Information Redundancy

- Code, codeword, binary code
- Error detection, error correction
- Hamming distance:
  - number of bits in which two words differ
- Odd/even parity
  - the total number of 1s is odd/even
- Basic parity approaches
  - bit-per-word
  - bit-per-byte
  - bit-per-chip
  - bit-per-multiple-chips
  - interlaced parity

Error Detection/Correction

- Let's look at an old principle to error correction
  - Hamming Code
  - any computer organization book will be a good reference
    - e.g. William Stallings' *Computer Organization and Architecture*
  - rely on check bits to identify whether bit has been changed
  - identification of changed bit allows for correction

Overlapped Parity

12	11	10	9	8	7	6	5	4	3	2	1	Bit Position
0	1	0	1	0	1	0	1	0	1	0	1	
0	1	1	0	0	1	1	0	0	1	1	0	
1	0	0	0	0	1	1	1	1	0	0	0	
1	1	1	1	1	0	0	0	0	0	0	0	
				C4				C3		C2	C1	Check Bit
D8	D7	D6	D5		D4	D3	D2		D1			Data Bit

$$2^k - 1 \geq m + k$$

m = data bits
k = parity bits

Overlapped Parity

- Syndrome is derived from comparing, i.e. XOR, transmitted and received/recomputed check bits.
- Syndrome has following characteristics (previous example)
  - if syndrome contains all 0's
    - » no error has been detected
  - if syndrome contains one and only one bit set to 1
    - » error has occurred in one of the 4 check bits
  - if syndrome contains more than one bit set to 1
    - » numerical value of the syndrome indicates the position of the data-bit error
    - » this bit is then inverted for correction

Compute Check

0	1	0	1	0	1	0	1	0	1	0	1	
0	1	1	0	0	1	1	0	0	1	1	0	
1	0	0	0	0	1	1	1	1	0	0	0	
1	1	1	1	1	0	0	0	0	0	0	0	
				C4				C3		C2	C1	Check Bit
D8	D7	D6	D5		D4	D3	D2		D1			Data Bit

$$C1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7$$

$$C2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$$

$$C3 = D2 \oplus D3 \oplus D4 \oplus D8$$

$$C4 = D5 \oplus D6 \oplus D7 \oplus D8$$

Overlapped Parity

- Example
  - data = 1110 0001
  - compute check bits:

$$C1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7$$
$$C2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$$
$$C3 = D2 \oplus D3 \oplus D4 \oplus D8$$
$$C4 = D5 \oplus D6 \oplus D7 \oplus D8$$

$$C1 = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \quad = 0 \longleftarrow \text{least significant bit}$$
$$C2 = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \quad = 1$$
$$C3 = 0 \oplus 0 \oplus 0 \oplus 1 \quad = 1$$
$$C4 = 0 \oplus 1 \oplus 1 \oplus 1 \quad = 1 \longleftarrow \text{most significant bit}$$

Overlapped Parity

- Example
  - data sent is 1110 0001 and transmitted check bits are 1110
  - assume received data is: 01100001
    - » note that most sig. bit has been corrupted/flipped
  - received check bits are: 1110
  - recomputed check bits:

$$C1 = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \quad = 0$$

$$C2 = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \quad = 1$$

$$C3 = 0 \oplus 0 \oplus 0 \oplus 0 \qquad = 0$$

$$C4 = 0 \oplus 1 \oplus 1 \oplus 0 \qquad = 0$$

  - Syndrome: 1110 XOR 0010 = 1100

Applying Syndrome

12	11	10	9	8	7	6	5	4	3	2	1	Bit Position
0	1	0	1	0	1	0	1	0	1	0	1	
0	1	1	0	0	1	1	0	0	1	1	0	
1	0	0	0	0	1	1	1	1	0	0	0	
1	1	1	1	1	0	0	0	0	0	0	0	
				C4				C3		C2	C1	Check Bit
D8	D7	D6	D5		D4	D3	D2		D1			Data Bit

Syndrome 1100 detects D8 as faulty

m-of-n codes

- All code words are *n* bits in length and contain exactly *m* 1s
- Simple implementation:
  - add/append second data word
  - select word such that code word contains *m* 1s
  - code is separable
  - 100% overhead
- Hamming distance is 2
  - e.g. 1st error sets bit, 2nd error resets other bit

Checksum

- Separable code to achieve error detection capability
- Checksum is the sum of the original data
- Single-precision checksum
  - overflow problem, i.e. adding n bits modulo 2^n
- Double-precision checksum
  - uses double precision, i.e. compute 2n-bit checksum from n-bit words using modulo-2^{2n} arithmetic.
- Honeywell checksum
  - compose word of double length by concatenating 2 consecutive words
  - compute checksum on these double words
- Residue checksum
  - like single-precision checksum, but overflow is now fed back as carry

Cyclic codes

- Cyclic Redundancy Checks (CRC)
  - Parity bits still subject to burst noise, uses large overhead (potentially) for improvement of 2-4 orders of magnitude in probability of detection.
  - CRC is based on a mathematical calculation performed on message. We will use the following terms:
    - » M - message to be sent (k bits)
    - » F - Frame check sequence (FCS) to be appended to message (n bits)
    - » T - Transmitted message includes both M and F $\Rightarrow$ ($k+n$ bits)
    - » G - a $n+1$ bit pattern (called generator) used to calculate F and check T

Cyclic codes

- Idea behind CRC
  - given a k-bit frame (message)
  - transmitter generates a n-bit sequence called frame check sequence (FCS)
  - so that resulting frame of size $k+n$ is exactly divisible by some predetermined number
- Multiply M by 2^n to shift and add F to padded 0s

$$T = 2^n M + F$$

Cyclic codes

- Dividing $2^n M$ by G gives quotient and remainder

$$\frac{2^n M}{G} = Q + \frac{R}{G}$$

remainder
is 1 bit less
than divisor

then using R as our FCS we get

$$T = 2^n M + R$$

on the receiving end, division by G leads to

$$\frac{T}{G} = \frac{2^n M + R}{G} = Q + \frac{R}{G} + \frac{R}{G} = Q$$

Note:
mod 2 addition,
no remainder

Cyclic codes

- Therefore, if the remainder of dividing the incoming signal by the generator G is zero, no transmission error occurred.
- Assume T + E was received (Note: E is the error)

$$\frac{T+E}{G} = \frac{T}{G} + \frac{E}{G}$$

since T/G does not produce a remainder, an error is detected only if E/G produces a non-zero value

Cyclic codes

- example, assume generator G(X) has at least 3 terms
  - G(x) has three 1-bits
    - » detects all single bit errors
    - » detects all double bit errors
    - » detects odd #'s of errors if $G(X)$ contains the factor $(X + 1)$
    - » any burst errors < length of FCS
    - » most larger burst errors
    - » it has been shown that if all error patterns likely, then the likelihood of a long burst not being detected is $1/2^n$

Cyclic codes

- What does all of this mean?
  - A polynomial view:
    - View CRC process with all values expressed as polynomials in a dummy variable X with binary coefficients, where the coefficients correspond to the bits in the number.
      - for M = 110011 we get $M(X) = X^5 + X^4 + X + 1$
      - for G = 11001 we get $G(X) = X^4 + X^3 + 1$
      - Math is still mod 2
    - An error E(X) is received and **undetected** iff it is divisible by $G(X)$

Cyclic codes

- Common CRCs
  - CRC-12 = $X^{12} + X^{11} + X^3 + X^2 + X + 1$
  - CRC-16 = $X^{16} + X^{15} + X^2 + 1$
  - CRC-CCITT = $X^{16} + X^{12} + X^5 + 1$
  - CRC-32 = $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Hardware Implementation:

$$G(X) = 1 + a_1 X + a_2 X^2 + \cdots + a_{n-1} X^{n-1} + a_n X^n$$