CSE3213 Computer Network I

Chapter 3.7 and 3.9

Digital Transmission Fundamentals

Course page:

http://www.cse.yorku.ca/course/3213

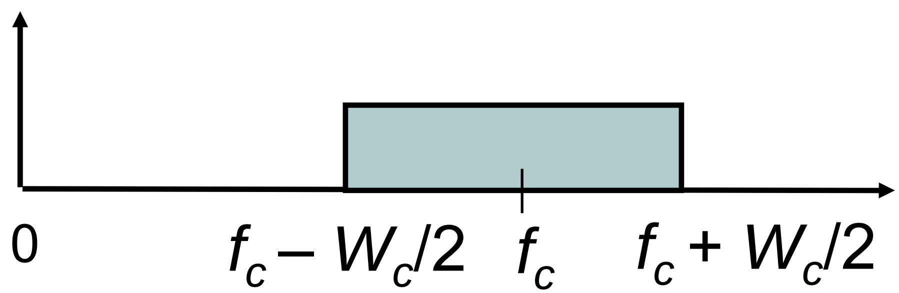
Slides modified from Alberto Leon-Garcia and Indra Widjaja

Modem and Digital Modulation

Data Encoding

- Digital Data, Digital Signals
  - NRZ, Bipolar, Manchester, etc.
- *Digital Data, Analog Signals*
  - *ASK, FSK, PSK, etc.*
- Analog Data, Digital Signals
  - PCM
- Analog Data, Analog Signals
  - AM, FM, PM

Bandpass Channels

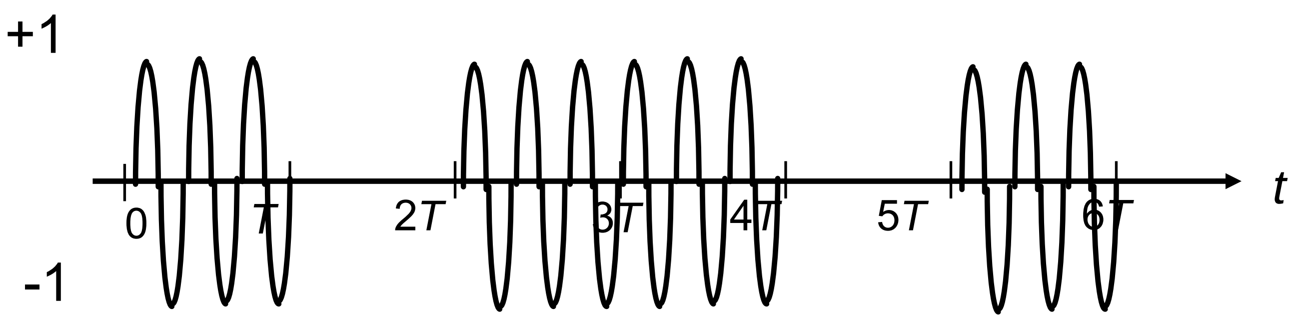


- Bandpass channels pass a range of frequencies around some center frequency f_c
  - Radio channels, telephone & DSL modems
- Digital modulators embed information into waveform with frequencies passed by bandpass channel
- Sinusoid of frequency f_c is centered in middle of bandpass channel
- Modulators embed information into a sinusoid

Amplitude Modulation and Frequency Modulation

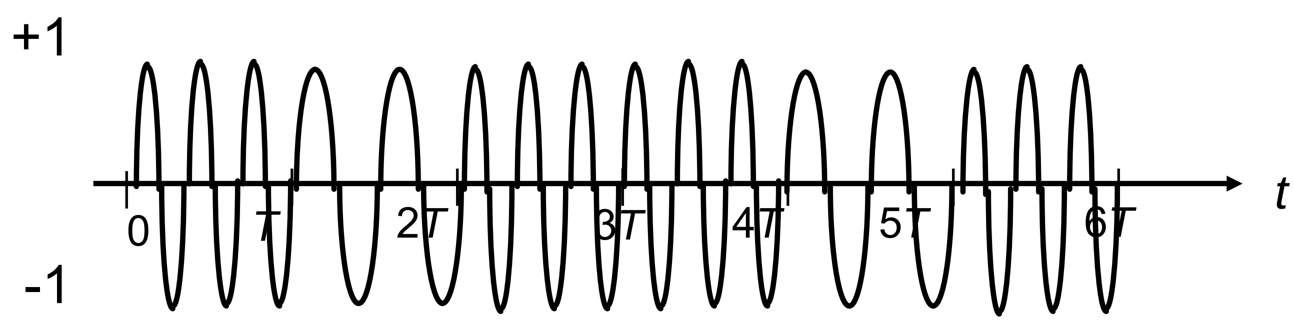
Information 1 0 1 1 0 1

Amplitude Shift Keying



Map bits into amplitude of sinusoid: “1” send sinusoid; “0” no sinusoid
Demodulator looks for signal vs. no signal

Frequency Shift Keying

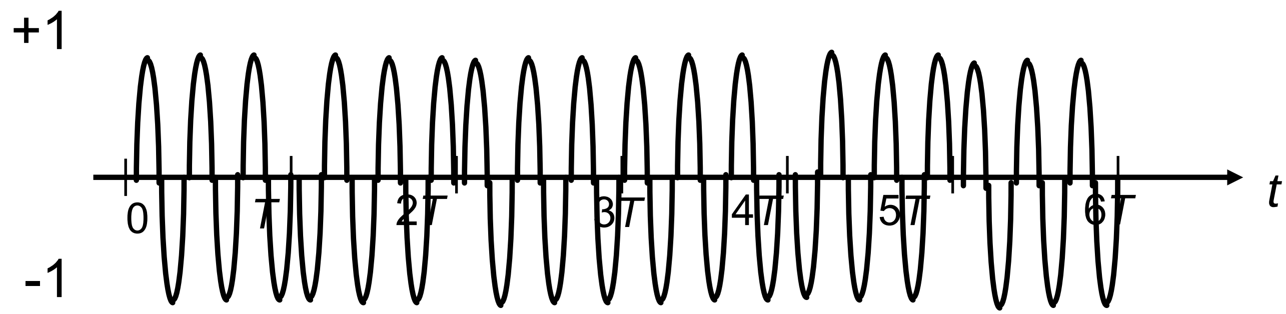


Map bits into frequency: “1” send frequency $f_c + \delta$; “0” send frequency $f_c - \delta$
Demodulator looks for power around $f_c + \delta$ or $f_c - \delta$

Phase Modulation

Information 1 0 1 1 0 1

Phase Shift Keying

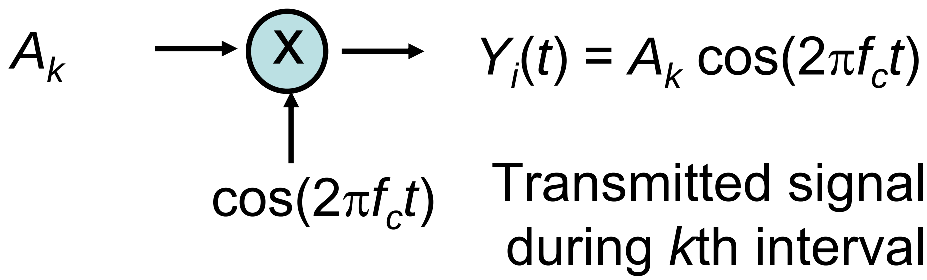


- Map bits into phase of sinusoid:
  - "1" send $A\cos(2\pi ft)$, i.e. phase is 0
  - "0" send $A\cos(2\pi ft+\pi)$, i.e. phase is π

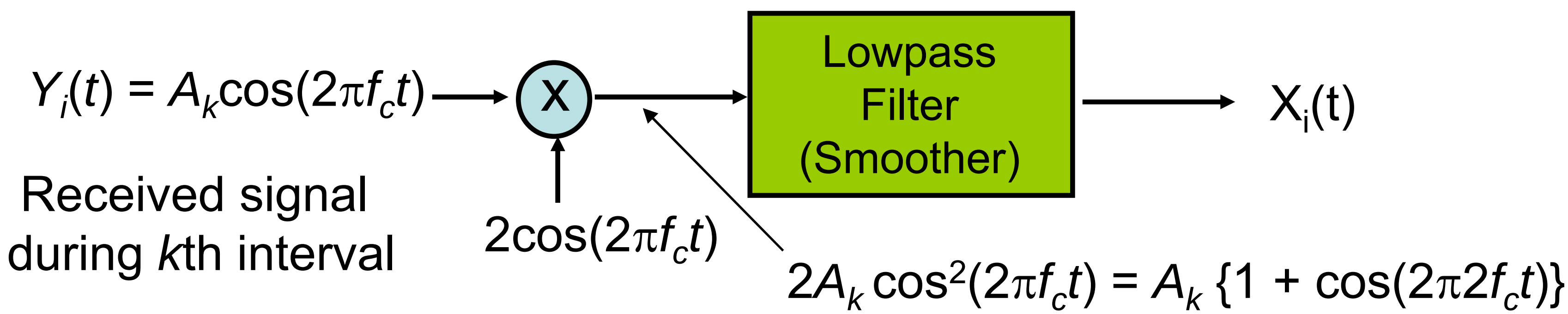
- Equivalent to multiplying $\cos(2\pi ft)$ by $+A$ or $-A$
  - "1" send $A\cos(2\pi ft)$, i.e. multiply by 1
  - "0" send $A\cos(2\pi ft+\pi) = -A\cos(2\pi ft)$, i.e. multiply by -1

Modulator & Demodulator

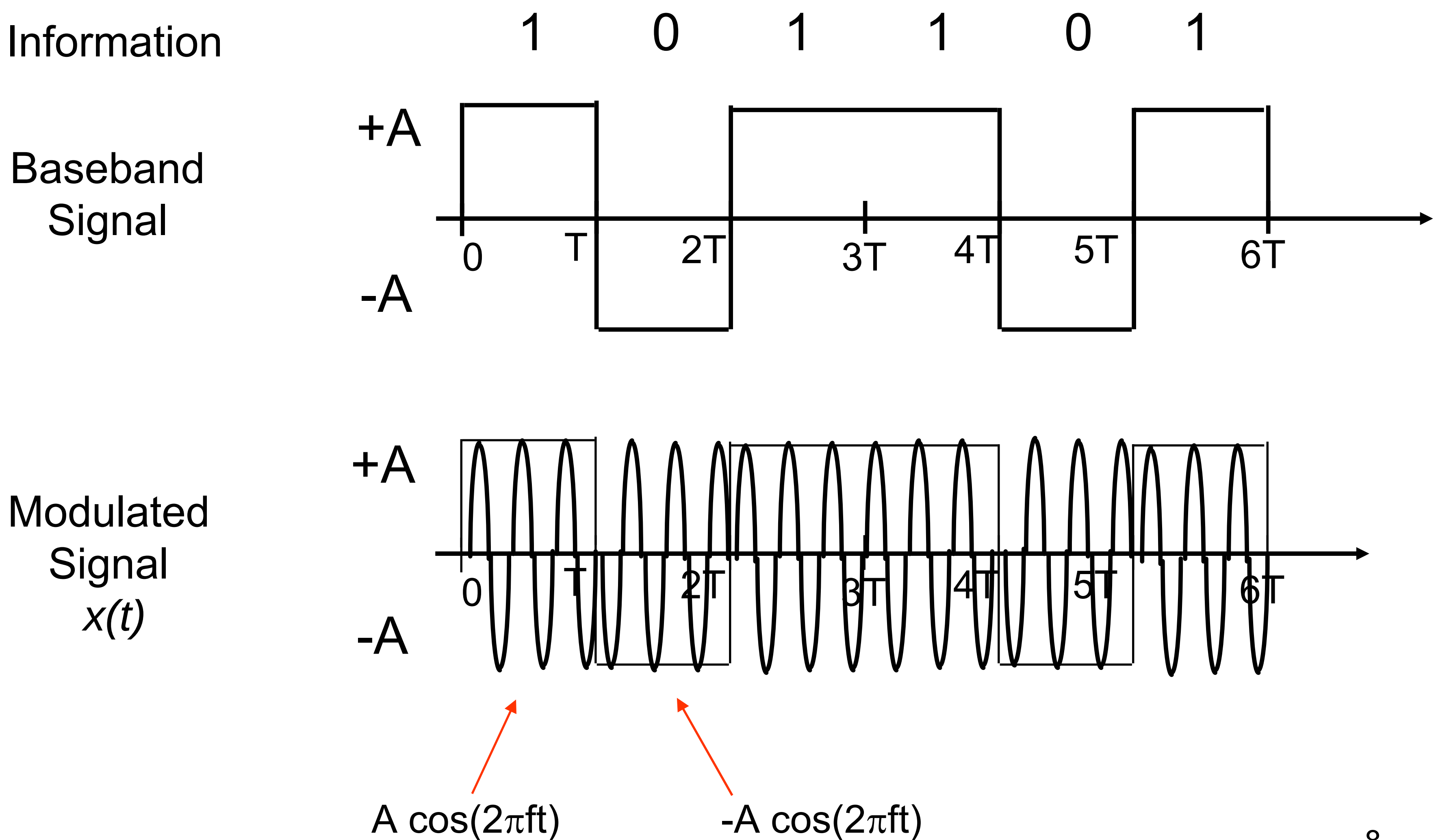
Modulate $\cos(2\pi f_c t)$ by multiplying by A_k for T seconds:



Demodulate (recover A_k) by multiplying by $2\cos(2\pi f_c t)$ for T seconds and lowpass filtering (smoothing):



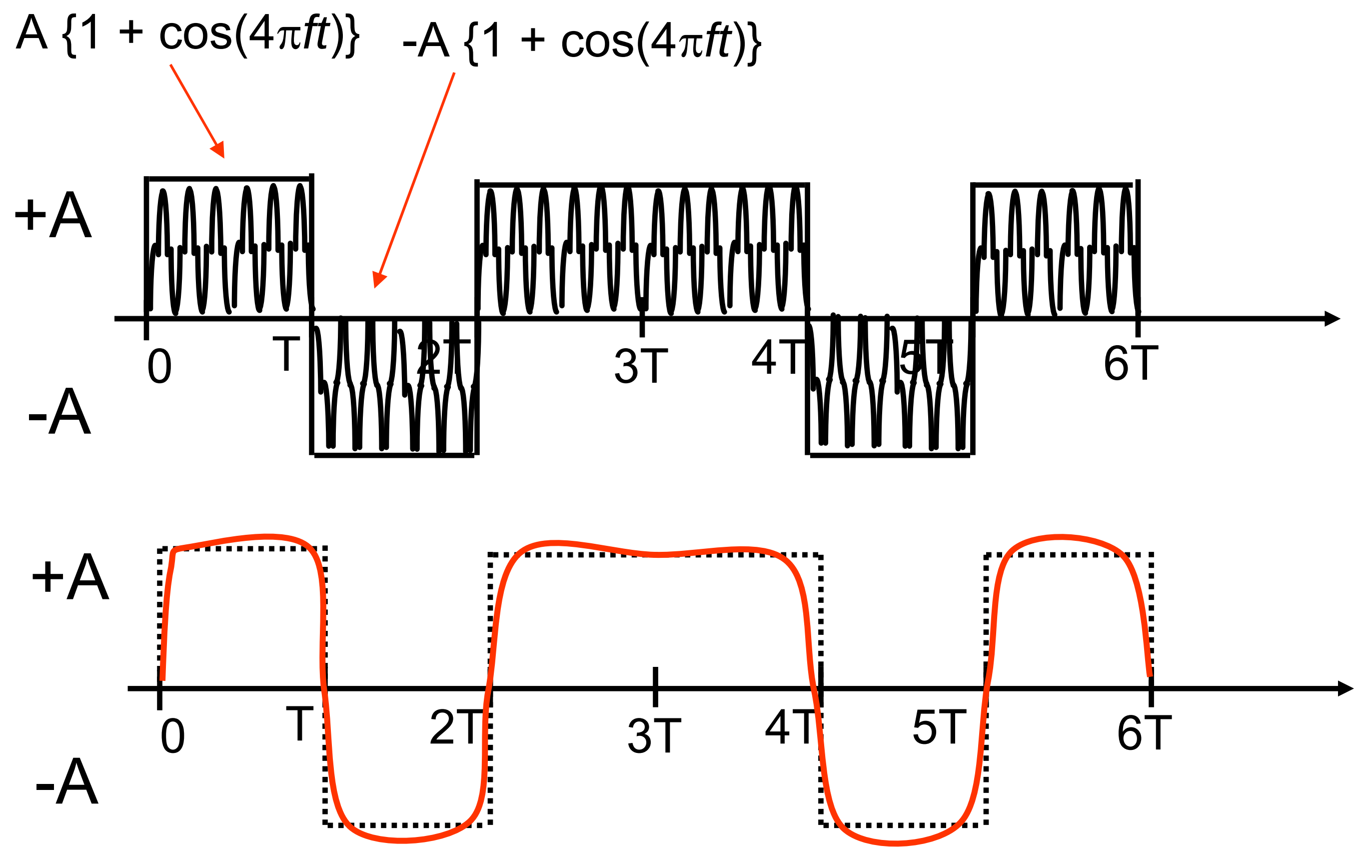
Example of Modulation



Example of Demodulation

After multiplication at receiver
$x(t) \cos(2\pi f_c t)$

Baseband signal discernable after smoothing



Recovered Information

1 0 1 1 0 1

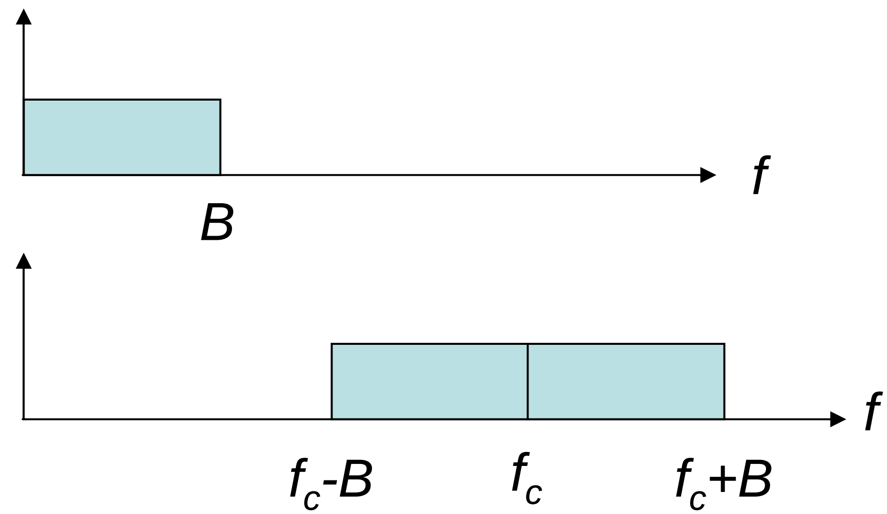
Fact from modulation theory

If

Baseband signal $x(t)$ with bandwidth B Hz

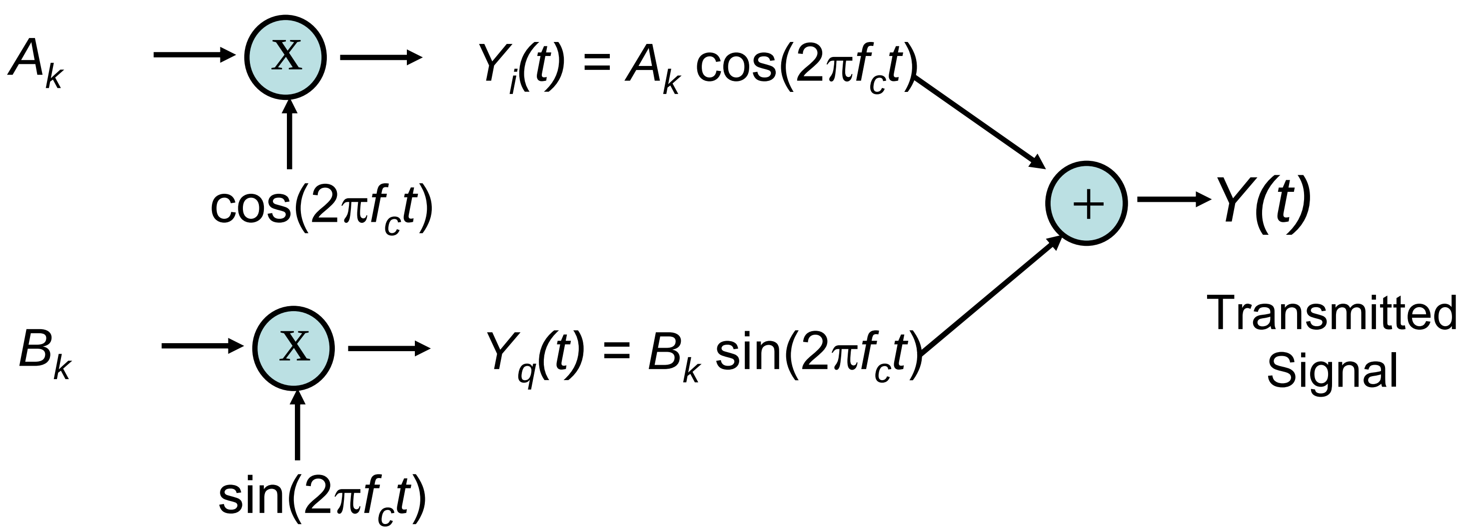
then

Modulated signal $x(t)\cos(2\pi f_c t)$ has bandwidth $2B$ Hz



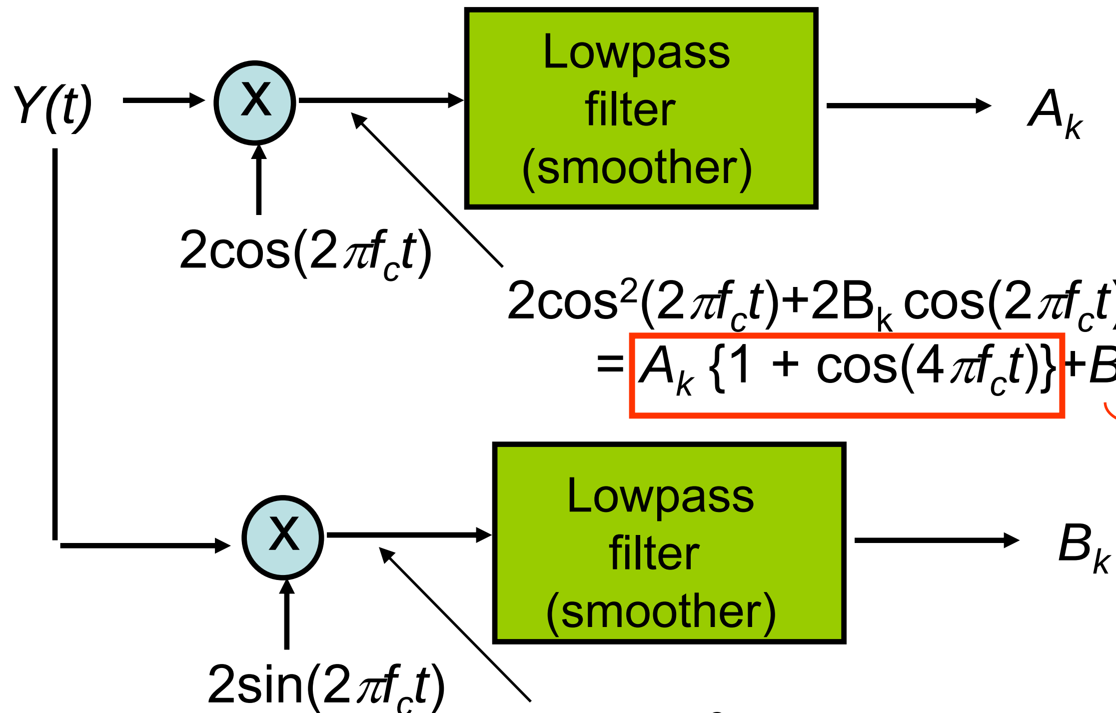
Quadrature Amplitude Modulation (QAM)

- QAM uses two-dimensional signaling
  - A_k modulates in-phase $\cos(2\pi f_c t)$
  - B_k modulates quadrature phase $\cos(2\pi f_c t + \pi/4) = \sin(2\pi f_c t)$
  - Transmit sum of inphase & quadrature phase components



- $Y_i(t)$ and $Y_q(t)$ both occupy the bandpass channel
- QAM sends 2 pulses/Hz

QAM Demodulation

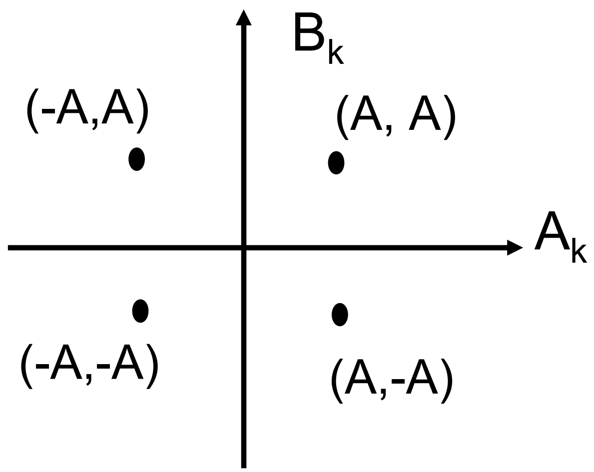


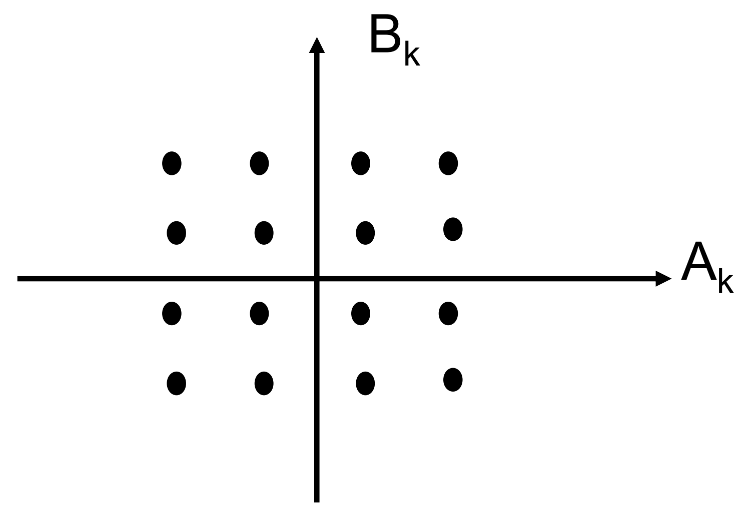
$$2\cos^2(2\pi f_c t) + 2B_k \cos(2\pi f_c t)\sin(2\pi f_c t)$$
$$= \boxed{A_k \{1 + \cos(4\pi f_c t)\}} + \underbrace{B_k \{0 + \sin(4\pi f_c t)\}}_{\text{smoothed to zero}}$$

$$2B_k \sin^2(2\pi f_c t) + 2A_k \cos(2\pi f_c t)\sin(2\pi f_c t)$$
$$= \boxed{B_k \{1 - \cos(4\pi f_c t)\}} + \underbrace{A_k \{0 + \sin(4\pi f_c t)\}}_{\text{smoothed to zero}}$$

Signal Constellations

- Each pair (A_k, B_k) defines a point in the plane
- *Signal constellation* set of signaling points





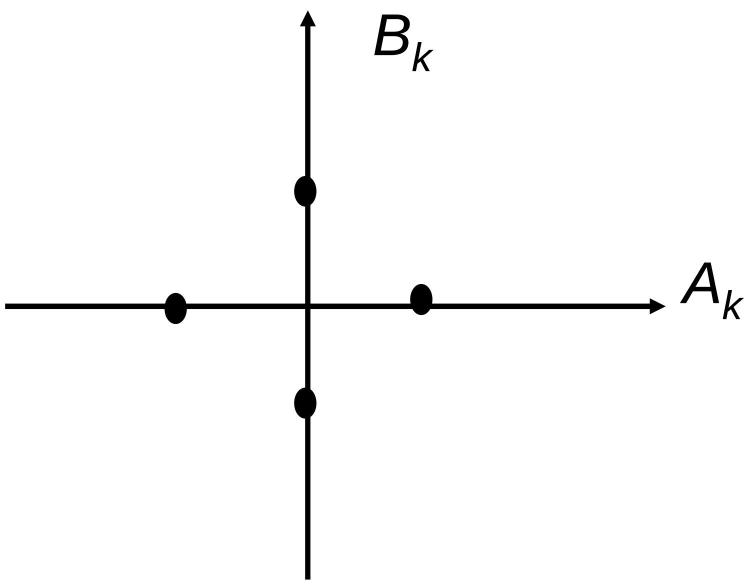
4 possible points per *T* sec.
2 bits / pulse

16 possible points per *T* sec.
4 bits / pulse

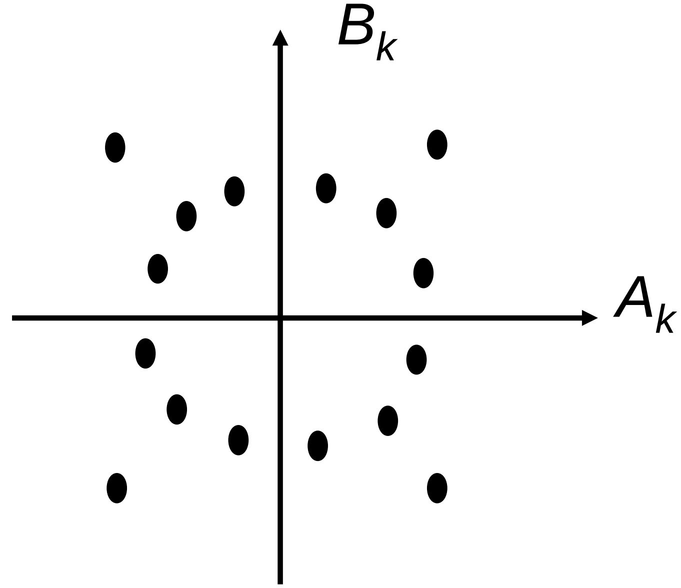
Other Signal Constellations

- Point selected by amplitude & phase

$$A_k \cos(2\pi f_c t) + B_k \sin(2\pi f_c t) = \sqrt{A_k^2 + B_k^2} \cos(2\pi f_c t + \tan^{-1}(B_k/A_k))$$



4 possible points per T sec.



16 possible points per T sec.

Telephone Modem Standards

Telephone Channel for modulation purposes has

W_c = 2400 Hz → 2400 pulses per second

Modem Standard V.32bis

- Trellis modulation maps m bits into one of 2^{m+1} constellation points

14,400 bps	Trellis 128	2400x6
9600 bps	Trellis 32	2400x4
4800 bps	QAM 4	2400x2

Modem Standard V.34 adjusts pulse rate to channel

- 2400-33600 bps Trellis 960 2400-3429 pulses/sec

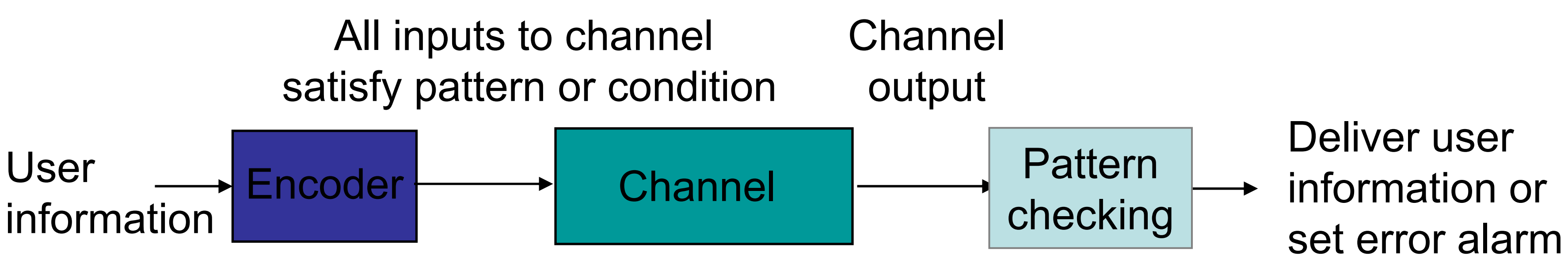
Error Detection

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)
  - 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: $(b_1, b_2, b_3, \ldots, b_{k,}, b_{k+!})$

- 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

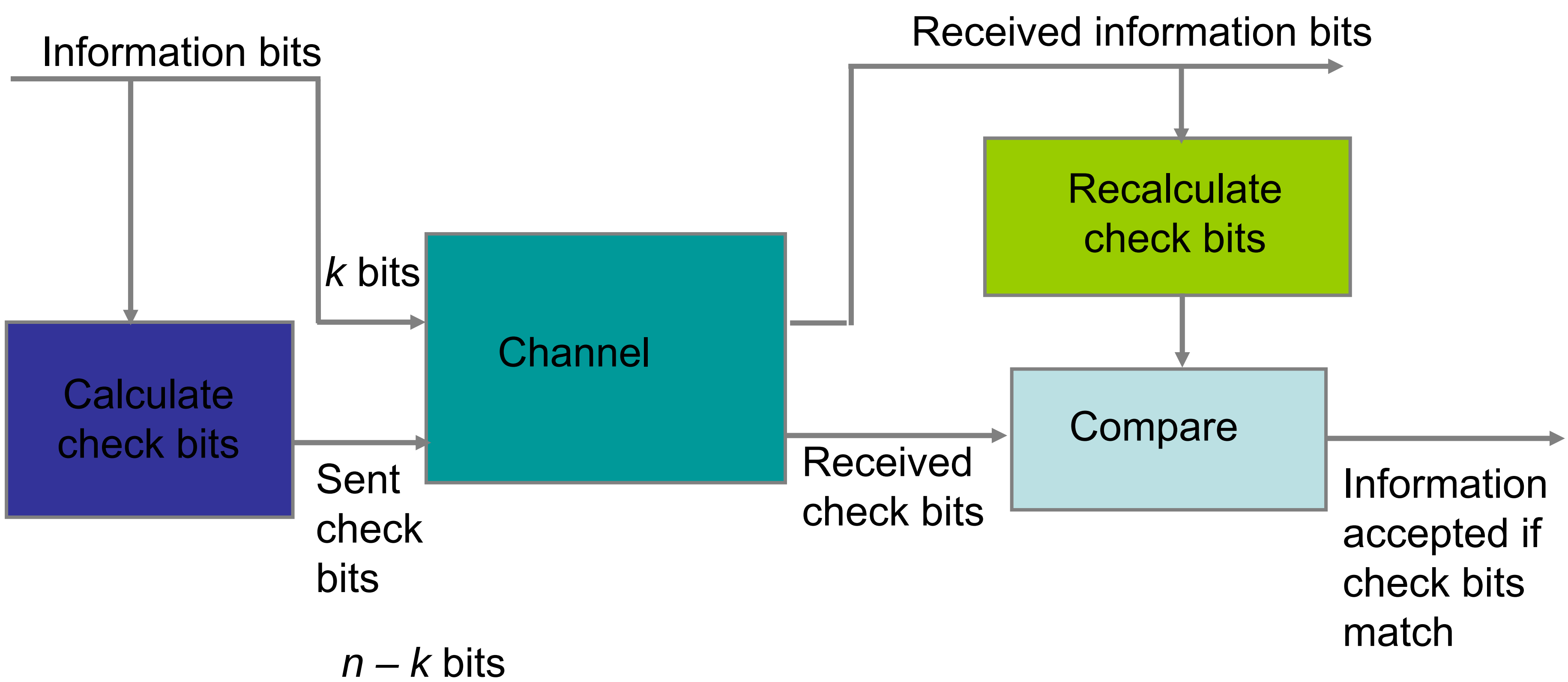
Example of Single Parity Code

- 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

Checkbits & Error Detection



How good is the single parity check code?

- *Redundancy*: Single parity check code adds 1 redundant bit per k information bits: overhead = $1/(k + 1)$
- *Coverage*: all error patterns with odd # of errors can be detected
  - An error pattern is a binary $(k + 1)$-tuple with 1s where errors occur and 0's elsewhere
  - Of 2^{k+1} binary $(k + 1)$-tuples, $\frac{1}{2}$ are odd, so 50% of error patterns can be detected
- Is it possible to detect more errors if we add more check bits?
- Yes, with the right codes

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

$$
\begin{array}{ccccc|c}
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1 & 1
\end{array}
$$

Last column consists of check bits for each row

Bottom row consists of check bit for each column

Error-detecting capability

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

One error

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

Two errors

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

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

Three errors

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

Four errors (undetectable)

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:
  - Internet Check Sums
  - CRC Polynomial Codes

Internet Checksum

- Several Internet protocols (e.g. IP, TCP, UDP) use check bits to detect errors in the *IP header* (or in the header and data for TCP/UDP)
- A checksum is calculated for header contents and included in a special field.
- Checksum recalculated at every router, so algorithm selected for ease of implementation in software
- Let header consist of L, 16-bit words,

  $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{L-1}$
- The algorithm appends a 16-bit checksum $\mathbf{b}_L$

Checksum Calculation

The checksum $\mathbf{b}_L$ is calculated as follows:

- Treating each 16-bit word as an integer, find

$$\mathbf{x} = \mathbf{b}_0 + \mathbf{b}_1 + \mathbf{b}_2 + \ldots + \mathbf{b}_{L-1} \text{ modulo } 2^{16}-1$$

- The checksum is then given by:

$$\mathbf{b}_L = -\mathbf{x} \quad \text{modulo } 2^{16}-1$$

Thus, the headers must satisfy the following ***pattern***:

$$\mathbf{0} = \mathbf{b}_0 + \mathbf{b}_1 + \mathbf{b}_2 + \ldots + \mathbf{b}_{L-1} + \mathbf{b}_L \text{ modulo } 2^{16}-1$$

- The checksum calculation is carried out in software using one's complement arithmetic

Internet Checksum Example

Use Modulo Arithmetic

- Assume 4-bit words
- Use mod 2^4-1 arithmetic
- $\underline{b}_0$=1100 = 12
- $\underline{b}_1$=1010 = 10
- $\underline{b}_0$+$\underline{b}_1$=12+10=7 mod15
- $\underline{b}_2$ = -7 = 8 mod15
- Therefore
- $\underline{b}_2$=1000

Use Binary Arithmetic

- Note 16 =1 mod15
- So: 10000 = 0001 mod15
- leading bit wraps around

$b_0 + b_1$ = 1100+1010
=10110
=10000+0110
=0001+0110
=0111
=7

Take 1s complement

b_2 = -0111 =1000

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 Arithmetic

- 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:

$$\begin{aligned}(x^7 + x^6 + 1) + (x^6 + x^5) &= x^7 + x^6 + x^6 + x^5 + 1 \\ &= x^7 + (1+1)x^6 + x^5 + 1 \\ &= x^7 + x^5 + 1 \quad \text{since } 1+1=0 \text{ mod2}\end{aligned}$$

Multiplication:

$$\begin{aligned}(x + 1)(x^2 + x + 1) &= x(x^2 + x + 1) + 1(x^2 + x + 1) \\ &= x^3 + x^2 + x) + (x^2 + x + 1) \\ &= x^3 + 1\end{aligned}$$

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

$$\begin{array}{rl}
 & x^3 + x^2 + x \qquad = q(x) \text{ quotient} \\
\cline{2-2}
x^3 + x + 1 \,) & x^6 + x^5 \\
 & x^6 + \qquad x^4 + x^3 \\
\cline{2-2}
 & \quad x^5 + x^4 + x^3 \\
 & \quad x^5 + \qquad x^3 + x^2 \\
\cline{2-2}
 & \qquad x^4 + \qquad x^2 \\
 & \qquad x^4 + \qquad x^2 + x \\
\cline{2-2}
 & \qquad\qquad\qquad x \quad = r(x) \text{ remainder}
\end{array}$$

divisor

dividend

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

Polynomial Coding

- Code has binary *generating 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 $n-1$

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

Polynomial Encoding: Steps

1. Multiply $i(x)$ by x^{n-k}
2. Divide $x^{n-k}i(x)$ by $g(x)$

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

3. Add remainder $r(x)$ to $x^{n-k}i(x)$

$$b(x) = x^{n-k}i(x) + r(x) \leftarrow \text{transmitted codeword}$$

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

$$
\begin{array}{rl}
 & x^3 + x^2 + x \\
\cline{2-2}
x^3 + x + 1 \,) & x^6 + x^5 \\
 & x^6 + \ x^4 + x^3 \\
\cline{2-2}
 & \ x^5 + x^4 + x^3 \\
 & \ x^5 + \ x^3 + x^2 \\
\cline{2-2}
 & \ x^4 + \ x^2 \\
 & \ x^4 + \ x^2 + x \\
\cline{2-2}
 & \ x
\end{array}
$$

$$
\begin{array}{rl}
 & 1110 \\
\cline{2-2}
1011 \,) & 1100000 \\
 & 1011 \\
\cline{2-2}
 & 1110 \\
 & 1011 \\
\cline{2-2}
 & 1010 \\
 & 1011 \\
\cline{2-2}
 & 010
\end{array}
$$

Transmitted codeword:

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

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

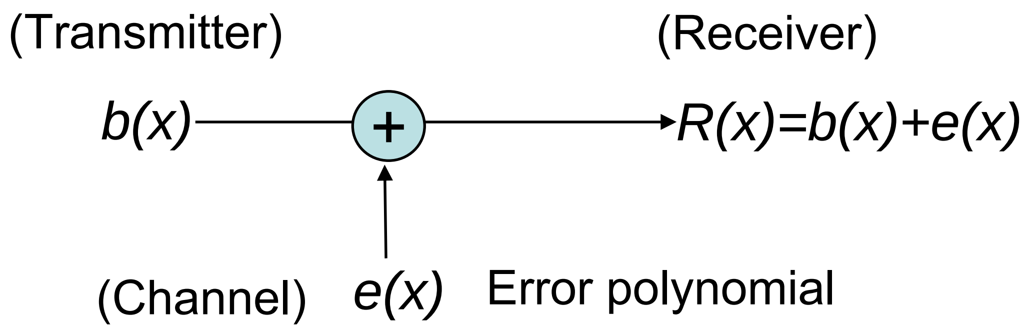
The *Pattern* in Polynomial Coding

- 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

Undetectable error patterns



- $e(x)$ has 1s in error locations & 0s elsewhere
- Receiver divides the received polynomial $R(x)$ by $g(x)$
- Blindspot: If $e(x)$ is a multiple of $g(x)$, that is, $e(x)$ is a nonzero codeword, then

$$R(x) = b(x) + e(x) = q(x)g(x) + q'(x)g(x)$$

- *The set of undetectable error polynomials is the set of nonzero code polynomials*
- *Choose the generator polynomial so that selected error patterns can be detected.*

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

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