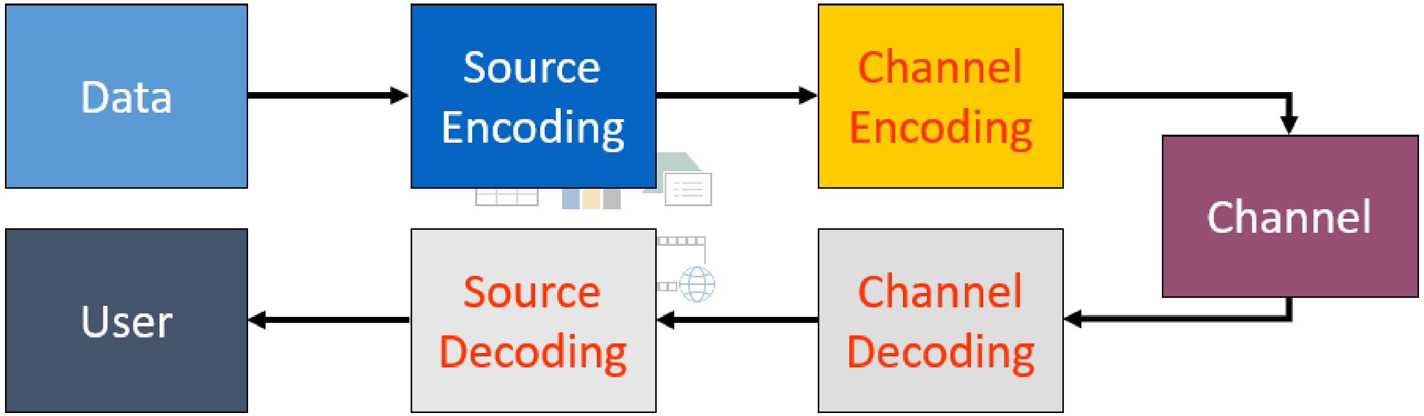
- **Why do we need channel coding for data communication ?**

- Severe transmission problems
  - In terrestrial mobile communications due to: Multipath fading
    - E.g., reflections / diffractions / scattering in cellular wireless communications
  - Low S/N (signal-to-noise) ratio

  - In satellite communications due to: Limited transmitting power for forward channels



- Is there any price paid for redundancy???
- YES: *needs higher bandwidth* to achieve better (lower) bit error rate.
- Increased *complexity* of communication system.

❖ Types of Error Control Codes:

- Error control coding can be categorized into ***two groups*** depending on its ability to detect or correct errors.

- In some systems wherein the probability of *errors* during transmission is so *less* that ***retransmission*** of erroneous frames would be a better choice than to use complicated algorithms at the receiver. → ***ARQ*** (Auto Repeat ReQuest) process.

- However, for some applications wherein the application demands a severe time constraint→ then **no retransmission possible.**

- Thus, receiver need to *detect and correct the errors*.→**forward error correction** (FEC).

- Appropriate for delay sensitive and one-way transmission (e.g., broadcast TV) of data.

- Thus, error correction codes can be *classified* as:

➢ **Block codes**: In block codes, a *k-tuple* binary message block is considered at a time and is encoded as n-tuple codeword.

- Ex: (7, 4) block encoding. 4-input bits → 7-transmitted encoded bits.

➢ **Convolution codes**: Here it uses a *memory*. The *output* encoded sequence depends on → the *current input and previous input* bits. Here a sequence of blocks are considered, instead of a single block.

- The key idea of Error Correction :
  - Transmit enough redundant data to *allow receiver to recover* from errors *all by itself*
    - No sender retransmission required

- Major categories of EC codes (really: forward error correction – FEC)
  - **Linear block codes**
    - Cyclic codes
    - Reed-Solomon codes
  - **Convolutional codes**
    - Turbo codes

- Another classification:

- Linear Codes
- Non – linear codes

- Linear codes have the unique property that when any two code words of linear code are added in modulo -2 adder, a *third code word is produced* *which is also a code*, which is not the case for non-linear codes.

- **Parity codes** to detect errors:
- For 'even' parity the additional bit is: $q = \sum_{i=1}^{k} d_i \pmod 2$
- For 'odd' parity the additional bit is *1-q*
- Even parity

  (i) data=(10110) so, codeword=(101101)

  (ii) data=(11011) so, codeword=(110110)

- That is, the additional bit ensures that there are an 'even' or 'odd' number of '1's in the codeword

- How to decode ?
- To decode
  - Calculate sum of received bits in block (mod 2)
  - **If sum is 0 (1) for even (odd) parity then the dataword is the first *k* bits of the received codeword**
  - Otherwise error ☺
  - Ex: data=(10110) so, codeword=(101101) [even]→ if error, say 100101 then p=1

- Code can detect single errors
- But cannot correct error since the error could be in any bit.

- Used in serial communications

➢What are Linear codes?

- Let A = $[A_1, A_2, A_3, .. A_n]$ and B = $[B_1, B_2, B_3, .. B_n]$ be two codewords of a code.
- The modulo-2 sum of A and B is defined as A $\oplus$B. [Bit-wise $\oplus$]
- Then, we define code C to be linear if the *sum of two codewords is also a codeword* in C .
- **Note**: A linear code C must contain a codeword having all 0's. [A $\oplus$A=0]

- *<u>Example</u>*: let us consider a code C = {0 0 0, 11 1}.
- Here codeword C has 2 codewords: Assume it as A= [0 0 0] and B= [1 1 1].

- Then, we have the followings:
- $A \oplus A$= [0 0 0]
- $B \oplus B$= [0 0 0]
- $A \oplus B$= [1 1 1] Similarly,
- $B \oplus A$= [1 1 1]

- Are all these 4 codewords a part of C ?
  - **If YES**: then C is a *linear code*.

➢ What are Block codes ?

- Consider a message block having k binary digits. These k binary digits will be referred data bits.

- The encoder that generates block codes will transform each data block of k bits into a **code block** having n binary digits → codeword

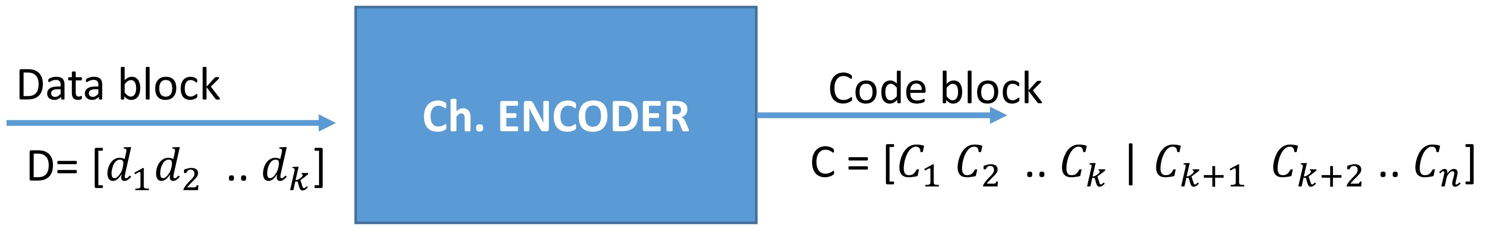
- The redundant $(n - k)$ bits are → check bits. (added extra)

➢What are Linear block codes ?

- **A coding scheme that satisfied the conditions of *both linear and block codes* gives a code known as linear block codes.**

- The message block of k bits must be transformed into a code block of n bits of codeword.

- **Note**: there will be 2^k message combinations, so *all should be transformed* to codewords of size n bits.



- Now, we will represent the **codewords** as follows:
- $C_i = d_i$, $i = 1, 2, .. k$ [First k bits]

$$
\begin{array}{l}
C_1 = 1.d_1 \oplus 0.d_2 \oplus 0.d_2 \oplus \ldots \oplus 0.d_k \\
C_2 = 0.d_1 \oplus 1.d_2 \oplus 0.d_3 \oplus \ldots \oplus 0.d_k \\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
C_k = 0.d_1 \oplus 0.d_2 \oplus 0.d_3 \oplus \ldots \oplus 1.d_k
\end{array}
$$

- And, the remaining $(n - k)$ check bits are expressed as a linear combination of the data bits.

- The remaining $n-k$ bits generated with the help of a parity matrix P

$$
\begin{aligned}
C_{k+1} &= P_{11}d_1 \oplus P_{21}d_2 \oplus P_{31}d_3 \oplus \ldots \oplus P_{k1}d_k \\
C_{k+2} &= P_{12}d_1 \oplus P_{22}d_2 \oplus P_{32}d_3 \oplus \ldots \oplus P_{k2}d_k \\
&\vdots \\
C_n &= P_{1,n-k}d_1 \oplus P_{2,n-k}d_2 \oplus P_{3,n-k}d_3 \oplus \ldots \oplus P_{k,n-k}d_k
\end{aligned}
$$

- Putting the above equations in matrix form, we get

$$[C_1\ C_2 \ldots C_k\ C_{k+1}\ C_{k+2} \ldots C_n] = [d_1\ \ d_2 \ldots d_k] \begin{bmatrix} 1 & 0 \ldots & 0 & P_{11} & P_{12} \ldots & P_{1,\,n-k} \\ 0 & 1 \ldots & 0 & P_{21} & P_{22} \ldots & P_{2,\,n-k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 \ldots & 1 & P_{k1} & P_{k2} \ldots & P_{k,\,n-k} \end{bmatrix}$$

Diagonal matrix from 1st Eqn   By the Parity matrix as in 2nd Eqn

- This can be represented in a general form as **C= D G**
- C= $[C_1, C_2, C_3 \ldots C_k \mid C_{k+1}, C_{k+2}, \ldots C_n]$ //An extended matrix
- D= $[d_1, d_2, \ldots d_k]$
- G= $[I_k \mid P]_{kxn}$ → *Generator matrix*

- Here I_k = Identity matrix : $\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ & & 1 \\ 0 & 0 & \\ & & \dots k \end{matrix}$ → k x k matrix

$$\mathbf{P} = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1,n-k} \\ P_{21} & P_{22} & \cdots & P_{2,n-k} \\ \vdots & \vdots & & \vdots \\ P_{k1} & P_{k2} & \cdots & P_{k,n-k} \end{bmatrix}$$

parity matrix (k x n-k)

- At the **receiver:**→ Parity-check matrix(**H**)→ $\mathbf{H} = [\mathbf{P}^{\mathrm{T}} \vdots \mathbf{I}_{n-k}]_{(n-k) \times n}$

- The purpose of parity-check matrix is to **check (examine)** *whether a codeword is generated by the generator matrix or not.*

- The procedure is, the decoder first computes CH^T.

- If $CH^T = 0$, then the conclusion is C is a **valid codeword.**

- Now, what happens at the receiver/ decoder?
- Let *R* be the received codeword→= the sum of transmitted codeword C and the error E (due to *noise introduced by the channel*).

$$\mathbf{R} = \mathbf{C} \oplus \mathbf{E}$$

- The requirement is that the decoder has to **extract C from R** and then find the message Vector **D from** C. (to get the original data)

- This decoding operation requires the computation of a vector *S* known as syndrome.
- OR

$$\mathbf{S} = \mathbf{R}\ \mathbf{H}^T$$

$$\mathbf{S} = [\mathbf{C} \oplus \mathbf{E}]\ \mathbf{H}^T$$
$$= \mathbf{C}\ \mathbf{H}^T \oplus \mathbf{E}\ \mathbf{H}^T$$

- This finally gives us: $\mathbf{S} = \mathbf{E}\,\mathbf{H}^T$ [*Note: $C\,H^T = 0$, well known proof. we will prove it later*]

- Thus, if S=0, then received codeword R *is* same as C.
- If $S \neq 0$, then C is corrupted by Noise.

- **Example**: let us consider a (3, 6) linear block code generated by the generator matrix given below. Also let the message $D = [0\ 0\ 1]$

$$\mathbf{G} = [\mathbf{I}_3 \mid \mathbf{P}]$$

$$= \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

- **Solution**: First, from the given data, find C= D G

- $$[C_1\ C_2\ C_3\ C_4\ C_5\ C_6] = [0 \quad 0 \quad 1]\left[\begin{array}{ccc:ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

$$C_1 = 0\times 1 \oplus 0\times 0 \oplus 1\times 0 = 0$$

$$C_2 = 0\times 0 \oplus 0\times 1 \oplus 1\times 0 = 0$$

$$C_3 = 0\times 0 \oplus 0\times 0 \oplus 1\times 1 = 1$$

$$C_4 = 0\times 1 \oplus 0\times 1 \oplus 1\times 0 = 0$$

$$C_5 = 0\times 1 \oplus 0\times 1 \oplus 1\times 1 = 1$$

$$C_6 = 0\times 0 \oplus 0\times 1 \oplus 1\times 1 = 1$$

$$\boxed{\mathbf{C} = [0\ 0\ 1\ 0\ 1\ 1]}$$

- If some error introduced by the channel, say $\mathbf{E} = [0\ 1\ 0\ 0\ 0\ 0]$
- Then, the received vector (at decoder) is: $\mathbf{R} = \mathbf{C} \oplus \mathbf{E}$

$$= [0\ 0\ 1\ 0\ 1\ 1] \oplus [0\ 1\ 0\ 0\ 0\ 0]$$

R= [0 1 1 0 1 1] (Error)

Because the C= [0 0 1 0 1 1]

- Next, we find the parity-check matrix H

- We know $H = [P^T \mid I_{n-k}]_{(n-k) \, X \, n}$ (note: P was given in the question)

$$\mathbf{H} = \left[\begin{array}{ccc:ccc} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

- The transpose of parity-check matrix:

$$\mathbf{H}^{\mathrm{T}} = \left[\begin{array}{ccc} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ \hdashline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]$$

- The H^T will look like →

$$\mathbf{H}^{\mathrm{T}} = \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_{n-k} \end{bmatrix}_{n\times(n-k)}$$

- Now, we need to find :

$$\mathbf{S} = \mathbf{R}\ \mathbf{H}^{\mathrm{T}}$$

$$\mathbf{S} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}_{1\times 3}$$

- This is the error correction stage:
- Since the matrix S is non zero (there is error !), and S is same as **2$^{\mathbf{nd}}$ row** of H^T, receiver understands that *2nd bit is in error.*

- **How it corrects it ?**

- The decoder changes the status of the second bit in R by adding E to R, using modulo-2 arithmetic…

- Thus, the *received* **corrected bit** vector is $R_C = R \oplus E$

$$= [0\ 1\ 1\ 0\ 1\ 1\ ] \oplus [0\ 1\ 0\ 0\ 0\ 0]$$

$$= [0\ 0\ 1\ 0\ 1\ 1]$$ → This is same as C. ☺

- Hence, the *Single error* in the received vector can be corrected, provided all the rows of H^T are distinct.