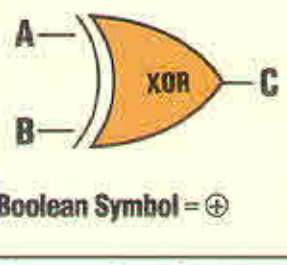
ON THE LEVEL

We are now at a stage in the Logic course where we can design quite complex computer circuits. In this instalment, we will follow through the whole design process — from initial specification, through truth table and simplified Boolean expression, to finished circuit diagram — for a parity bit generating circuit and a priority encoder.

Before beginning to look at the design of these two advanced applications, we will first take a detailed look at another important logic gate — the Exclusive OR (XOR) gate. This gate has already been briefly considered (see page 47), but we have not yet given the Boolean algebra or circuit diagram symbols for it:

Truth Table			Circuit Symbol
A	B	C	
0	0	0	
0	1	1	
1	0	1	
1	1	0	

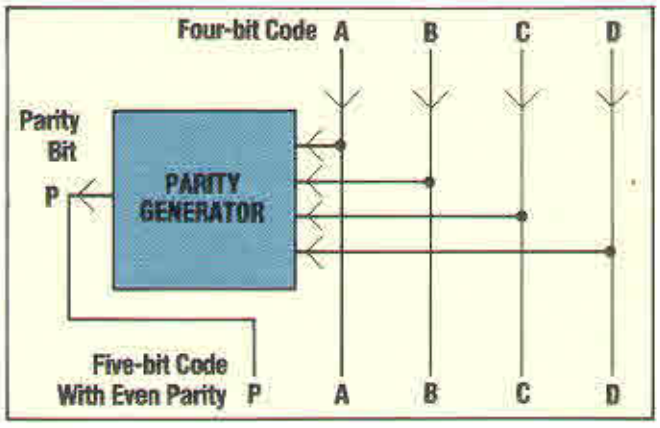


From the truth table, it can be seen that the output C can be expressed in two ways:

a) $C = A \oplus B = \bar{A}.B + A.\bar{B}$
b) $\bar{C} = \overline{A \oplus B} = \bar{A}.\bar{B} + A.B$

The second expression is formed by considering the cases when C is not one (i.e. zero). This gate will be of particular use in our first application.

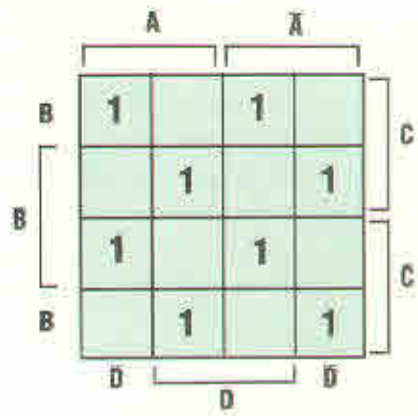
A PARITY BIT GENERATOR



Parity is an important concept in the design of data transmission systems. The parity bit (see previous diagram) of a binary code is added to the rest of the code in order to make all the codes transmitted have an even number of ones. (Another convention is to make all the codes contain an odd number of ones — this is known as *odd parity*). A parity bit acts as a checking system to ensure that the correct transmission has taken place. The circuit we shall design will accept a four-bit code and produce an appropriate parity bit. With a small modification the circuit may also act as a parity checker of incoming data. The truth table for this circuit is given in the margin. Representing these values on a k-map gives:

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

PGB Truth Table



The symmetrical pattern produced on the k-map, unfortunately, does not allow simplification because no groups can be formed. The resulting expression for P is:

$$P=\bar{A}.\bar{B}.\bar{C}.D+\bar{A}.\bar{B}.C.\bar{D}+\bar{A}.B.\bar{C}.\bar{D}+\bar{A}.B.C.D$$
$$+A.\bar{B}.\bar{C}.\bar{D}+A.\bar{B}.C.D+A.B.\bar{C}.D+A.B.C.\bar{D}$$

By grouping the red terms together and the blue terms together, we can simplify the expression:

$$P=(\bar{A}.\bar{B}+A.B).(\bar{C}.D+C.\bar{D})$$
$$+(A.\bar{B}+\bar{A}.B).(\bar{C}.\bar{D}+C.D)$$

Now, by referring to the expressions for an XOR gate that we introduced at the start of this article, we can further simplify the expression to get:

$$P=(\overline{A\oplus B}).(C\oplus D)+(A\oplus B).(\overline{C\oplus D})$$

By considering each bracketed term as an input to an XOR gate, the expression may be further reduced:

$$P=(A\oplus B)\oplus(C\oplus D)$$

and the circuit formed is a 'cascade' of XOR gates: