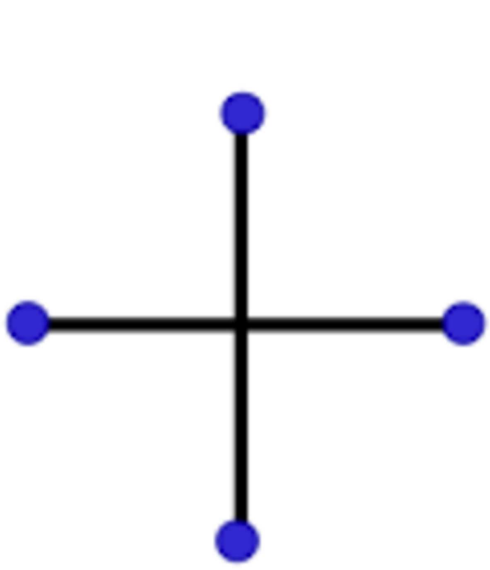
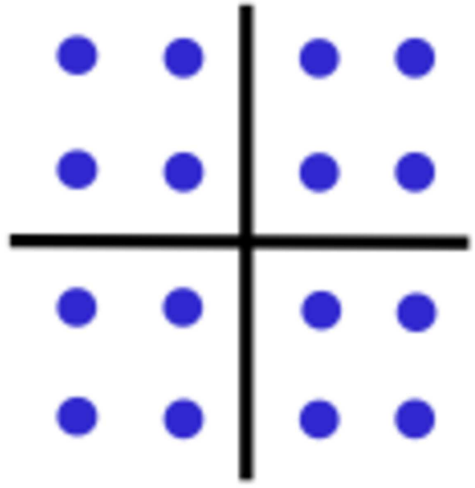
Error Protection: Detection and Correction

- Communication channels are subject to noise.
- Noise distorts analog signals.
- Noise can cause digital signals to be received as different values.
  - Bits can be flipped
  - Points in a signal constellation can be shifted.
- Changes in a digital signal are called *errors*.

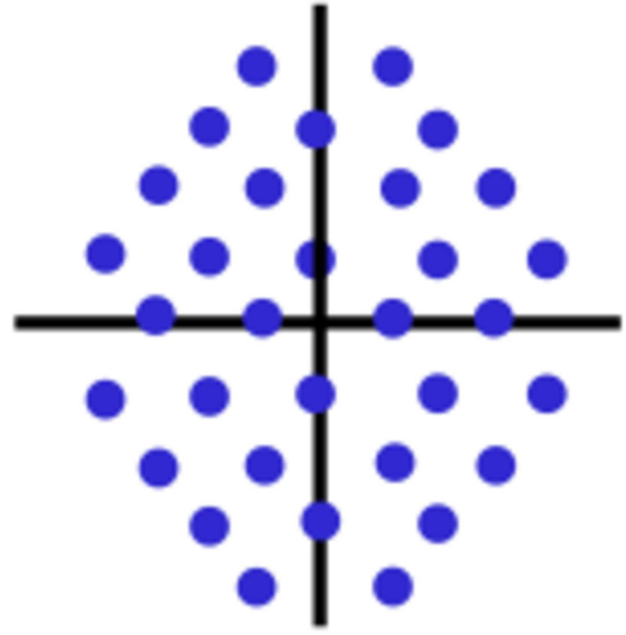
Constellation Examples



V.22
600 baud 1200 bps
PSK



V.22bis
600 baud 2400 bps
QAM

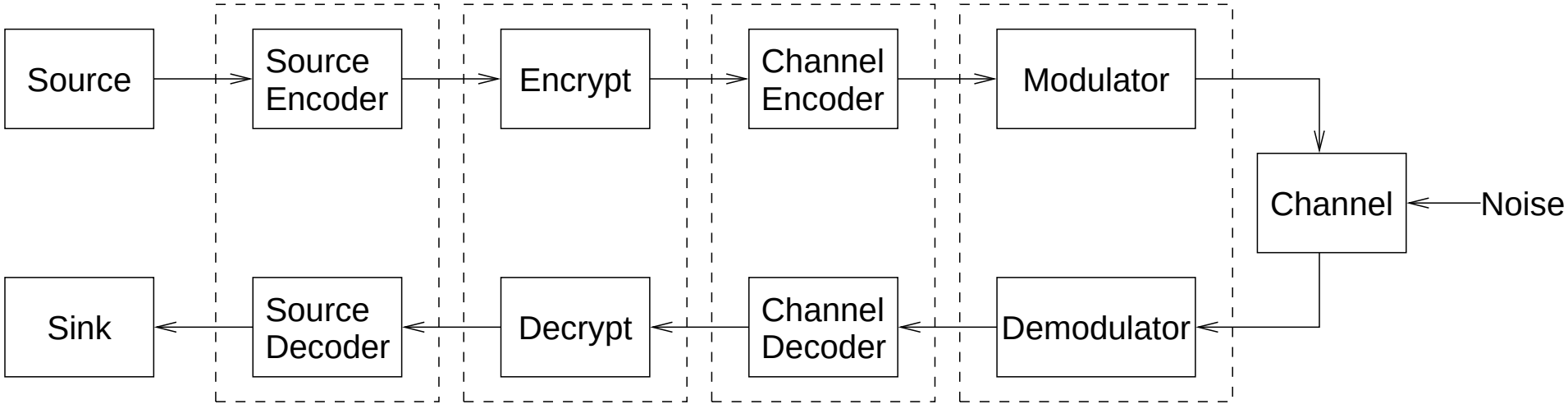


V.32
2400 baud 9600 bps
TCM

- baud = symbol per second
- baud "rate" is proportional to bandwidth

Communication Systems

Recall the communication system block diagram:



We have concentrated on the modulator/demodulator blocks.

Error protection involves channel encoder and decoder.

Error Control Classification

The error control problem can be classified in several ways.

- Type of errors: how much clustering — random, burst, catastrophic
- Type of modulator output: digital ("hard") vs. analog ("soft")
- Type of error control coding: detection vs. correction
- Type of codes: block vs. convolutional
- Type of decoder: algebraic vs. probabilistic

The first two classifications are used to select a coding scheme according to the last three classifications.

Types of Error Potection

- Error detection

  Goal: avoid accepting faulty data.

  Lost data may be unfortunate; wrong data may be disastrous.

  Solution: *checksums* are included in messages (packets, frames, sectors).

  If any part of the message is altered, then the checksum is *not* valid (with high probability).

- (Forward) error correction (FEC or ECC).

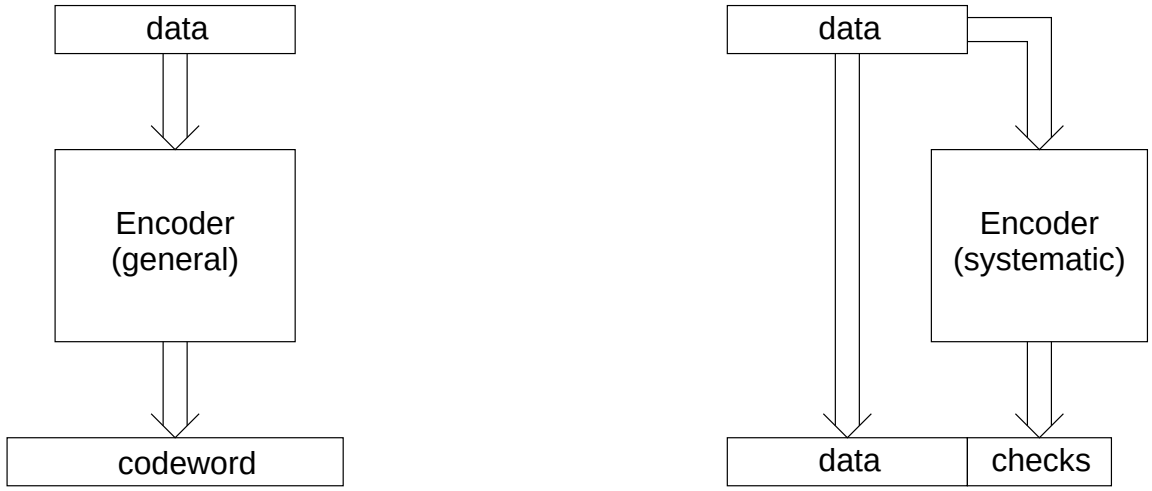
  Use redundancy in encoded message to estimate from the received data (*senseword*) what message was actually sent.

  Optimal estimate is message that is most probable given what is received (MAP, *maximum a posteriori*). The best estimate is typically the message that is "closest" to the senseword.

Types of Error Protecting Codes

- Block codes

  Data is *blocked* into k-vectors of information digits, then encoded into n-digit codewords ($n \geq k$) by adding $p = n - k$ redundant check digits.



  There is no memory betwen blocks. The encoding of each data block is independent of past and future blocks.

  An encoding in which the information digits appear unchanged in the codewords is called *systematic*.

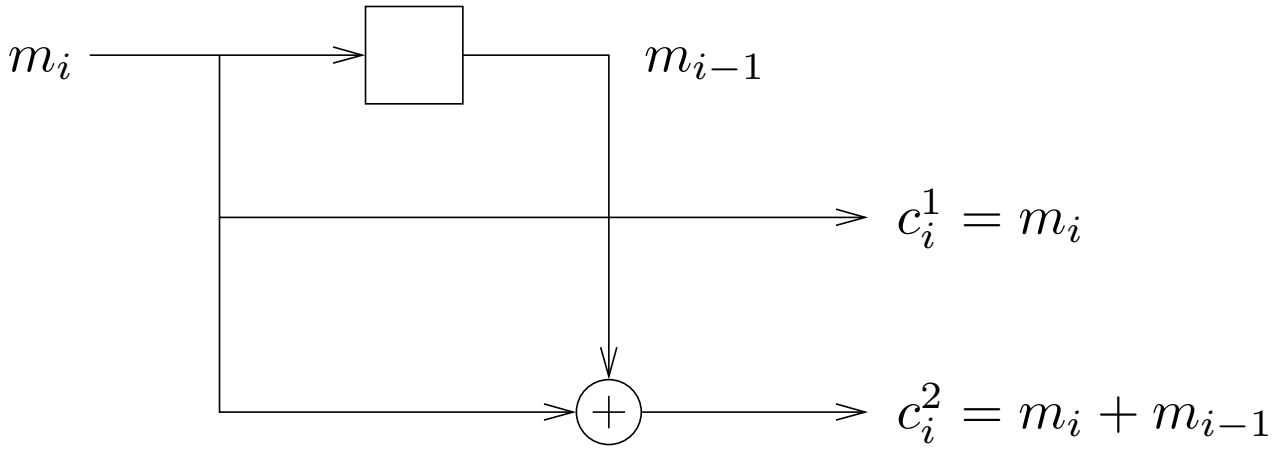
Types of Error Protecting Codes (cont.)

- Convolutional codes

  Time-invariant encoding scheme: each n-bit codeword block depends on current information digits and on the past m n-bit information blocks

  The parameter m is called the *memory order*. The *constraint length* is $(m+1)n$; this is the number of bits that the decoder must consider.

  Here is the simplest convolutional code.



  For this rate $1/2$ convolutional code, $m = 1$ and $n = 2$.

Error Detection: Simple Parity-Check Codes

Append one check bit to data bits so that all codewords have the same overall parity — either even or odd.

Even-parity codewords are defined by a single *parity-check equation*:

$$c_1 \oplus c_2 \oplus \cdots \oplus c_n \;=\; (c_1 + c_2 + \cdots + c_n) \bmod 2 \;=\; 0\,,$$

where $\oplus$ denotes the exclusive-or operation.

If we XOR c_n to both sides of the above equation, we obtain an *encoding equation*:

$$c_n \;=\; c_1 \oplus c_2 \oplus \cdots \oplus c_{n-1}\,.$$

This shows how to compute the *check bit* c_n from the *data bits* $c_1, \ldots, c_{n-1}$.

Any single bit error (or any odd number of errors) can be detected.

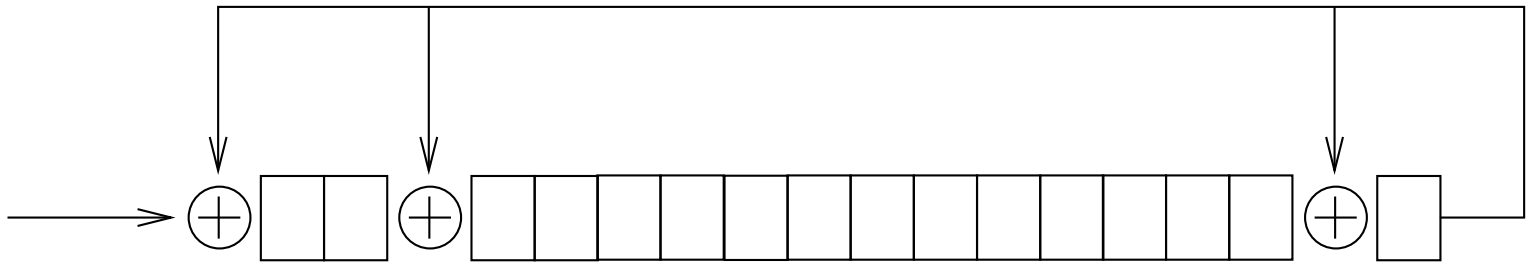
Any bit c_i can be considered to be the check bit because it can be computed from the other $n-1$ bits: $c_i = \sum_{j \neq i} c_j$.

Cyclic Redundancy Check (CRC)

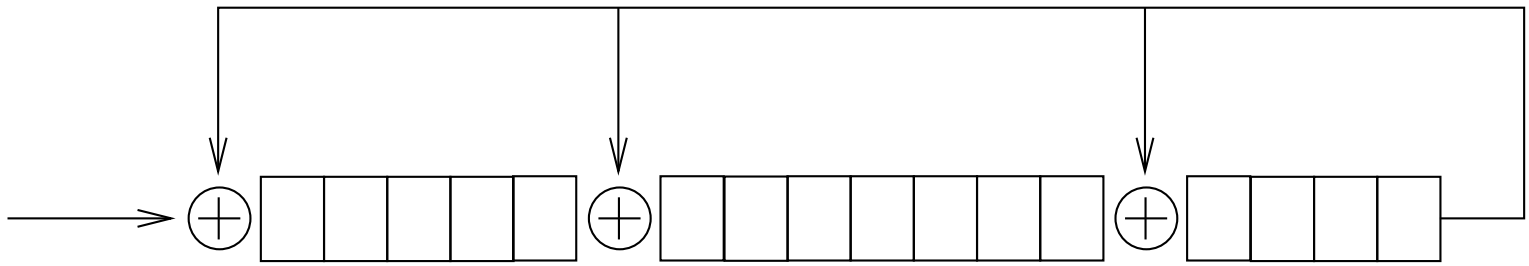
To detect more than one error (guaranteed), we need more equations.

One method to define more equations is by polynomial division, which can be implemented using linear feedback shift registers.

CRC-16:



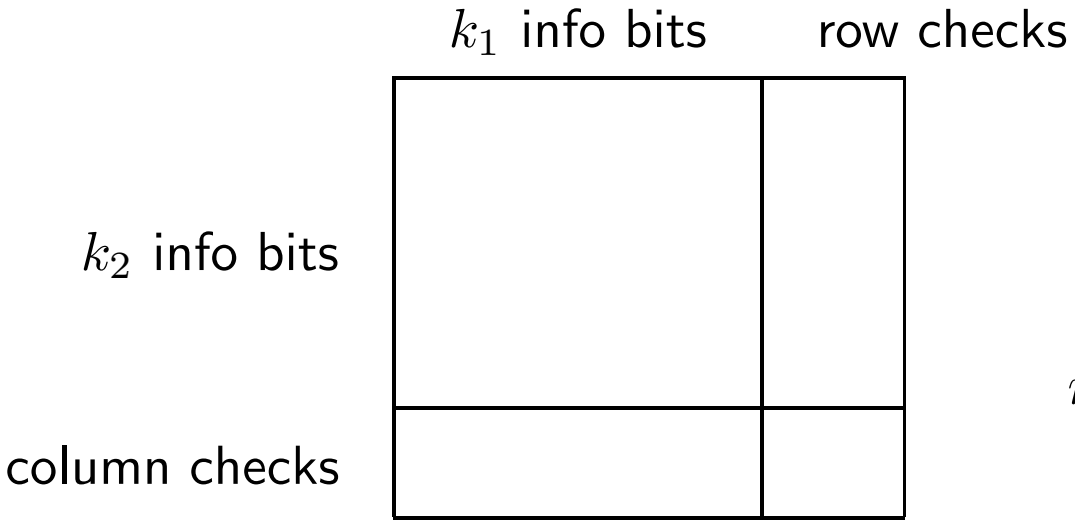
CRC-CCITT:



Both codes can detect two errors in $2^{15} - 1 = 32767$ bits.

Error Correction: Simple Product Codes

Arrange data bits in a two-dimensional array. Append parity check bits at the end of each row and column.



$$\begin{aligned} n &= (k_1+1)(k_2+1) \\ k &= k_1 k_2 \\ n-k &= k_1 + k_2 + 1 \end{aligned}$$

Single error causes failure of one row equation and one column equation. Incorrect bit is located at the intersection of the bad row and bad column.

Double errors can be detected — two rows or two columns (or both) have the wrong parity — but cannot be corrected.

Some triple errors cause miscorrection. Which?

Error Correction: Hamming Codes

Simple product codes are simple but inefficient:

- a failed parity-check equation locates row or column of error
- however, a satisfied equation gives little information

An "efficient" equation gives one bit of information about the error location. It "looks" at half the codeword bits and is "independent" of other equations.

The following table defines a $(7, 4)$ *Hamming* parity-check code.

c_1	c_2	c_3	c_4	c_5	c_6	c_7
1	0	1	0	1	0	1
0	1	1	0	0	1	1
0	0	0	1	1	1	1

} 3 parity-check equations

The 1's indicate which codeword bits affect which parity-check equations.

Hamming Codes: Parity-Check Equations and Matrix

The following three equations are satisfied by all (and only) valid codewords:

$$\begin{aligned} c_1 \oplus c_3 \oplus c_5 \oplus c_7 &= 0 \\ c_2 \oplus c_3 \oplus c_6 \oplus c_7 &= 0 \\ c_4 \oplus c_5 \oplus c_6 \oplus c_7 &= 0 \end{aligned}$$

The check equations can be described by a *parity-check* matrix:

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Codewords are characterized *analytically* by the following equations:

$$H \begin{bmatrix} c_1 \\ \vdots \\ c_7 \end{bmatrix} = 0_{3\times 1} \Leftrightarrow \quad [\, c_1 \ \ldots \ c_7 \,] H^T = 0_{1\times 3}$$

In other words, a 7-tuple $\mathbf{c}$ is a codeword if and only if $\mathbf{c}H^T = 0$.

Hamming Codes: Encoding Equations

Each of the codeword bits c_1, c_2, c_4 appears in only one equation.

Therefore c_1, c_2, c_4 can be computed from the other bits, c_3, c_5, c_6, c_7.

$$
\begin{aligned}
c_1 &= c_3 \oplus c_5 \oplus c_7 \\
c_2 &= c_3 \oplus c_6 \oplus c_7 \\
c_4 &= c_5 \oplus c_6 \oplus c_7
\end{aligned}
$$

These *linear* encoder equations can be written as a vector-matrix product.

$$
[\, c_1 \; c_2 \; c_4 \,] = [\, c_3 \; c_5 \; c_6 \; c_7 \,] P = [\, c_3 \; c_5 \; c_6 \; c_7 \,] \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.
$$

We could choose other sets of check bits, such as $\{c_2, c_3, c_4\}$.

Not all sets work. E.g., c_1, c_2, c_3 cannot be determined from c_4, c_5, c_6, c_7 since the leftmost 3 columns of H form a singular (not invertible) matrix.