New data

111001110 → 1 Block

110111010 — 2 Block

001110001 1 — 3 block

10101001... — 4 block

101000101 — 5 block.

for transmission

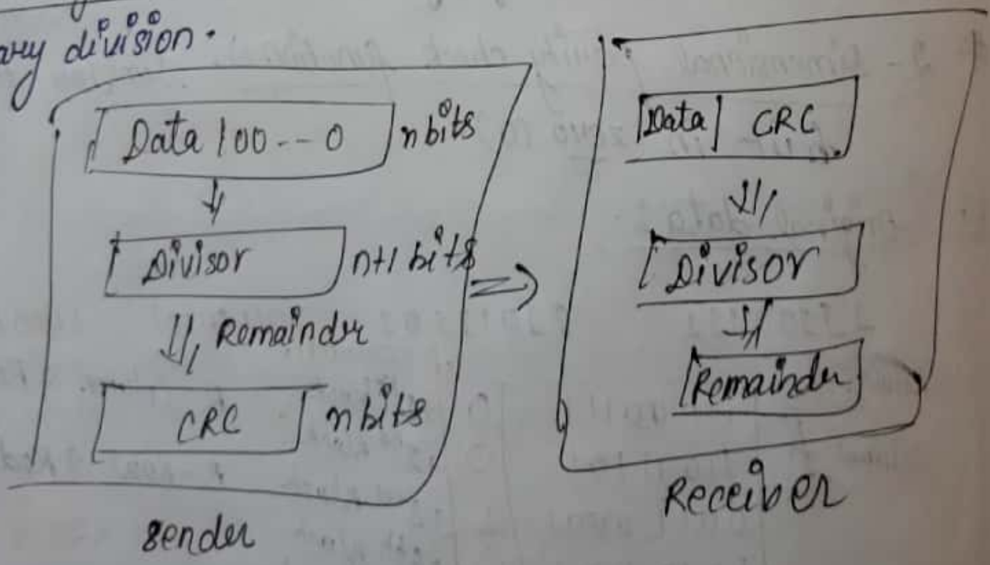
For Error checking:-

111 00 111	0
110 11 101	0
00 1 100 01	1
101 0 1 001	0
1010 0010	1
00 000000	0

✓ check bits of 2D parity function.

→ In 2D parity check function, a block of bits is divided in rows and the redundant row 4 column of bits are added to the whole block.

CRC (Cyclic Redundancy check) :- CRC is based on binary division.



Scanned with CamScanner

Ex 100100 ← added

G(x) = 1101 → 4 bits = n

n-1 zeroes

```
1101 ) 100100000 ( 111101
       1101
       ----
       X1000
        1101
        ----
        X1010
         1101
         ----
         X1110
          1101
          ----
          X0110
           0000
           ----
           X1100
            1101
            ----
            X001 → CRC
```

same = 0
Diff = 1

New data :- 100100 + CRC

100100001

$\underbrace{001}_{\text{CRC}}$

At receiver End :-

Data is 100100 $\underline{001}$ → CRC

```
1101 ) 1001 00 001 ( 111101
        1101
       x1000
        1101
       x1010
        1101
       x1110
        1101
       x0110
        0000
       x1101
        1101
       x000  — CRC (so) No error
                         is there.
                  else error is there
```

⇒ Polynomial form :-

Data ⇒ $x^7+x^6+x^4+x^3+x+1$

Divisor = x^2+x.

Convert data into binary = $x^7+x^6+x^4+x^3+x^{1}+1$

Polynomial function :- $x^7+x^6+x^5+x^4+x^3+x^2+x^1+x^0$.

Binary data = 11011011 → 8 bits

New polynomial function :- $1\cdot x^7+1\cdot x^6+0\cdot x^5+1\cdot x^4+1\cdot x^3+0\cdot x^2+$
$1\cdot x^1+1\cdot x^0$.

Data – 11011011 → 8 bits

Divisor = $x^2+x^1+x^0$.

= 110

```
          10010010
110 ) 1101101100
      110
      x001
       000
       x011
        000
        x110
         110
         x001
          000
          x011
           000
           x110
            110
             000
             000
             x00  — CRC
```

checksum:- The error detection method used by the higher layer protocols is called checksum.

like VRC, LRC, CRC, checksum is based on the concept at redundancy.

Sender [Data + checksum]

section 1 → n bits
section 2 → n bits
|
|
checksum → All 0s
|
section k → n bits

sum → n bits

complement [n bits] → checksum

n-bits

Receiver

section 1 → n bits
section 2 → n bits
checksum → n bits
—
—
section k → n bits

sum → n bits

complement [n bits] → Result

if result is zero then accept, otherwise discard.

Ex Data = AEFG, checksum = ?

soln

A → 65
E → 69
F → 70
G → 71

sum → 275

+ 2

77

n bits = 2

complement FF

− 77

88 → checksum.

Now new data is [AEFG88]

For receiver Data = AEFG88

Scanned with CamScanner

A → 65
E → 69
F → 70
G → 71
checksum → 88

sum → 363
↳+3
66

FF
−66
99

AEFG88

A → 65
E → 69
F → 70
G → 71
275
↳+2
77

FF
−77
88

↳ Matched
80 ~~no error~~
80 no error.

Let at receiver Data = ABCG88

A → 65
B → 66
C → 67
G → 71
269
↳+2
71

FF
−71
8E → New checksum

Not matched
Yes, error

Ex Suppose the following block of 16 bits is to be send using a checksum of 8 bits.

← 10101001 00111001 (16 bits).

Soln

Section 1 → 10101001 8 bits
Section 2 → 00111001 8 bits.
Checksum → 00000000 (optional)

sum → 11100010 (8 bit)
↳ Complement
00011101 → checksum

Data + checksum

(Data for transmission is) 10101001 00111001 00011101 ←

For receiver end :-

section 1 → 10101001
section 2 → 00111001
checksum → 00011101

sum → 11111111

complement → $\boxed{00000000}$ → No error

Assume

section 1 → 10101101
section 2 → 00101001
checksum → 00011101

10010011

01101100 → Non-zero
Error is there.

22 bit
↓
8
8
6

EX Now suppose there is a burst error of length five that effects four bits.

modified modified
10101111 11111001 00011101

section 1 → 10101111
section 2 → 11111001
checksum → 00011101

sum → 111000101
+1
11000110 → complement → $\boxed{00111001}$

Not all zero so error and discarded.

Scanned with CamScanner

Error Correction :– Adding of redundancy bits mechanism

→ Redundancy bits :– r – redundancy bits
m → Data bits.

Total No. of bits → m + r

$$\boxed{2^r \geq m+r+1}$$

↓
To identify How many redundancy bits can be added to data?.

(m) No. of data bits	(r) No. of Redundancy	Total bits (m+r)
1	2	3
2	3	5
3	3	6
4	3	7
5	4	9
6	4	10
7	4	11

calcu

$2^r \geq 1+r+1$
$2^r \geq 2+r$.
$r=1$
$2 \geq 2+1$
$r=2$
$4 \geq 2+2$.

$2^r \geq 2+r+1$
$2^r \geq 3+r$
$r=3$
$8 \geq 6$

Hamming Code :–

$r_1, r_2, r_4, r_8, r_{16}$ ....

$m=7, r=4$

11	10	9	8	7	6	5	4	3	2	1
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

* All the redundancy bits are responsible for some position of error correction.
Like.

For 1 bit = $d r_2 r_1$
For 2 bit = $d r_4 d r_2 r_1$

Positions :–

1st r_1 → 1, 3, 5, 7, 9, 11, 13, ...
2nd r_2 → 2, 3, 6, 7, 10, 11, ...
3rd r_4 → 4, 5, 6, 7, ...
4th r_8 → 8, 9, 10, 11, ...

criteria like
r_1 → All position which has LSB 1 is responsible.
↓
r_1

For Ex $r_1 \rightarrow$ 1, 3, 5, 7, 9.

0001, 0011, 0101, 0111, 1001

r_1 - is responsible. because 1 is at I^{st} position.

similarly,

$r_2 \rightarrow$ 2, 3, 6, 7, 10.

0010 0011 0110 0111 1010

r_2 is resposible. because 1 is at 2nd position.

Now

$r_4 \rightarrow$ 4, 5, 6, 7.

0100 0101 0110 0111

r_4 is responsible because 1 is at 3rd position.

$r_8 \rightarrow$ 8, 9, 10, 11 ....

1000 1001 1010 1011

r_8 is responsible because 1 is at 4th position.

Ex Data ⇒ 1001101

↳ 7 bits in the data

↳ so no. of redundant bits are r_1, r_2, r_4, r_8.

so total data :- 7 + 4 = 11

11	10	9	8	7	6	5	4	3	2	1
1	0	0	r_8 | 0	1	1	0	r_4 | 0	1	r_2 | 0	r_1 | 1

for counting

Now calculate the value of redundant bits r_1, r_2, r_4, r_8?

r_1 responsible for 1, 3, 5, 7, 9, 11.

No. of one's at these position = 3 – odd

so for even parity

$$\boxed{r_1 = 1}$$

Similarly for r_2.

$r_2 \rightarrow$ 2, 3, 6, 7, 10, 11.

No. of one's at these position = 4 – even

so

$$\boxed{r_2 = 0}$$

$r_4 \rightarrow$ 4, 5, 6, 7.

No. of 1's at these position = 2 – even.

so $\boxed{r_4 = 0}$

$r_8 \rightarrow$ 8, 9, 10, 11

No. of 1's at these position = 1 – odd.

$$\boxed{r_8 = 1}$$

Now fix these redundancy bits at the respective places:-

11	10	9	8	7	6	5	4	3	2	1
1	0	0	1	1	1	0	0	1	0	1
			r_8				r_4		r_2	r_1

New data for the transmission.

or $\rightarrow$ 11 bit data

$$\boxed{10011100101}$$

Suppose some bits are modified at receiver's end:-

1 0 0 1 1/0 1 0 0 1 0 1

$\uparrow$ Error at receiving End

$$\boxed{10010100101}$$

Scanned with CamScanner

~~Error Correction~~

Error Detection

$r_8 \; r_4 \; r_2 \; r_1$

$r_1 \rightarrow 1, 3, 5, 7, 9, 11 \rightarrow 1$
$r_2 \rightarrow 2, 3, 6, 7, 10, 11 \rightarrow 1$
$r_4 \rightarrow 4, 5, 6, 7 \rightarrow 1$
$r_8 \rightarrow 8, 9, 10, 11 \rightarrow 0$

New redundant bits at receiving end.

r_8	r_4	r_2	r_1
0	1	1	1

decimal value = 7.

Take complement of bit present at 7th position

Data = 10011100101

check '0' → '1'

If No error

$r_1 \rightarrow 0$
$r_2 \rightarrow 0$
$r_3 \rightarrow 0$
$r_4 \rightarrow 0$

No error

Hamming code will identify the burst error but can do single bit error correction. as it provide only single position where error is there.

Note: Multi-bit error detection & single bit error correction.