Error checking

• Simple system - Parity
Extra bit added on a per character
basis so that the total number of '1's
transmitted is even (or odd).

eg. ASCII code (even parity)

'A' = 1000001 parity bit =0

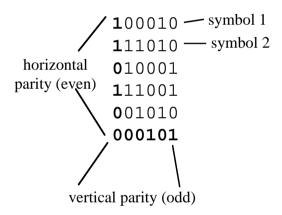
'B' = 1000010 parity bit = 0

C' = 1000011 parity bit = 1

if 1 bit is corrupt error is spotted if 2 bits are corrupt error is missed if 3 bits are corrupt error is spotted

Parity (cont)

• To spot more errors might use 'vertical parity' - added as an extra character



Not good on noisy lines

parity (cont)

 Horizontal and vertical (row and column) parity spots more errors some may still be missed.

> 00101 11110 01001 01100 10001—odd parity

- if all bits corrupted then errors are missed.
- need a better system...

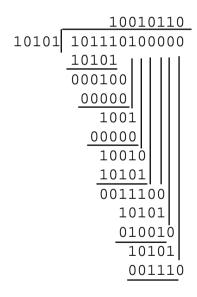
Polynomial Codes - CRC's

- Need systems good at spotting error bursts -Cyclic Redundancy Check
- CRC consists of a frame of data plus N extra bits called the Frame Check Sequence (FCS)
- 1. Take frame and add N zeros
- 2. Divide by polynomial *G* using modulo 2 binary arithmetic (*G* has *N*+1 digits)
- 3. Use remainder as CRC FCS.
- 4. Transmit frame and CRC.
- 5. Receiver checks CRC.

[Note: division - binary modulo 2 is same as XOR at each stage of division]

CRC example

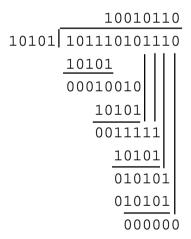
• 8 bit frames, 4 bit CRC, G = 10101



• CRC = 1110 so 101110101110 transmitted

CRC example (cont.)

• Receiver repeats division of all received bits - 0 remainder implies no errors



No remainder implies no errors

CRC example 2

- 4 bit frame, 2 bit FCS, G=101
- data = 1100

CS25110 - Communications and Telematics Information theory and coding

Error correcting codes **Hamming code**

- To identify all single bit errors (and locate) in 4 bit data
- check bits occupy all bit positions which are a power of two

7 6 5 4 3 2 1
$$D_4$$
 D_3 D_2 C_3 D_1 C_2 C_1

• check bits are the sum (modulo 2 of) the binary representation of data positions containing a '1'.

ie. for data =
$$1100$$

7 6 5 4 3 2 1
1 1 0
$$C_3$$
 0 C_2 C_1
7 = 111
6 = 110
 $C = 001$

• transmit 1100001

Hamming code (cont.)

• On receipt add binary values for all bits with '1's present:

```
76...1
1100001 => 7=111
              6=110
              1=001
                000
                      <= OK
say:
1101001 => 7=111
              6=110
              4=100
              1=001
                100
                      <= bit 4 in error
0101001 =>
              6=110
              4=100
              1=001
                011<=wrong(2bit error)
1000001 =>
             7=111
              1=001
                      <=bit 6 in error
```

Observations

- FEC needs lots of bits (expensive)
- normally simply detect error and use a system of requesting re transmission
- Typically CRCs detect errors or corruption.