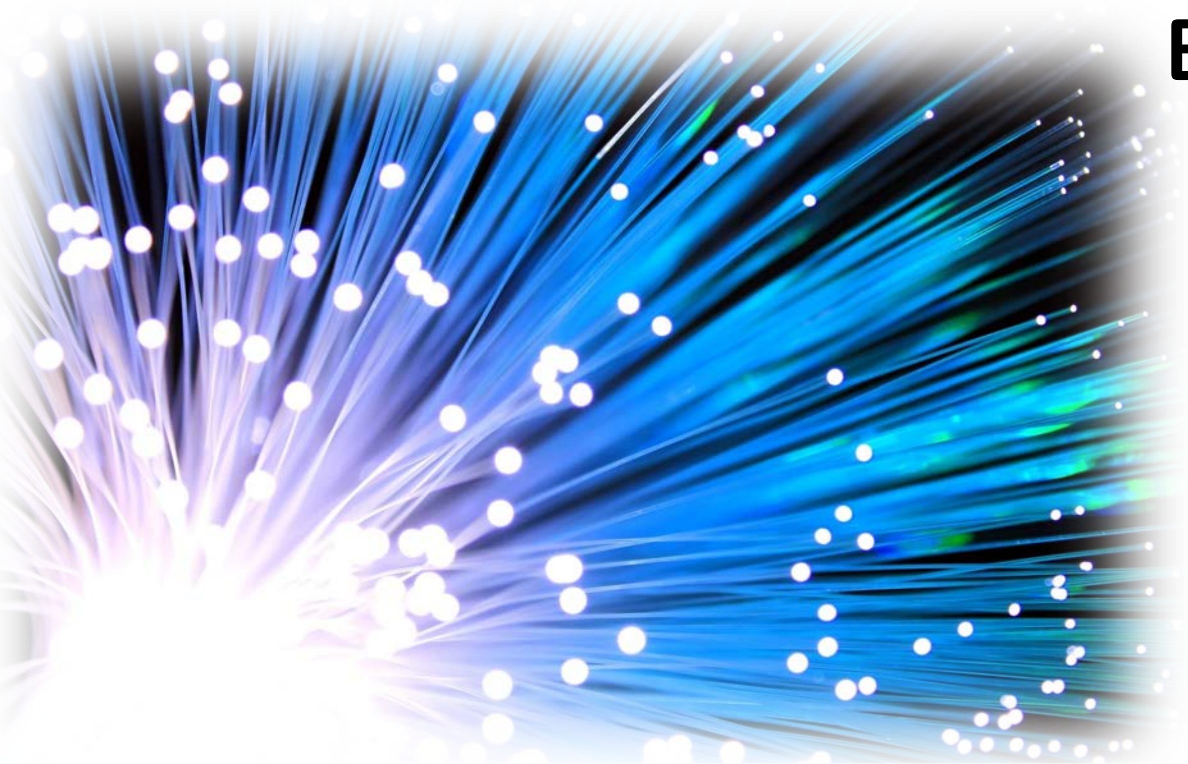
Lecture 3
Digital Transmission Fundamentals

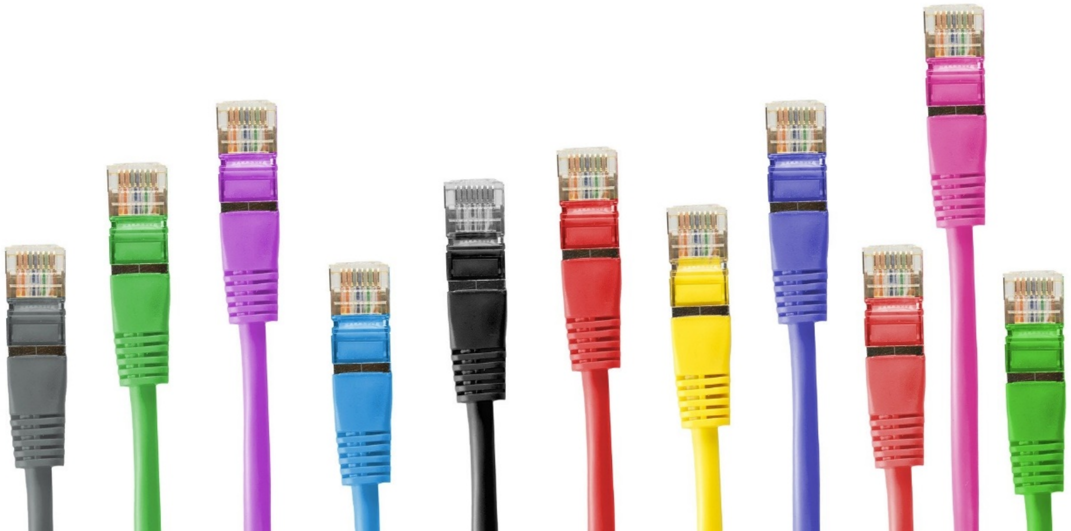


Line Coding

Error Detection and Correction

Lecture 3
Digital Transmission Fundamentals

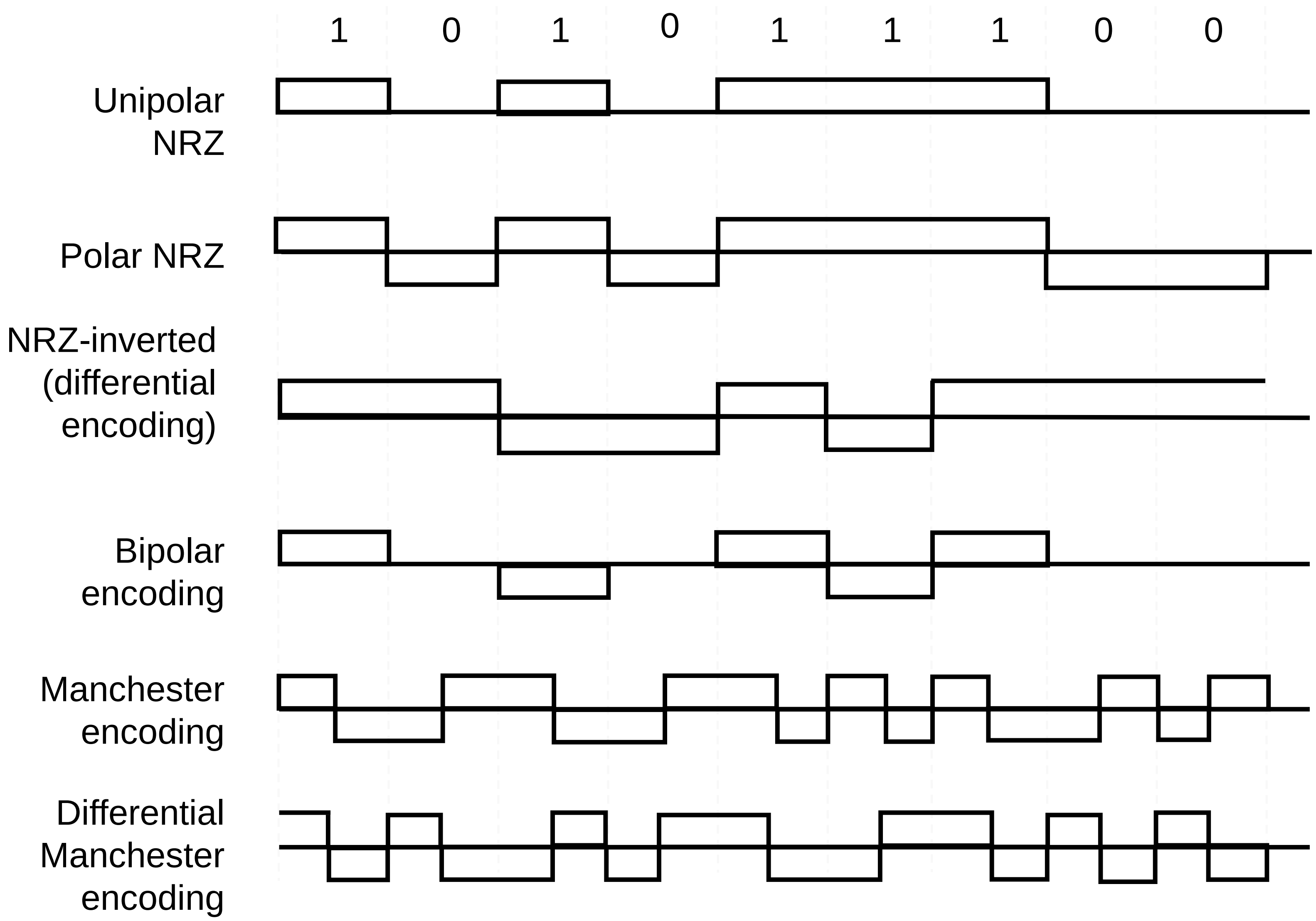
Line Coding



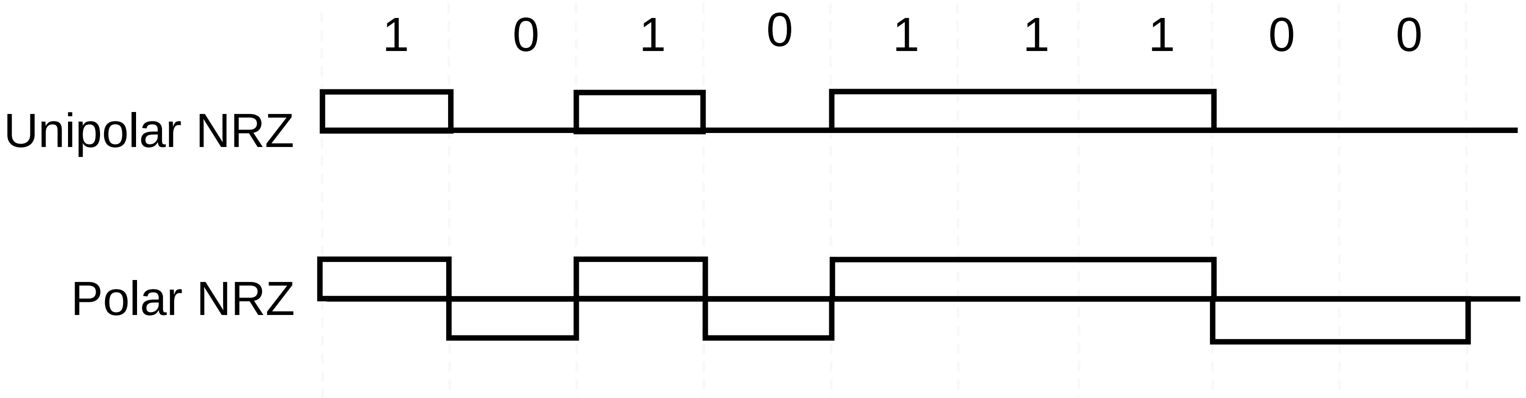
What is Line Coding?

- Mapping of binary information sequence into the digital signal that enters the channel
  - Ex. “1” maps to +A square pulse; “0” to −A pulse

Line coding examples



Unipolar & Polar Non-Return-to-Zero (NRZ)



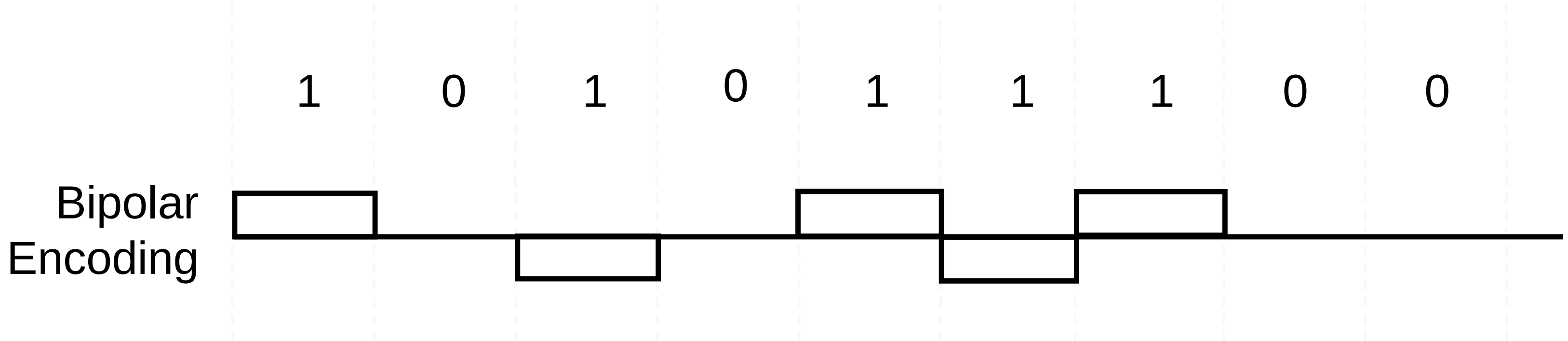
Unipolar NRZ

- "1" maps to +A pulse
- "0" maps to no pulse
- High Average Power
- Long strings of A or 0
  - Poor timing
  - Low-frequency content
- Simple

Polar NRZ

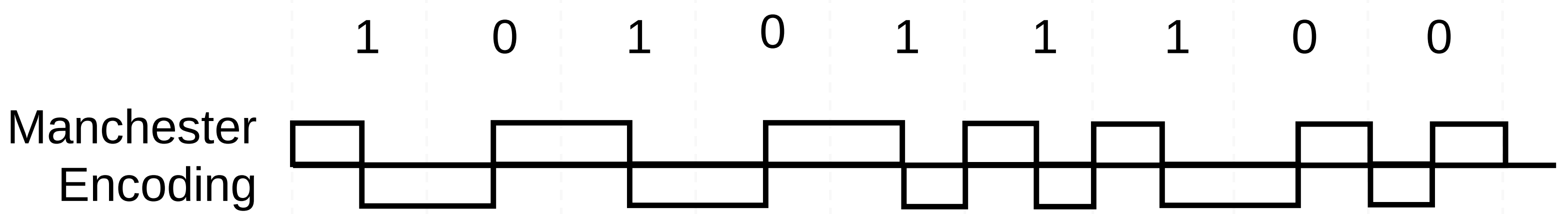
- "1" maps to +A/2 pulse
- "0" maps to –A/2 pulse
- Better Average Power
- Long strings of +A/2 or –A/2
  - Poor timing
  - Low-frequency content
- Simple

Bipolar Code



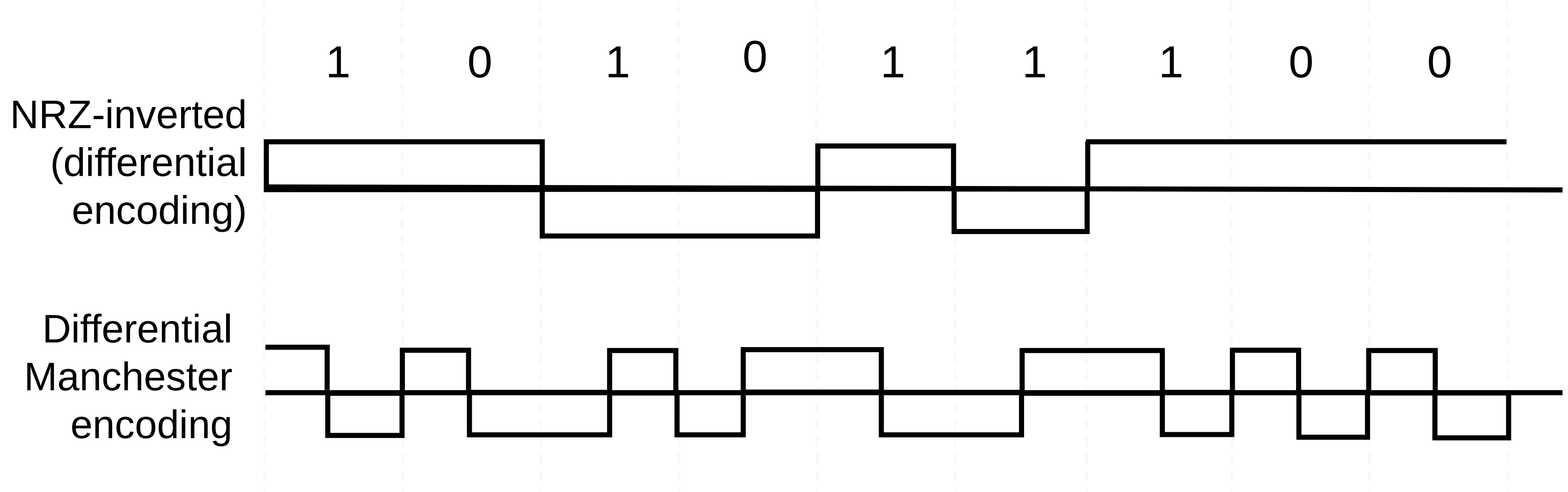
- Three signal levels: {-A, 0, +A}
- "1" maps to +A or –A in alternation
- "0" maps to no pulse
- String of 1s produces a square wave
  - Spectrum centered at $T/2$
- Long string of 0s causes receiver to lose synch
- Zero-substitution codes

Manchester code



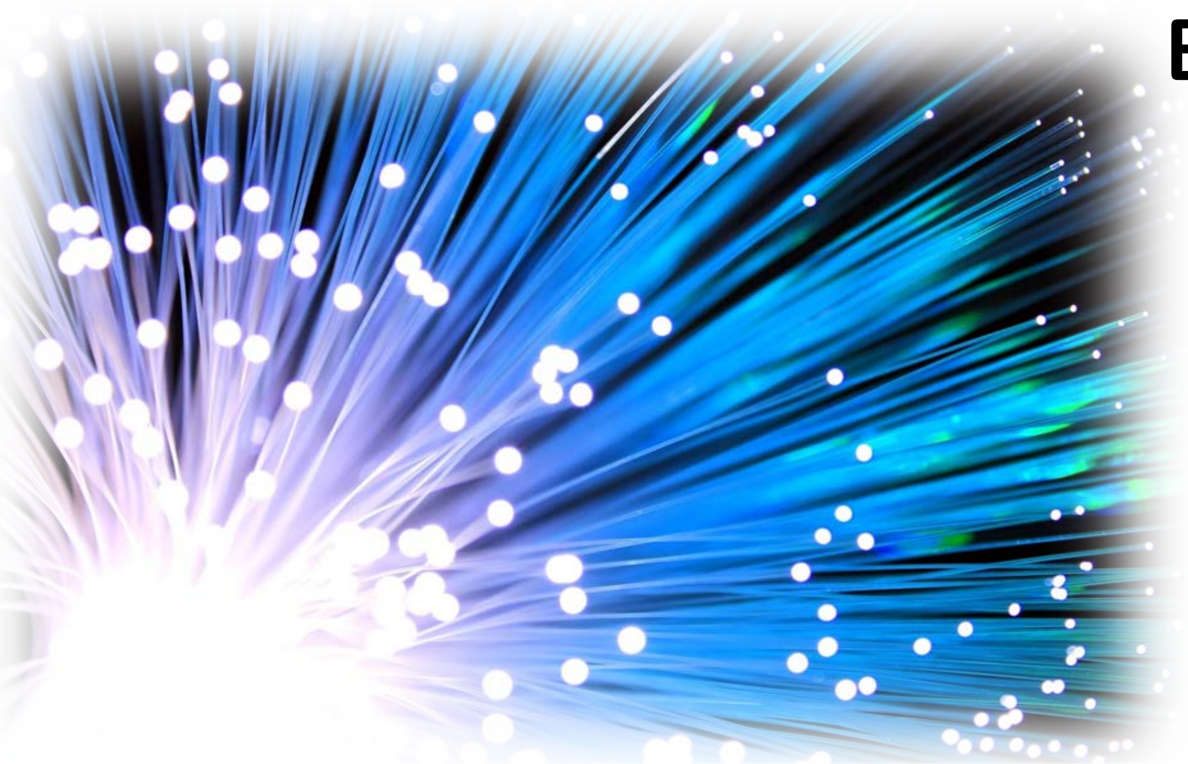
- "1" maps into A/2 first *T*/2, -A/2 last T/2
- "0" maps into -A/2 first *T*/2, A/2 last *T*/2
- Every interval has transition in middle
  - Timing recovery easy
  - Uses double the minimum bandwidth
- Simple to implement
- Used in 10-Mbps Ethernet & other LAN standards

Differential Coding



- “1” mapped into transition in signal level
- “0” mapped into no transition in signal level
- Also used with Manchester coding

Lecture 3
Digital Transmission Fundamentals



Line Coding

Error Detection and Correction

Lecture 3
Digital Transmission Fundamentals

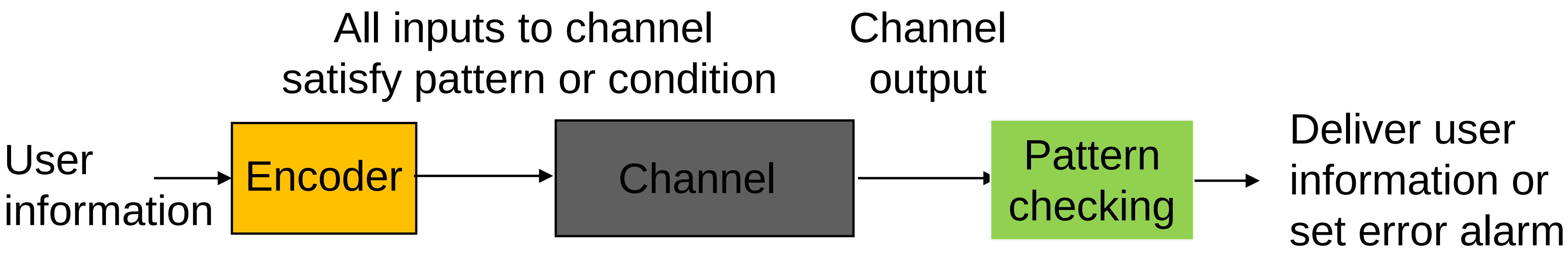
Error Detection and Correction

Error Control

- Digital transmission systems introduce errors
- Applications require certain reliability level
  - Data applications require error-free transfer
  - Voice & video applications tolerate some errors
- Error control used when transmission system does *not* meet application requirement
- Error control ensures a data stream is transmitted to a certain level of accuracy despite errors
- Two basic approaches:
  - Error *detection* & retransmission (ARQ: Automatic Retransmission Request)
  - Forward error *correction* (FEC)

Key Idea

- All transmitted data blocks ("codewords") satisfy a pattern
- If received block doesn't satisfy pattern, it is in error
- Redundancy: Only a subset of all possible blocks can be codewords
- Blindspot: when channel transforms a codeword into another codeword



Single Parity Check

- Append an overall parity check to k information bits

Info Bits: $b_1, b_2, b_3, \ldots, b_k$

Check Bit: $b_{k+1} = b_1 + b_2 + b_3 + \ldots + b_k$ modulo 2

Codeword: $(b_1, b_2, b_3, \ldots, b_k, b_{k+1})$

- All codewords have even # of 1s
- Receiver checks to see if # of 1s is even
  - All error patterns that change an odd # of bits are detectable
  - All even-numbered patterns are undetectable
- Parity bit used in ASCII code

Example of Single Parity Code

- Information (7 bits): (0, 1, 0, 1, 1, 0, 0)

  Parity Bit: $b_8 = 0 + 1 + 0 + 1 + 1 + 0 = 1$

  Codeword (8 bits): (0, 1, 0, 1, 1, 0, 0, 1)

- If single error in bit 3 : (0, 1, 1, 1, 1, 0, 0, 1)
  - # of 1's =5, odd
  - Error detected

- If errors in bits 3 and 5: (0, 1, 1, 1, 0, 0, 0, 1)
  - # of 1's =4, even
  - Error not detected

Two-Dimensional Parity Check

- More parity bits to improve coverage
- Arrange information as columns
- Add single parity bit to each column
- Add a final "parity" column
- Used in early error control systems

1	0	0	1	0	0
0	1	0	0	0	1
1	0	0	1	0	0
1	1	0	1	1	0
1	0	0	1	1	1

Last column consists of check bits for each row

Bottom row consists of check bit for each column

Error-detecting capability

One error

1	0	0	1	0	0
0	0	0	0	0	1
1	0	0	1	0	0
1	1	0	1	1	0
1	0	0	1	1	1

Two errors

1	0	0	1	0	0
0	0	0	0	0	1
1	0	0	1	0	0
1	0	0	1	1	0
1	0	0	1	1	1

1, 2, or 3 errors can always be detected; Not all patterns >4 errors can be detected

Three errors

1	0	0	1	0	0
0	0	0	1	0	1
1	0	0	1	0	0
1	0	0	1	1	0
1	0	0	1	1	1

Four errors (undetectable)

1	0	0	1	0	0
0	0	0	1	0	1
1	0	0	1	0	0
1	0	0	0	1	0
1	0	0	1	1	1

Arrows indicate failed check bits

Other Error Detection Codes

- Many applications require very low error rate
- Need codes that detect the vast majority of errors
- Single parity check codes do not detect enough errors
- Two-dimensional codes require too many check bits
- The following error detecting codes used in practice:
  - CRC Polynomial Codes

Polynomial Codes

- Polynomials instead of vectors for codewords
- Polynomial arithmetic instead of check sums
- Implemented using shift-register circuits
- Also called *cyclic redundancy check (CRC)* codes
- Most data communications standards use polynomial codes for error detection
- Polynomial codes also basis for powerful error-correction methods

Binary Polynomial Division

- **Binary vectors map to polynomials**

$$(i_{k-1}, i_{k-2}, \ldots, i_2, i_1, i_0) \rightarrow i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_2x^2 + i_1x + i_0$$

Addition:

$$(x^7 + x^6 + 1) + (x^6 + x^5) = x^7 + x^6 + x^6 + x^5 + 1$$

$$= x^7 + (1+1)x^6 + x^5 + 1$$

$$= x^7 + x^5 + 1 \quad \text{since } 1+1=0 \text{ mod2}$$

Multiplication:

$$(x + 1)(x^2 + x + 1) = x(x^2 + x + 1) + 1(x^2 + x + 1)$$

$$= x^3 + x^2 + x + (x^2 + x + 1)$$

$$= x^3 + 1$$

Binary Polynomial Division

- **Division with Decimal Numbers**

```
         34      <- quotient
      ______
  35 ) 1222      <- dividend
       105
       ____
        172
        140
       ____
         32      <- remainder
divisor
```

dividend = quotient x divisor +remainder

$1222 = 34 \times 35 + 32$

- **Polynomial Division**

$x^3 + x^2 + x$ = $q(x)$ quotient

divisor: $x^3 + x + 1$) $x^6 + x^5$ (dividend)

$x^6 + x^4 + x^3$

$x^5 + x^4 + x^3$

$x^5 + x^3 + x^2$

$x^4 + x^2$

$x^4 + x^2 + x$

x = $r(x)$ remainder

Note: Degree of r(x) is less than degree of divisor

Polynomial Coding

- Code has binary generator polynomial of degree n−k

$$g(x) = x^{n-k} + g_{n-k-1}x^{n-k-1} + \ldots + g_2x^2 + g_1x + 1$$

- k information bits define polynomial of degree k − 1

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_2x^2 + i_1x + i_0$$

- Find remainder polynomial of at most degree n − k − 1

$$\begin{array}{r} q(x) \\ g(x) \overline{)\, x^{n-k}\, i(x)} \\ r(x) \end{array} \qquad x^{n-k}i(x) = q(x)g(x) + r(x)$$

- Define the codeword polynomial of degree n − 1

$$\underbrace{b(x)}_{n \text{ bits}} = \underbrace{x^{n-k}i(x)}_{k \text{ bits}} + \underbrace{r(x)}_{n-k \text{ bits}}$$

Polynomial example: $k = 4$, $n-k = 3$

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

Information: (1,1,0,0) $i(x) = x^3 + x^2$

Encoding: $x^3 i(x) = x^6 + x^5$

$$
\begin{array}{r l}
 & x^3 + x^2 + x \\
\hline
x^3 + x + 1 \;) & x^6 + x^5 \\
 & x^6 + \phantom{x^5 +{}} x^4 + x^3 \\
\hline
 & x^5 + x^4 + x^3 \\
 & x^5 + \phantom{x^4 +{}} x^3 + x^2 \\
\hline
 & x^4 + \phantom{x^3 +{}} x^2 \\
 & x^4 + \phantom{x^3 +{}} x^2 + x \\
\hline
 & x
\end{array}
\qquad
\begin{array}{r l}
 & 1110 \\
\hline
1011 \;) & 1100000 \\
 & 1011 \\
\hline
 & 1110 \\
 & 1011 \\
\hline
 & 1010 \\
 & 1011 \\
\hline
 & 010
\end{array}
$$

Transmitted codeword:

$b(x) = x^6 + x^5 + x$

$\Longrightarrow$ $\underline{b} = (1,1,0,0,0,1,0)$

The *Pattern* in Polynomial Coding

- All codewords satisfy the following pattern:

$$b(x) = x^{n-k}i(x) + r(x) = q(x)g(x) + r(x) + r(x) = q(x)g(x)$$

- All codewords are a multiple of g(x)!
- Receiver should divide received n-tuple by g(x) and check if remainder is zero
- If remainder is nonzero, then received n-tuple is not a codeword

Standard Generator Polynomials

CRC = cyclic redundancy check

- **CRC-8:**

$= x^8 + x^2 + x + 1$ ATM

- **CRC-16:**

$= x^{16} + x^{15} + x^2 + 1$ Bisync

$= (x + 1)(x^{15} + x + 1)$

- **CCITT-16:**

$= x^{16} + x^{12} + x^5 + 1$ HDLC, XMODEM, V.41

- **CCITT-32:**

IEEE 802, DoD, V.42

$= x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$