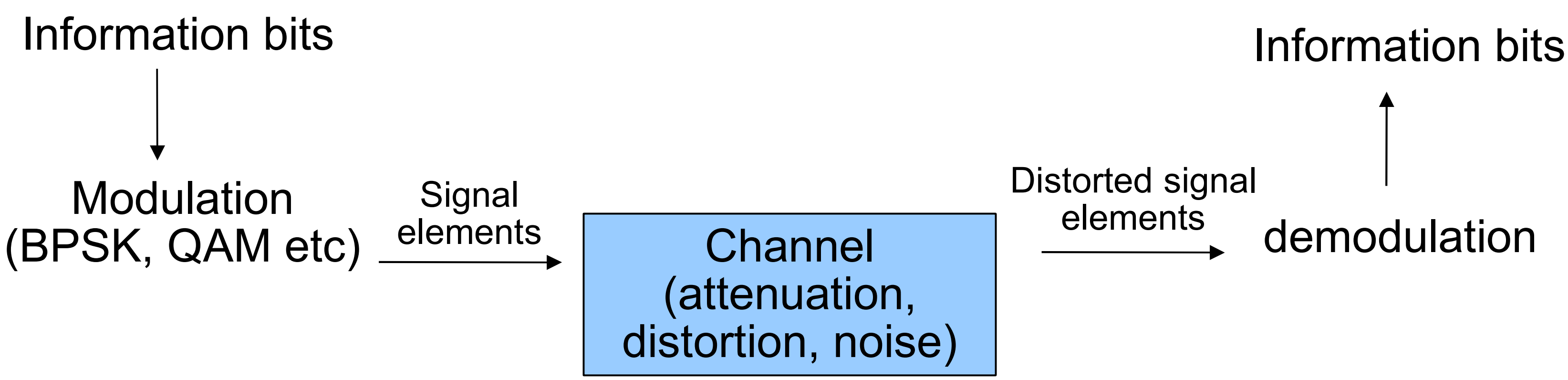
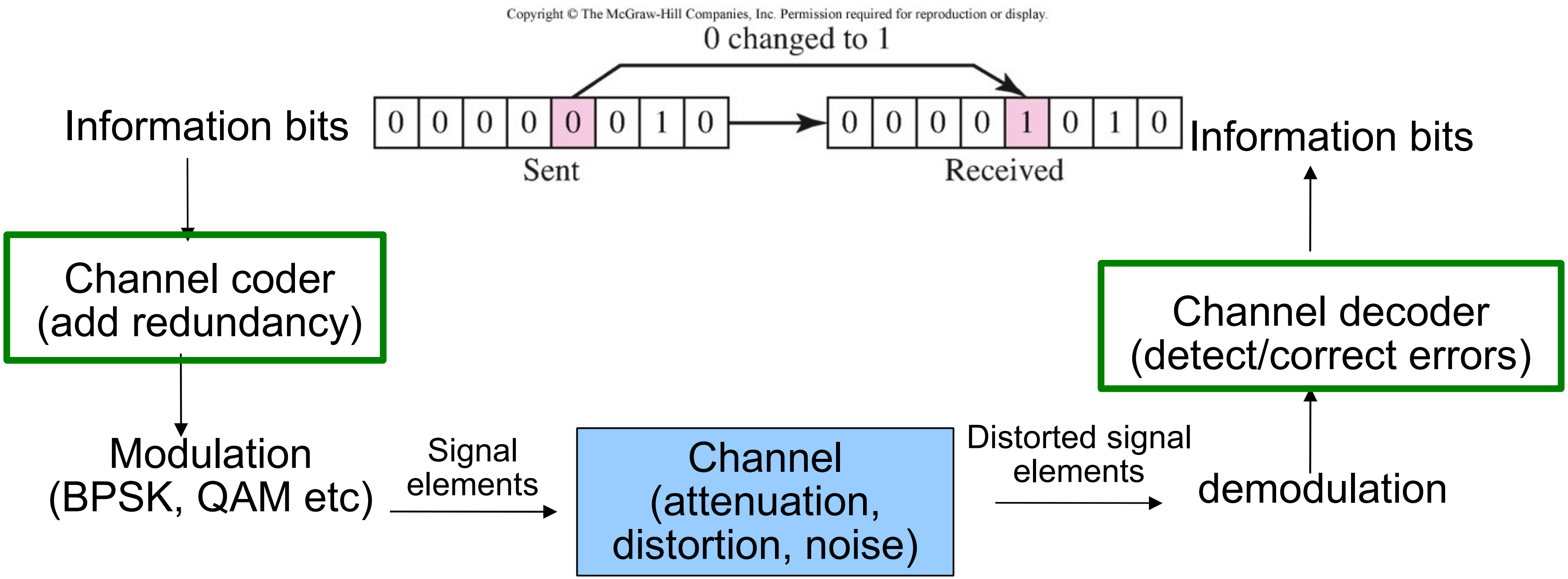
Error Detection and Correction

Bit Errors



- Modulation
  - provides some robustness against errors
  - Cannot guarantee zero error
- Some applications require zero error
  - Examples: ....
- How to detect errors?

Channel Coding

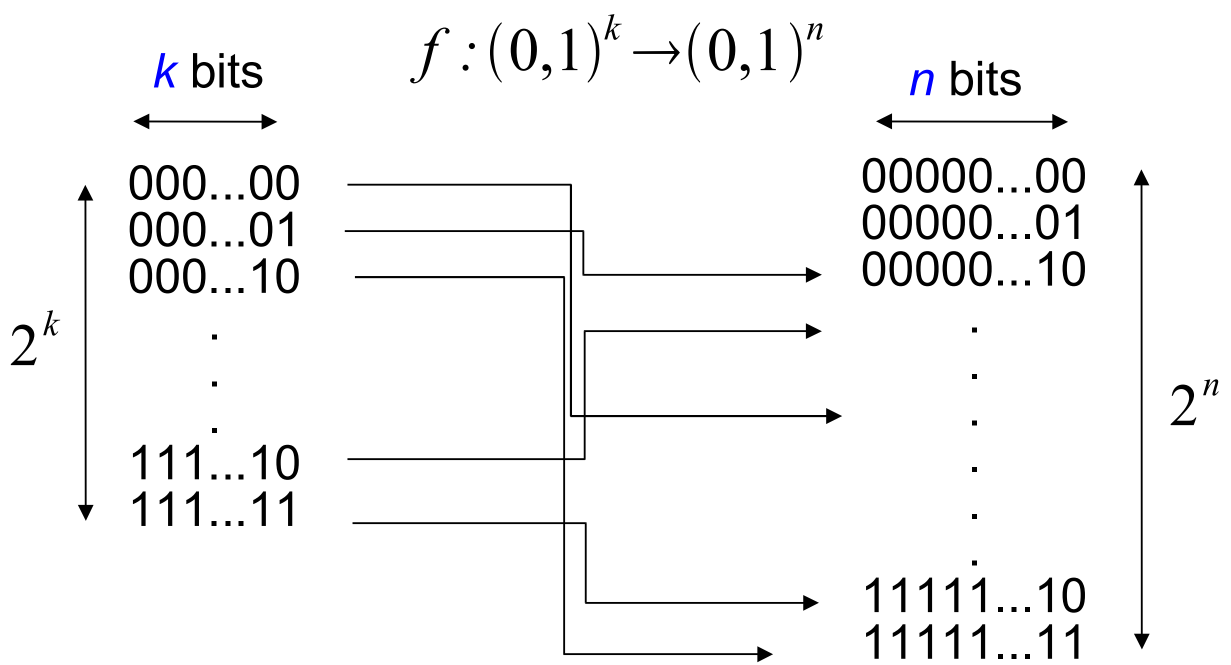


- Channel coder - introduce some redundant bits
  - Sort of signature correlated with information bits
- Channel decoder
  - Check if signature and information match each other

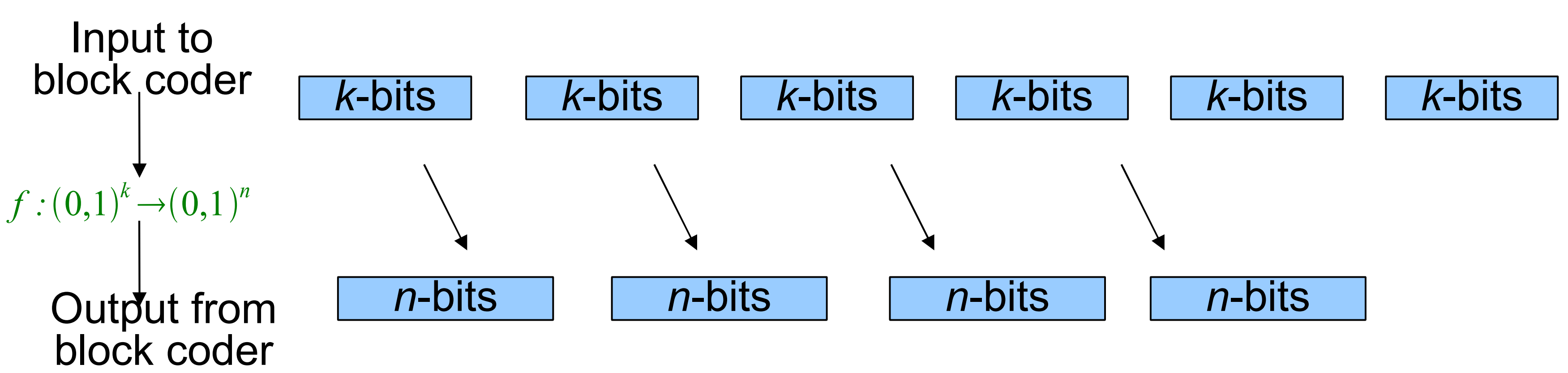
How to Code?

- Ideas?

Block Codes



- Map each *k*-bit word to a unique *n*-bit word
- Split input data stream into blocks of *k*-bits



Design of Block Code

- Is the following a good mapping *f* ?

00	→	0000
01	→	0001
10	→	1000
11	→	1001

Hamming Distance

- Hamming distance $d(x,y)$ between two binary words x and y is the number of differences between corresponding bits

- Examples: $d(000,011)=2$; $d(011,101)=2$;

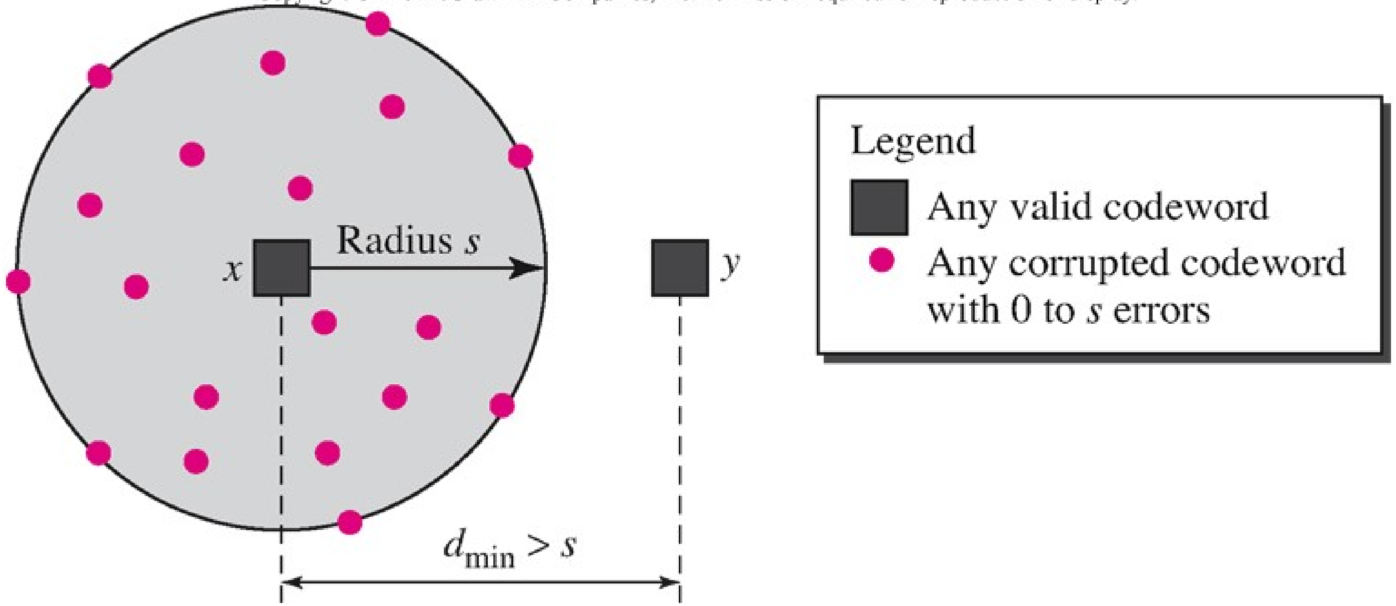
- Minimum Hamming distance of a set of words $\{x_1, x_2, \ldots\}$ is

$$d_{min} = \min_{\substack{i,j \\ i \neq j}} d(x_i, x_j)$$

Detecting Errors

- To guarantee the detection of up to *s* bit errors in all cases

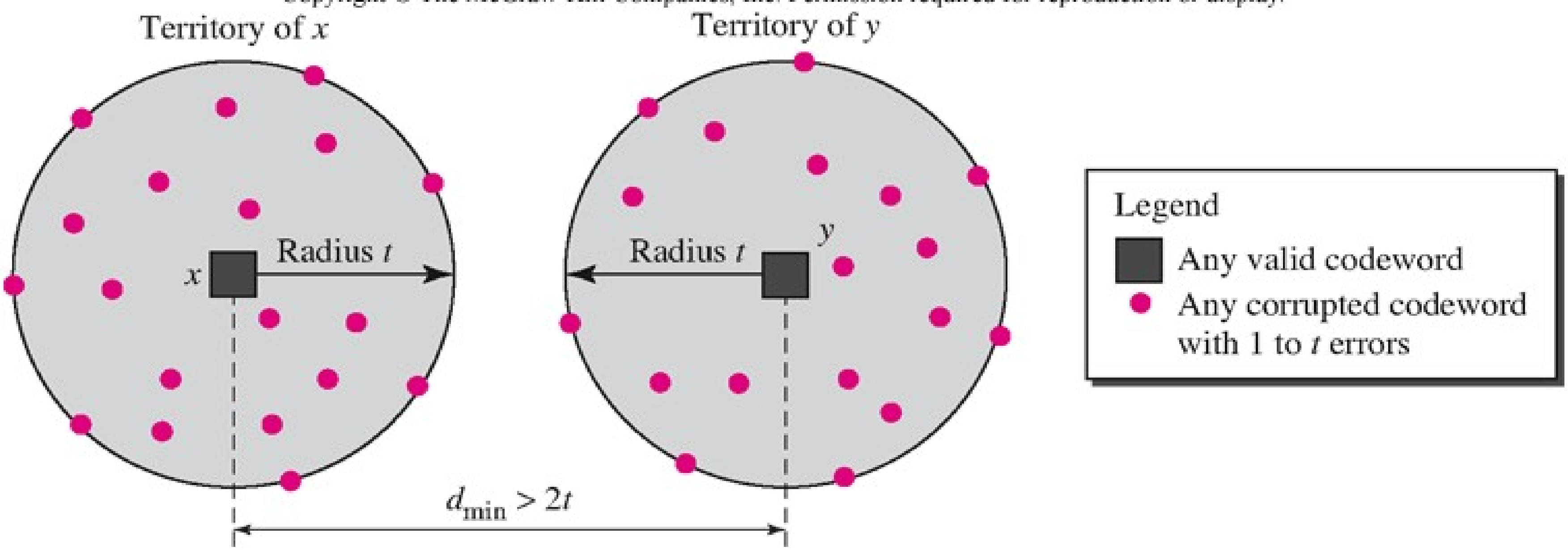
$$d_{min} > s$$



Correcting Errors

- To guarantee correction of up to *t* errors in all cases

$$d_{min} > 2t$$



- How to design good codes?

Repetition Codes

- Simply repeat each bit *n* times

(3,1) repetition code

Input: 10010...

Output: 111000000111000.....

Notation for Block Codes

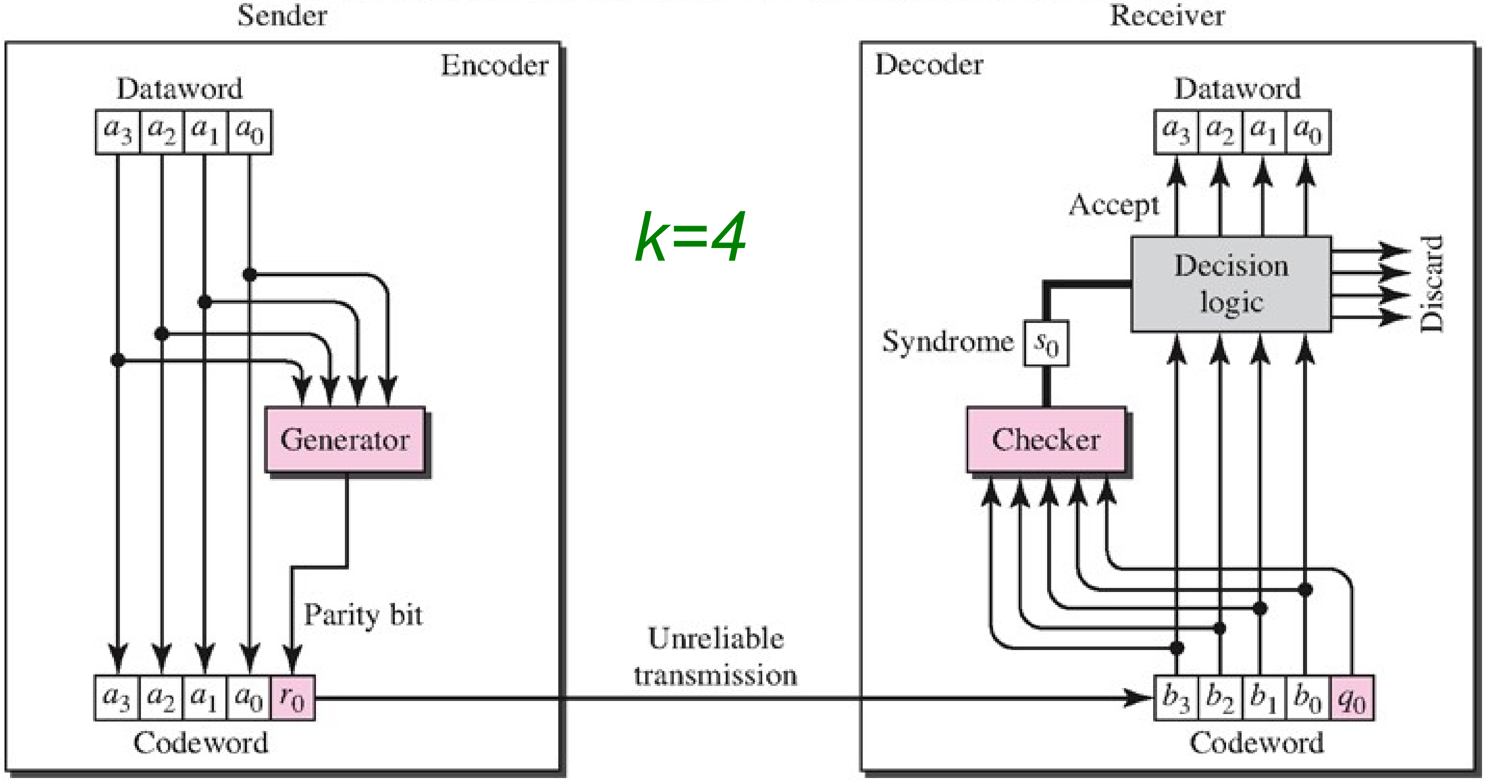
(n,k)

- How many bit errors can we detect?

- How many bit errors can we correct?

Simple Parity Check Code

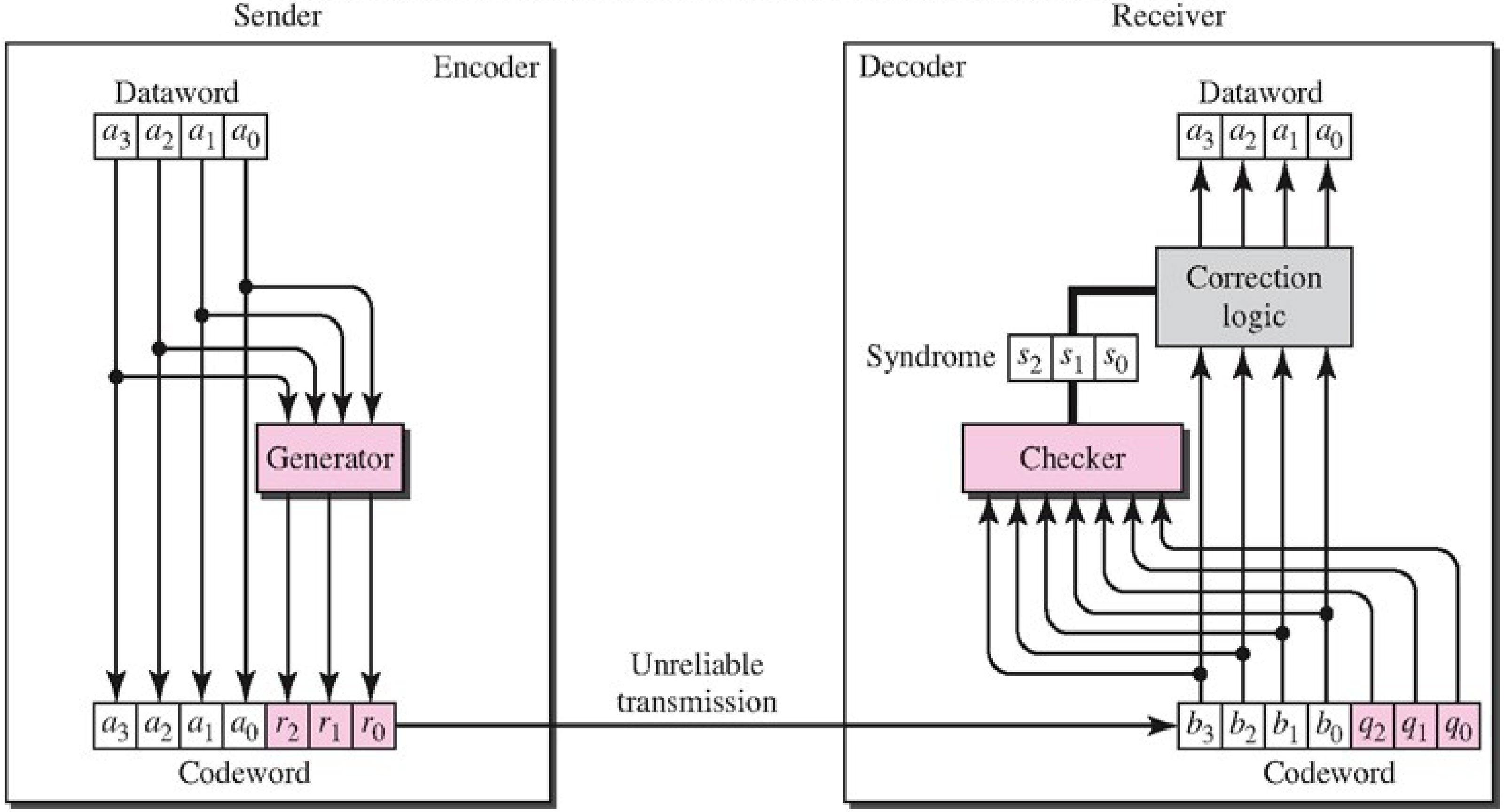
- *n=k+1*
- Add a bit to make total number of 1's even



- How many errors can we detect, correct?

Hamming Codes

- Multiple parity bits; each corresponds to different set of input bits



$$r_0 = a_0 \oplus a_1 \oplus a_2$$
$$r_1 = a_3 \oplus a_1 \oplus a_2$$
$$r_2 = a_0 \oplus a_1 \oplus a_3$$

$$s_0 = b_0 \oplus b_1 \oplus b_2 \oplus q_0$$
$$s_1 = b_3 \oplus b_1 \oplus b_2 \oplus q_1$$
$$s_2 = b_0 \oplus b_1 \oplus b_3 \oplus q_2$$

Hamming Codes: Error Correction

- We can correct 1 bit errors by looking at the syndrome

$$s_0 = b_0 \oplus b_1 \oplus b_2 \oplus q_0$$

$$s_1 = b_3 \oplus b_1 \oplus b_2 \oplus q_1$$

$$s_2 = b_0 \oplus b_1 \oplus b_3 \oplus q_2$$

Syndrome $(s_0 s_1 s_2)$	000	001	010	011	100	101	110	111
Error	-	q_0	q_1	b_2	q_2	b_0	b_3	b_1

Generator and Parity Check Matrices

- Write down generation of codewords and checking of parity in matrix form

$$\underline{x} = {\color{darkred}G}\,\underline{a}$$

Generator matrix (G), input ($\underline{a}$)

$$\underline{s} = {\color{darkblue}H}\,\underline{x}$$

syndrome ($\underline{s}$), Parity check matrix (H)

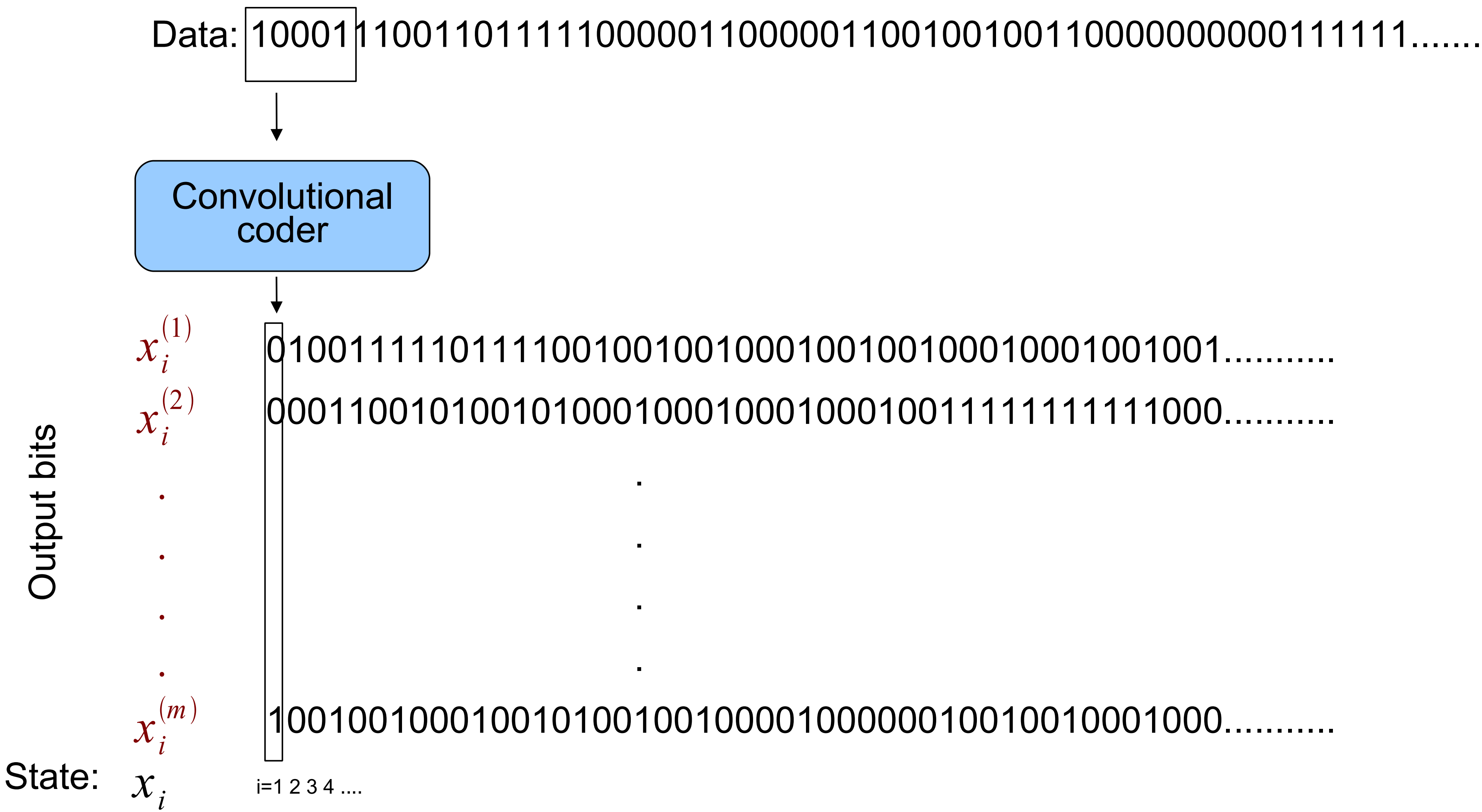
Hamming code (7,4) :

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

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

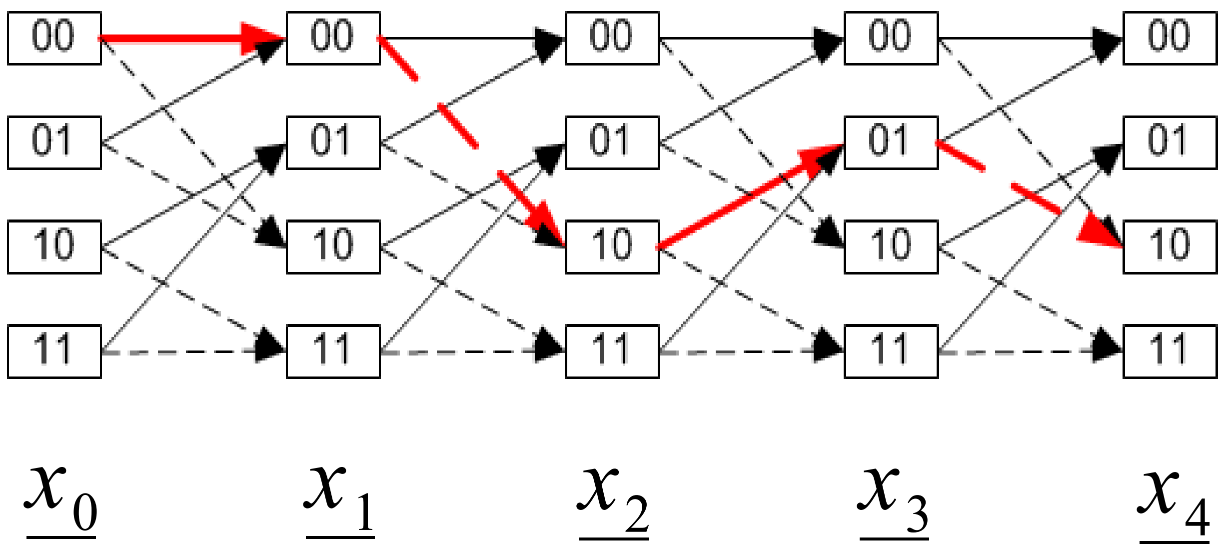
Convolutional Codes

- Do not split data into separate blocks like in block codes
- Compute parity bits over moving window



Trellis

- Correlation between one state and next since generated from overlapping windows of data
- Captured by trellis



- Some paths valid, some not
- If path invalid, find “nearest valid path” using fast algorithm called “Viterbi algorithm”