$(0,1,0,0,0,1,1) = \underline{d} \leftarrow$  $p(x) = x_0 + x_2 + x$ Transmitted codeword:

## EICHKE 3

the circuit. final remainder is similarly steps 4 step 3, correspond quotient term and This corresponds contains a "1," a p the example, cont nomial at any give of the register con the same division three columns in the corresponds to the the highest order c dend polynomial: circuit for the ger

to determine where The same div

## sbnate 2.9.8

been used in the E of these generato in ATM networks modem standard. LAN standards at used in the HDL protocol for contr The CRC-12 and Table 3.7 gives ge:

> $x + z^{x} + y^{x}$ 1101 1010 1101 0111  $\frac{c_x + b_x}{c_x + b_x} + b_x$ 1101 1011) 1100000 0111 Encoding:  $x^3i(x) = x^6 + x^5$  $\xi_X + \xi_X = (x)i \leftarrow (0,0,1,1)$ : nobsernolal CRC encoding. Generator polynomial:  $g(x) = x^3 + x + 1$ EIGURE 3.68 Example of

010

adding the remainder r(x) from  $x^{n-k}i(x)$ : The final step in the encoding procedure obtains the binary codeword b(x) by

$$(\xi \xi.\xi) \qquad (x) \gamma + (x) i^{\lambda - n} x = (x) d$$

the CRC bits. contain the original four information bits and how the lower three positions contains corresponds to the binary codeword (1,1,0,0,0,1,0). Note how the first four positions remainder polynomial r(x) = x. The codeword polynomial is then  $x^6 + x^5 + x$ , which check bits. In the example in Figure 3.68, the division of  $x^3i(x)$  by g(x) gives the the information bits and in which the n-k lower-order terms are the cyclic redundancy encoding process introduces a binary polynomial in which the k higher-order terms are that the upper k positions were occupied by the information bits. We thus see that this ously introduced register of length n, r(x) will occupy the lower n-k positions. Recall degree n - k - 1 or lower. Therefore r(x) has at most n - k terms. In terms of the previ-Because the divisor g(x) had degree n-k, the remainder v(x) can have maximum

divisible by g(x) because that 122 - 17 is evenly divisible by 35. Similarly, the codeword polynomial b(x) is that by subtracting the remainder 17 from both sides, we obtain 122 - 17 = 3(35) so quotient of 3 and a remainder of 17. This result implies that 122 = 3(35) + 17. Note In Figure 3.66 we showed that in normal division dividing 122 by 35 yields a

$$(\forall \xi ' \xi) \qquad (x)b(x)\delta = (x)\lambda + (x)\lambda + (x)b(x)\delta = (x)\lambda + (x)!_{y-u}x = (x)q$$

nonzero, then an error has been detected. the pattern is satisfied by dividing the received polynomial by g(x). If the remainder is pattern that must be checked by the receiver. The receiver can check to see whether implies that all codewords are multiples of the generator polynomial g(x). This is the where we have used the fact that in modulo 2 arithmetic r(x) + r(x) = 0. Equation (3.54)

mined by the coefficients of the generator polynomial. Figure 3.69 shows the division register circuit that implements division. The feedback taps in this circuit are deter-The Euclidean Division algorithm can be implemented using a feedback shift-

## EICHEE 3

the circuit. final remainder is similarly steps 4 step 3, correspond quotient term and This corresponds contains a "1," a p the example, cont nomial at any give of the register con the same division three columns in the corresponds to the the highest order c dend polynomial: circuit for the ger

The same div

## sbnatz 2.9.8

Table 3.7 gives ge. The CRC-12 and protocol for continued in the HDL LAN standards at modem standard. in ATM networks of these generate of these generate

FIGURE 3.68 Example of CRC encoding.

Generator polynomial: 
$$g(x) = x^3 + x + 1$$

Information:  $(0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0) + (1,0,0,0,1,0,1,0) + (1,0,0,0,1,0,1,0) + (1,0,0,0,1,0,1,0) + (1,0,0,0,1,0,1,0) + (1,0,0,0,1,0,1,0) + (1,$ 

The final step in the encoding procedure obtains the binary codeword b(x) by adding the remainder r(x) from  $x^{n-k}i(x)$ :

$$(\xi \xi.\xi) \qquad (x) r + (x) i^{\lambda - n} x = (x) d$$

Because the divisor g(x) had degree n - k, the remainder v(x) can have maximum degree n - k - 1 or lower. Therefore v(x) has at most n - k terms. In terms of the previously introduced register of length n, v(x) will occupy the lower n - k positions. Recall that the upper k positions were occupied by the information bits. We thus see that this encoding process introduces a binary polynomial in which the k higher-order terms are the cyclic redundancy the information bits and in which the n - k lower-order terms are the cyclic redundancy check bits. In the example in Figure 3.68, the division of  $x^3i(x)$  by g(x) gives the remainder polynomial v(x) = x. The codeword polynomial is then  $x^6 + x^5 + x$ , which corresponds to the binary codeword (1,1,0,0,0,1,0). Note how the first four positions contains the original four information bits and how the lower three positions contains the original four information bits and how the lower three positions contains the ORC bits.

In Figure 3.66 we showed that in normal division dividing 122 by 35 yields a quotient of 3 and a remainder of 17. This result implies that 122 = 3(35) + 17. Note that by subtracting the remainder 17 from both sides, we obtain 122 - 17 = 3(35) so that 122 - 17 is evenly divisible by 35. Similarly, the codeword polynomial b(x) is divisible by g(x) because

where we have used the fact that in modulo 2 arithmetic r(x) + r(x) = 0. Equation (3.54) implies that all codewords are multiples of the generator polynomial g(x). This is the pattern that must be checked by the receiver. The receiver can check to see whether the pattern is satisfied by dividing the received polynomial by g(x). If the remainder is nonzero, then an error has been detected.

The Euclidean Division algorithm can be implemented using a feedback shift-register circuit that implements division. The feedback taps in this circuit are determined by the coefficients of the generator polynomial. Figure 3.69 shows the division