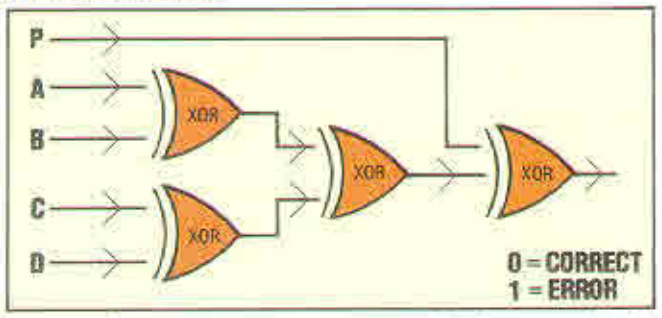
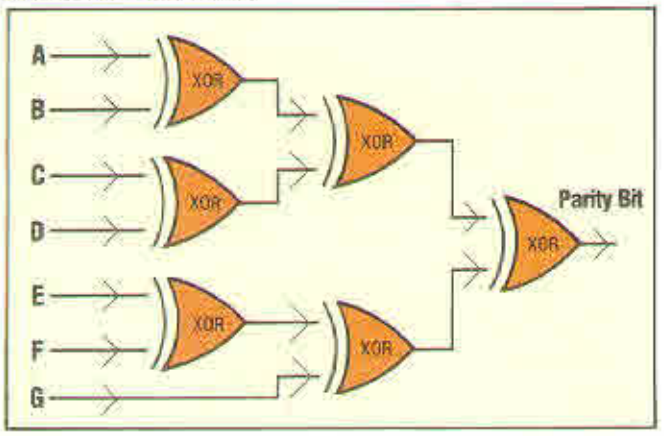
This circuit can be modified to act as a parity checker by simply adding another XOR gate to compare the received parity bit with one generated by the circuit at the receiving end.

Five-bit Received Code



In practice, most computers use the international standard ASCII code for data transmission. This is an eight-bit code with seven information bits and one even parity bit. It is easy to envisage a parity bit generator for ASCII codes:

Seven-bit Information Code



A PRIORITY ENCODER

Many computers use 'priority interrupt' systems to control the flow of data to and from peripheral devices. In these systems the CPU's operation is interrupted by a signal from the peripheral when it needs the CPU's attention. Where two or more peripherals interrupt the CPU at the same time, however, an order of priority must be followed for the CPU to 'service' the most important device first. The priority encoder that we shall design will link four peripheral devices into a circuit that can identify which peripheral is signalling, and will operate a priority system in the case of simultaneous signals from two or more devices.

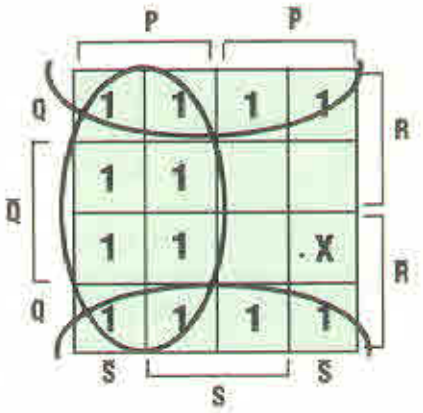
In order to output information specifying one of four devices, two output lines are required. In addition, a third output line will be used to signal that an interrupt is required. Let the four peripherals be P, Q, R and S; P having the highest priority and S having least. The output lines will be called A and B, to identify the peripheral, and Z to signal that an interrupt is required. The truth table for the encoder can be made using an X for conditions that the encoder 'doesn't care' about.

P	Q	R	S	A	B	Z
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

To help you to understand how this table is made up, look at the final row. In this case P is signalling an interrupt and as P is the device with highest priority we do not care whether or not the lower priority devices are signalling as well.

The three output lines from the circuit must be analysed independently. Starting with A, the k-map is:

For A



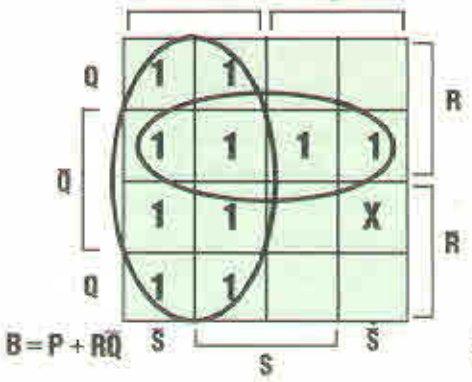
A = P+Q

The 'don't care' case on the output side for A is represented on the k-map as an X, but the 'don't care' cases on the input side are dealt with differently. Take the case where P is one and Q, R and S are 'don't cares'. Here we must fill in all the boxes on the k-map where P is one — there are eight in all. From the k-map we get the simplified expression:

$$A = P + Q$$

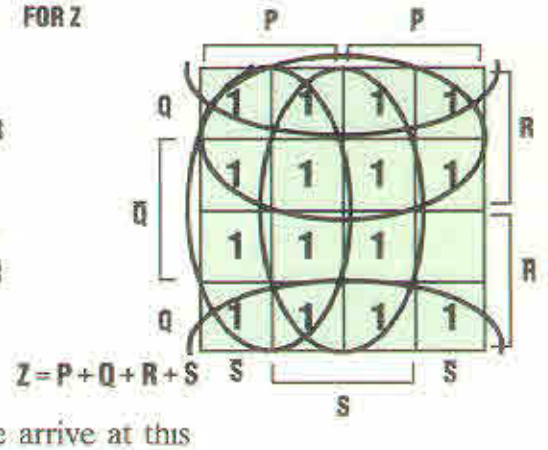
Similarly for B and Z, the k-maps are:

FOR B



B = P + RQ̄

FOR Z



Z = P+Q+R+S

Using these three expressions, we arrive at this circuit design: