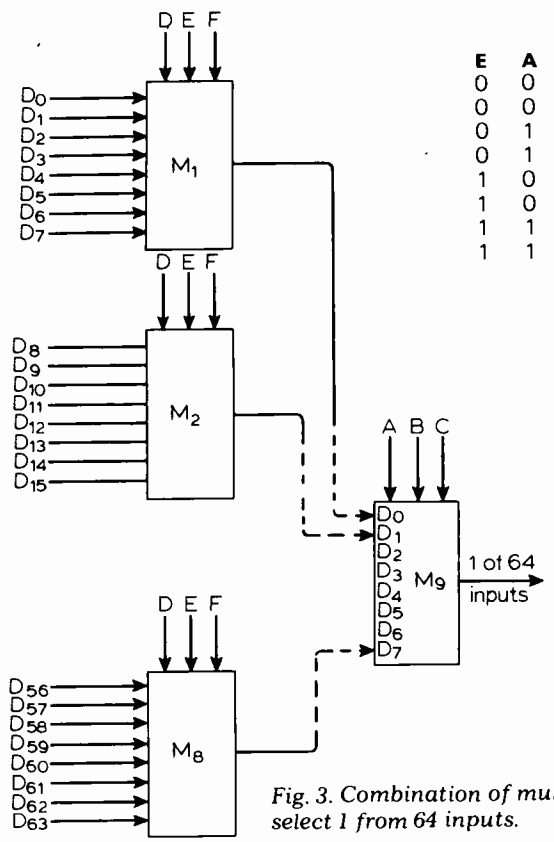


Table 1

E	A	B	E_A	E_B	
0	0	0	0	1	
0	0	1	0	1	M_B inhibited
0	1	0	0	1	
0	1	1	0	1	
1	0	0	1	0	
1	0	1	1	0	M_A inhibited
1	1	0	1	0	
1	1	1	1	0	

Fig. 3. Combination of multiplexers to select 1 from 64 inputs.

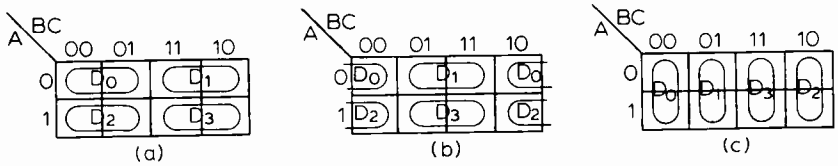


Fig. 4. Association of data lines with control signals for 4-input multiplexer. Control variables are A and B in (a), A and C in (b) and B and C in (c).

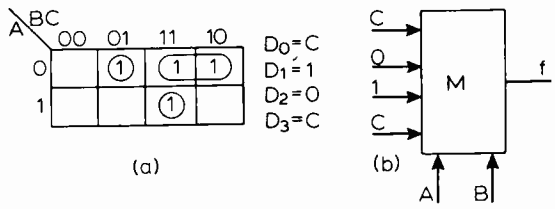
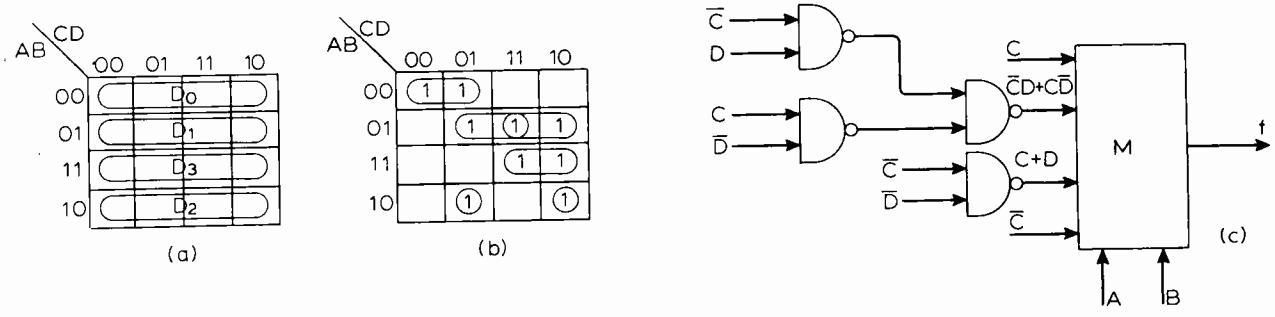


Fig. 5. Generation of $f = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + ABC$ using a 4-input multiplexer.

Fig. 6. Generation of f = 0,1,5,6,7,9,10,14,15. Association of data lines with variable A and B is seen at (a) and the Karnaugh map is at (b). The diagram at (c) is the implementation.



Example 1. Implement the 3-variable function

$$f = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

using a 4-input multiplexer.

Plot the function on a K-map as shown in Fig. 5(a) and make an arbitrary choice of control variables, say A and B. Next simplify the four 1-variable functions associated with each data line. For example, the two cells associated with D_1 are both marked with a 1, hence the input to data line D_2 is $C + \bar{C} = 1$. The remaining inputs are determined in the same manner and the implementation of the function is shown in Fig. 5(b).

Example 2 Implement the 4-variable function

$$f = \Sigma 0,1,5,6,7,9,10,14,15$$

using a 4-input multiplexer.

The function has been represented as the sum of a number of canonical terms, each term being represented as a decimal number. For example the term $\bar{A}BC\bar{D}$, represented in binary, is $0110 = 6$ in decimal.

Since a 4-input multiplexer is to be used, the application of two variables to its control lines will leave residue functions of two variables to be applied to the data lines. There are six possible ways of choosing the control variables – AB, AC, AD, BC, BD, and CD. These various combinations of control variables can be associated with the data lines as indicated previously in Fig. 4. It will be assumed in this example that A and B are chosen as the control variables and the K-map associating these control variables with the data lines is shown in Fig. 6(a). The 4-variable K-map has now been divided into four 4-cell, 2-variable maps and simplification can only take place within the confines of the 2-variable maps.

The K-map plot of the function is shown in Fig. 6(b) and the data line inputs obtained from the four rows of this map are:

$D_0 = \bar{C}$ address $\bar{A}\bar{B}$
$D_1 = C + D$ address $\bar{A}B$
$D_2 = \bar{C}D + C\bar{D}$ address $A\bar{B}$
$D_3 = C$ address AB

The implementation of the function is shown in Fig. 6(c).

It should be pointed out that it is useful to examine the various possible choices of control variables to ascertain whether there is a simpler solution. In this case it is left to the reader to show that a simpler solution is obtained if C and D are chosen as control variables.

As the number of variables associated with the Boolean function to be implemented increases, it becomes necessary to use more than one level of multiplexing and this technique is illustrated in the next example.

Table 2. Determination of the inputs to the 1st level multiplexer.

	$\bar{D}\bar{E}$	$\bar{D}E$	$D\bar{E}$	DE
$f = \bar{A}\bar{B}\bar{C}\bar{D}\bar{E}$	$\bar{A}\bar{B}\bar{C}$			
$+\bar{A}\bar{B}\bar{C}DE$		$\bar{A}\bar{B}\bar{C}$		
$+\bar{A}\bar{B}C\bar{D}E$				$\bar{A}\bar{B}C$
$+\bar{A}B\bar{C}\bar{D}\bar{E}$	$\bar{A}B\bar{C}$			
$+\bar{A}B\bar{C}DE$		$\bar{A}B\bar{C}$		
$+\bar{A}BC\bar{D}E$				$\bar{A}BC$
$+\bar{A}BCD\bar{E}$	$\bar{A}BC$			
$+\bar{A}BCDE$		$\bar{A}BC$		
$+A\bar{B}C\bar{D}\bar{E}$			$A\bar{B}C$	
$+A\bar{B}C\bar{D}E$	$A\bar{B}C$			
$+A\bar{B}CD\bar{E}$		$A\bar{B}C$		
$+A\bar{B}CDE$			$A\bar{B}C$	
$+AB\bar{C}D\bar{E}$			$AB\bar{C}$	
$+AB\bar{C}DE$				$AB\bar{C}$
$+ABCDE$				ABC

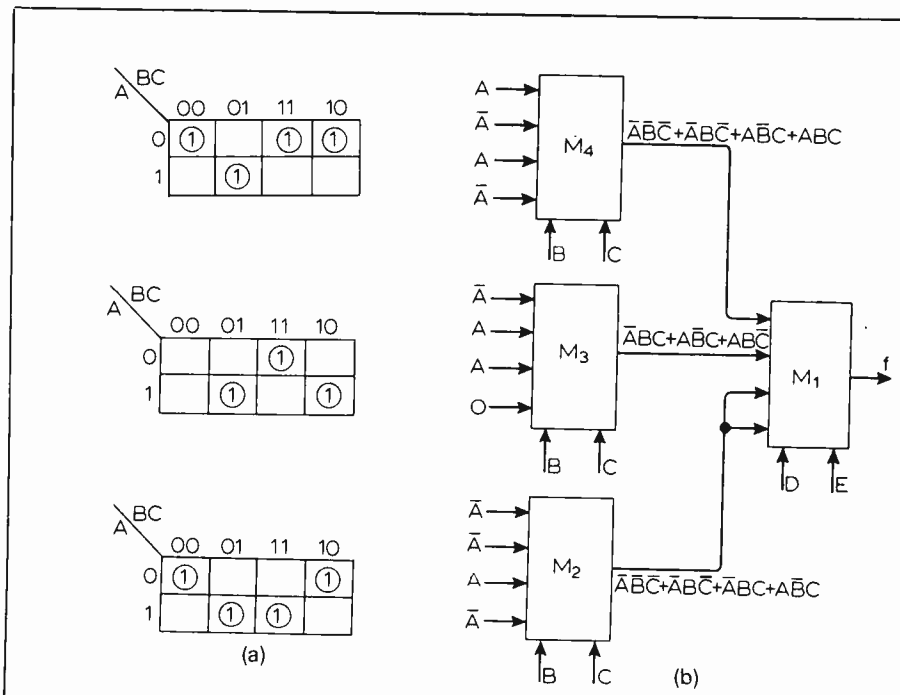
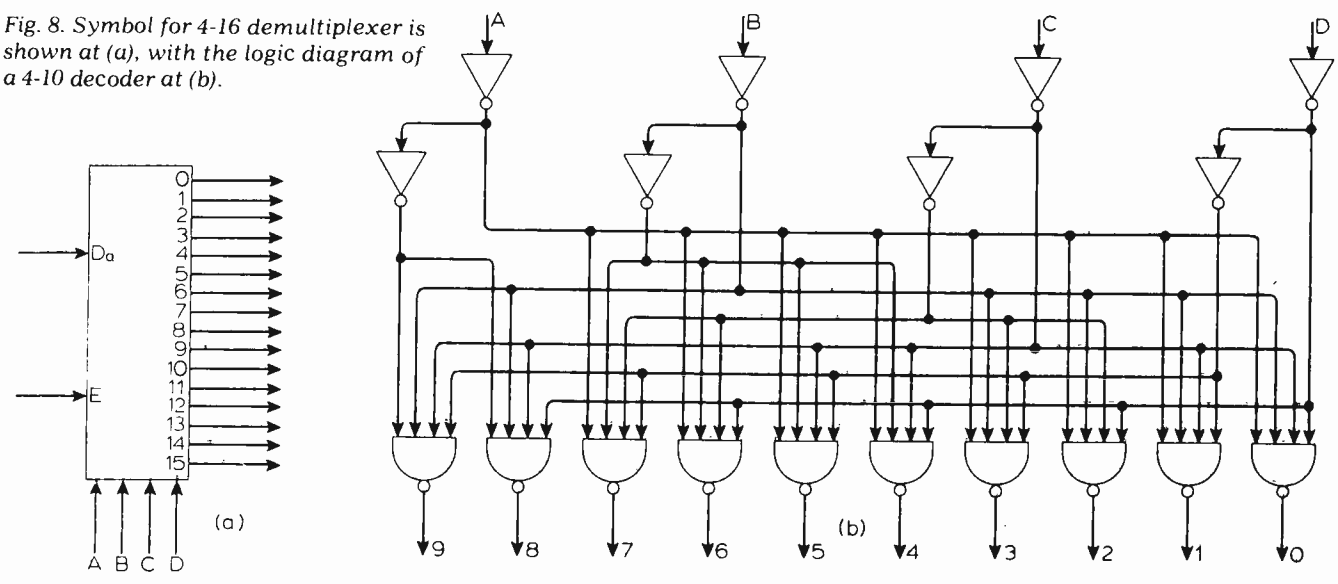


Fig. 7. Five-variable multiplexer circuit to produce the function of example 3.

Fig. 8. Symbol for 4-16 demultiplexer is shown at (a), with the logic diagram of a 4-10 decoder at (b).



Example 3. Implement the 5-variable function

$f = \Sigma 0,1,3,8,9,11,12,13,14,20,21,22,23,26,31$

For the first level of multiplexing the control variables D and E have been arbitrarily chosen. The function is now listed at the left-hand side of Table 2, which contains four columns headed $\bar{D}\bar{E}$, $\bar{D}E$, $D\bar{E}$ and DE respectively. In the column headed $\bar{D}\bar{E}$ are listed all those terms of three variables A, B, and C which are associated with $\bar{D}\bar{E}$. For example, in the case of the term $\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}$ the entry in the $\bar{D}\bar{E}$ column will be $\bar{A}\bar{B}\bar{C}$. This procedure is repeated for each term in the 5-variable function and an entry is made in the appropriate column in each case.

The input functions for the first level multiplexer are now seen to be:

$$D_{01} = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C$$
$$D_{11} = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C$$
$$D_{21} = \bar{A}BC + A\bar{B}C + AB\bar{C}$$
$$D_{31} = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}C + ABC$$

These three variable functions can be generated with 4-input multiplexers, as described in example 1, at the second level of multiplexing. However it should be noticed that $D_{01} = D_{11}$ and this function need only be generated once, hence only three second level multiplexers are required.

For the second level of multiplexing B and C have been chosen as the control variables. The K-maps for determining the inputs to the data lines for the second level multiplexers are shown in Fig. 7(a) and from these maps the various input signals are found to be:

$$D_{02} = \bar{A} \quad D_{03} = 0 \quad D_{04} = \bar{A}$$
$$D_{12} = A \quad D_{13} = A \quad D_{14} = A$$
$$D_{22} = \bar{A} \quad D_{23} = A \quad D_{24} = \bar{A}$$
$$D_{32} = \bar{A} \quad D_{33} = \bar{A} \quad D_{34} = A$$

The implementation of the function is shown in Fig. 7(b).

Decoders or Demultiplexers

A decoder or demultiplexer performs the opposite function to that of a multiplexer. A block diagram of the device is shown in Fig. 8(a). A single data input line can be connected to one of many output lines by the appropriate choice of signal on the control lines. With 4 control lines A, B, C, and D there are sixteen possible addresses and hence the maximum number of output lines that can be selected is sixteen.

A commonly used decoder has 4 input lines and 10 output lines. The logic diagram for this device is shown in Fig. 8(b). If A = 0, B = 0, C = 0 and D = 0, the output line marked 0 will be at logical 0 whilst all the other outputs will be at logical 1.

The device illustrated in Fig. 8(b) can be used as a decoder, but in a 4-to-16 line demultiplexer there are additionally enable and data lines as shown in Fig. 8(a). These are connected to the sixteen output gates via the circuit shown in Fig. 9 which is in effect a NOR gate. This input arrangement allows of two modes of operation. In the first mode, if E = O & $D_a = 0$, K = 1, thus enabling all output gates. For any other values of E & D_a, K = 0, thus disabling all output gates.

In this mode the 4-to-16 line demultiplexer will act as a decoder allowing, for example, a b.c.d. input on lines A. B. C and D to be converted to a decimal output. Alternatively the circuit can be operated as a generator of the sixteen canonical terms of four Boolean variables. If $P_3 = \bar{A}\bar{B}CD$ is the input to the control lines then the output on line 3 = $\bar{P}_3$.

In the second mode E = 0, $D_a = 0$, hence K = 1. Control signal $P_2 = \bar{A}\bar{B}C\bar{D}$. The output on line 2 = 0 = D_a.

E = 0, $D_a = 1$, hence K = 0. Control signal $P_2 = \bar{A}\bar{B}C\bar{D}$. The output on line 2 = 1 = D_a.

In this mode the data on the data line is transferred to the output gate selected by the address applied to the control lines, in this case $\bar{A}\bar{B}C\bar{D}$.

Example 4 Using a 4-to-10 line decoder develop a circuit for converting n.b.c.d. to the Lorenz code.

The two codes are tabulated alongside each other in Table 3.

	NBCD				Lorenz				
	A	**B**	**C**	**D**	**P**	**Q**	**R**	**S**	**T**
P_0	0	0	0	0	1	0	0	1	1
P_1	0	0	0	1	1	0	1	0	1
P_2	0	0	1	0	1	1	0	0	1
P_3	0	0	1	1	0	0	1	1	1
P_4	0	1	0	0	0	1	0	1	1
P_5	0	1	0	1	0	1	1	0	1
P_6	0	1	1	0	0	1	1	1	0
P_7	0	1	1	1	1	0	1	1	0
P_8	1	0	0	0	1	1	0	1	0
P_9	1	0	0	1	1	1	1	0	0

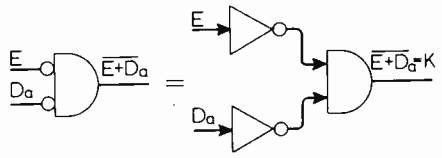


Fig. 9. Input data and enable arrangements.

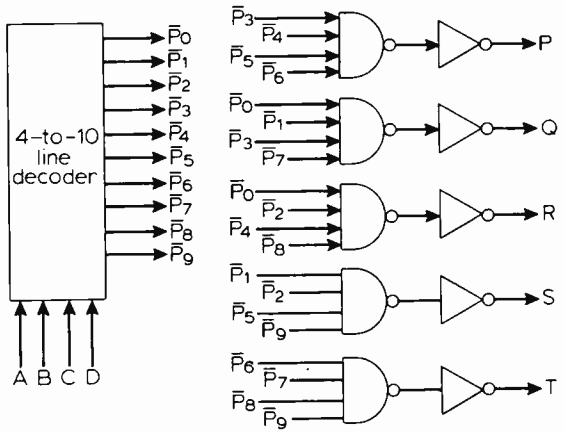


Fig 10. Natural binary-coded decimal to Lorenz converter.

From the tabulation:

$$P = P_0 + P_1 + P_2 + P_7 + P_8 + P_9$$
$$Q = P_2 + P_4 + P_5 + P_6 + P_8 + P_9$$
$$R = P_1 + P_3 + P_5 + P_6 + P_7 + P_9$$
$$S = P_0 + P_3 + P_4 + P_6 + P_7 + P_8$$
$$T + P_0 + P_1 + P_2 + P_3 + P_{-4} + P_5$$

Now $\bar{P} = P_3 + P_4 + P_5 + P_6$

Hence $\overline{\bar{P} = P_3 + P_4 + P_5 + P_6}$

and $\bar{P} = \overline{\bar{P}_3\,\bar{P}_4\,\bar{P}_5\,\bar{P}_6}$

Similarly $\bar{Q} = \overline{\bar{P}_0\bar{P}_1\bar{P}_3\bar{P}_7}$

$\bar{R} = \overline{\bar{P}_0\bar{P}_2\bar{P}_4\bar{P}_8}$

$\bar{S} = \overline{\bar{P}_1\bar{P}_2\bar{P}_5\bar{P}_9}$

$\bar{T} = \overline{\bar{P}_6\bar{P}_7\bar{P}_8\bar{P}_9}$

The implementation of the code converter is shown in Fig. 10.

The technique used in this example is useful where there are many functions of the same number of variables to be implemented. In comparison the multiplexer requires less additional gating, but one multiplexer at least is required to implement each function.

The second part of this article will deal with the applications of read-only memories.