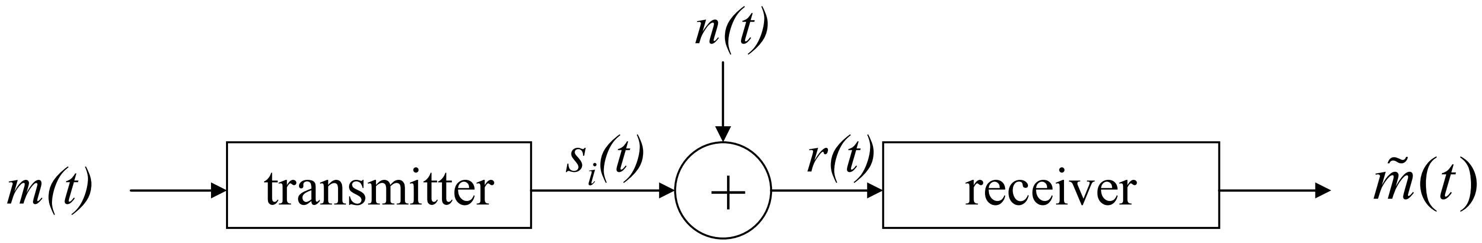
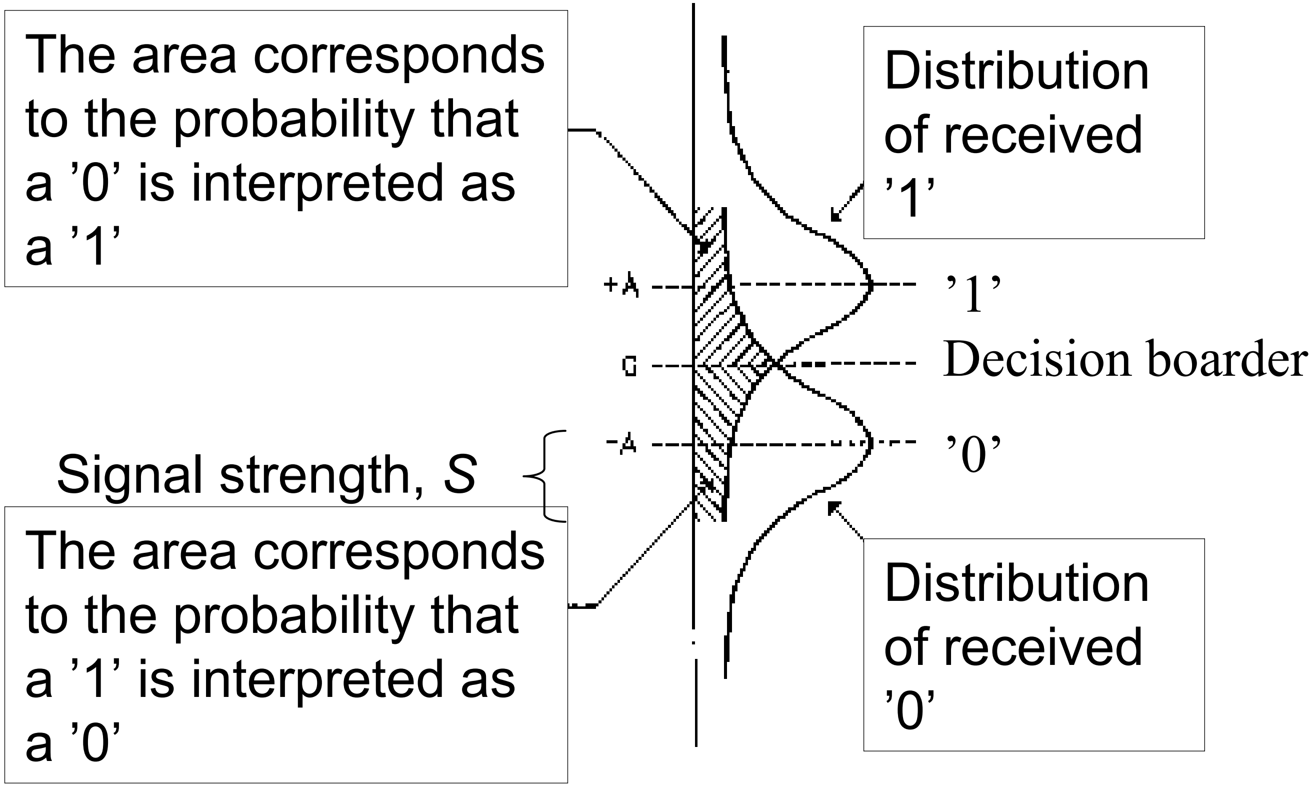
Channel coding

AWGN



$$\frac{E_b}{N_0} = \frac{B}{R}\frac{S}{N}$$


The area corresponds to the probability that a '0' is interpreted as a '1'

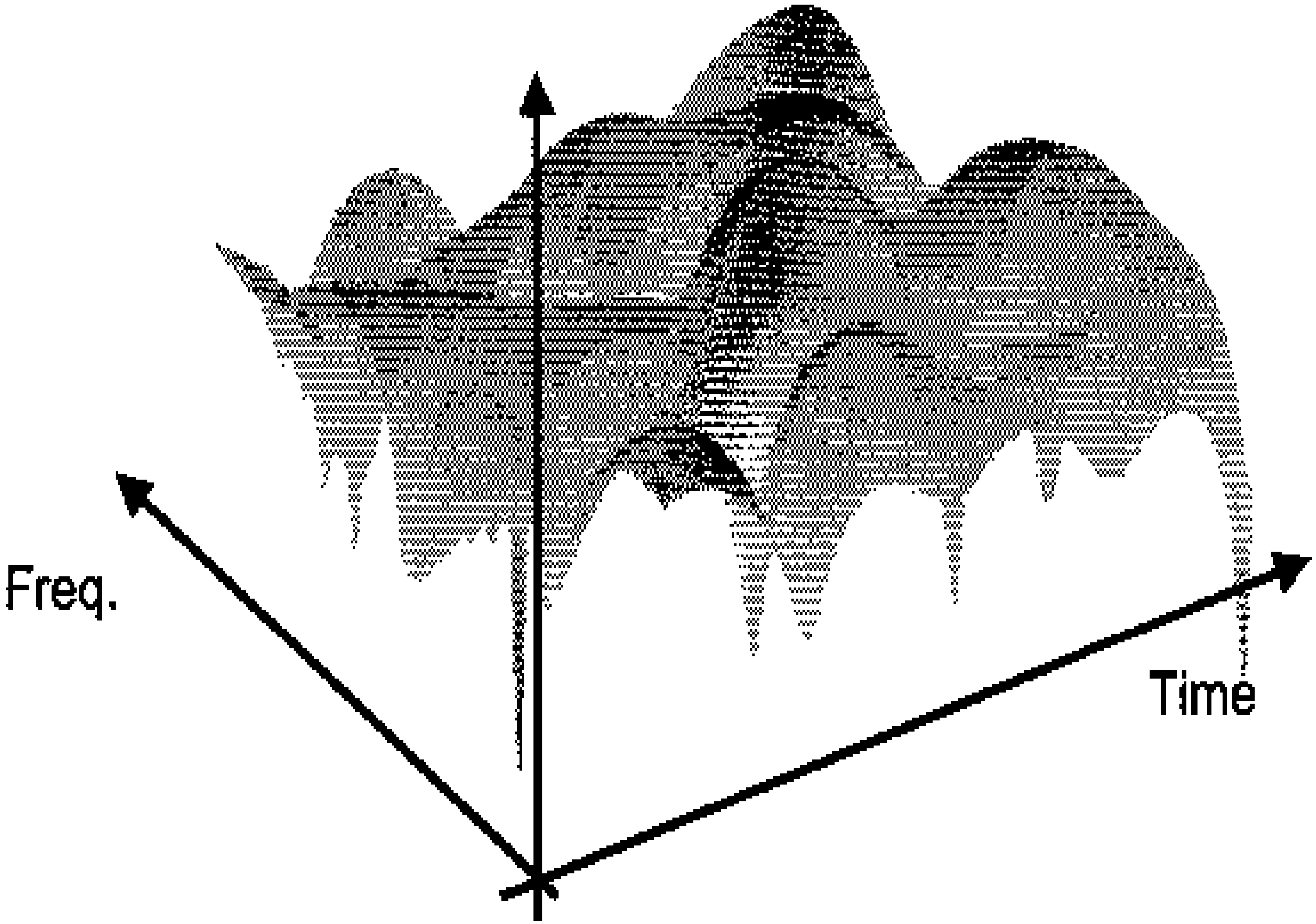
Distribution of received '1'

Signal strength, *S*

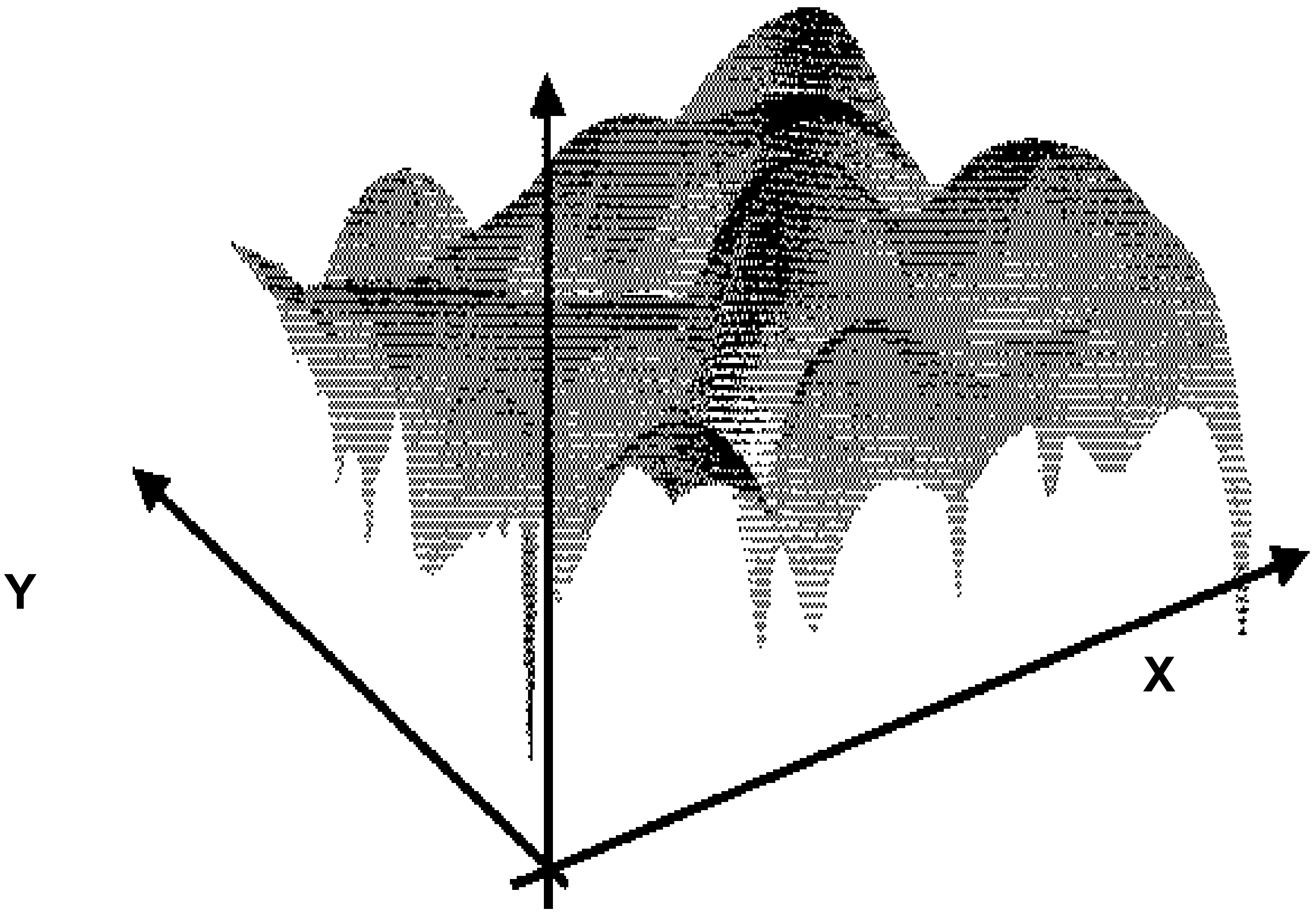
The area corresponds to the probability that a '1' is interpreted as a '0'

Distribution of received '0'

Fading



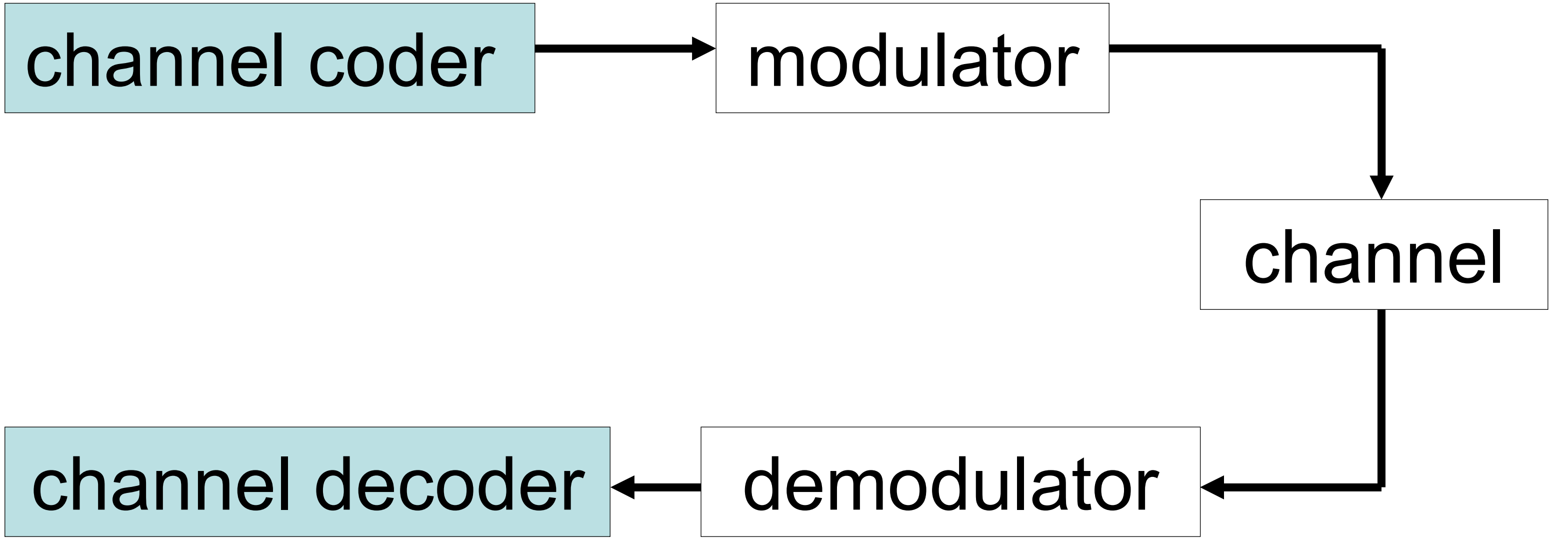
Fading



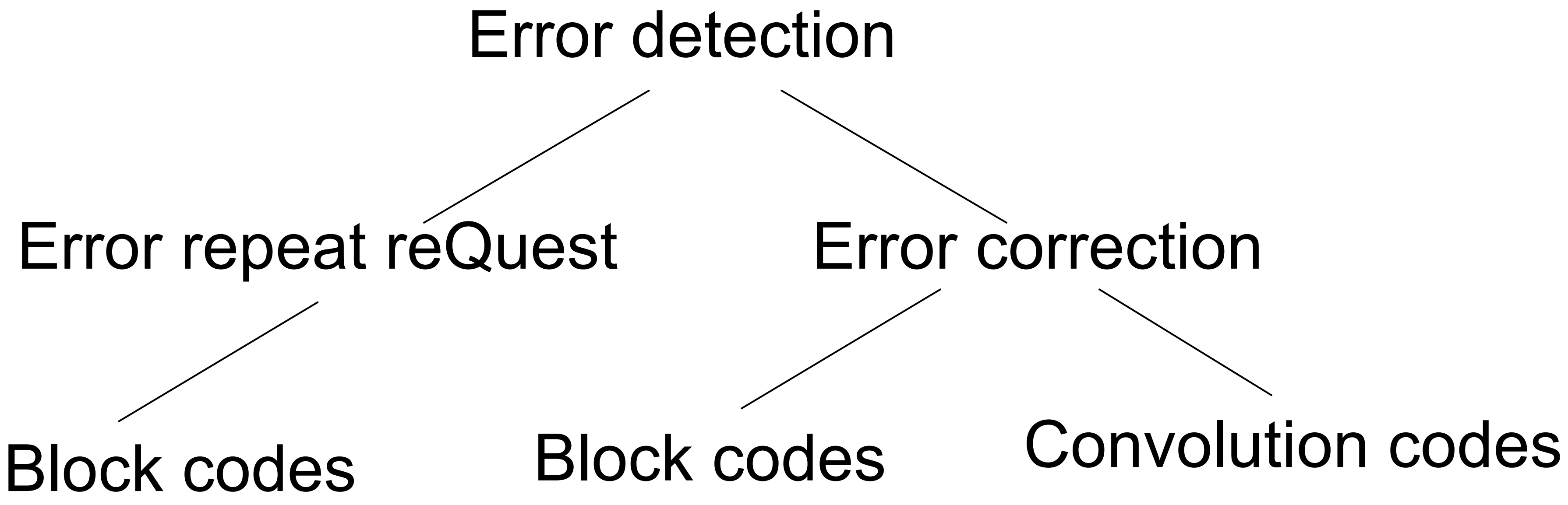
Interference

- Co-channel interference
- Adjacent channel interference

Channel coding



Channel coding



Error detection

Error repeat reQuest

Error correction

Block codes

Block codes

Convolution codes

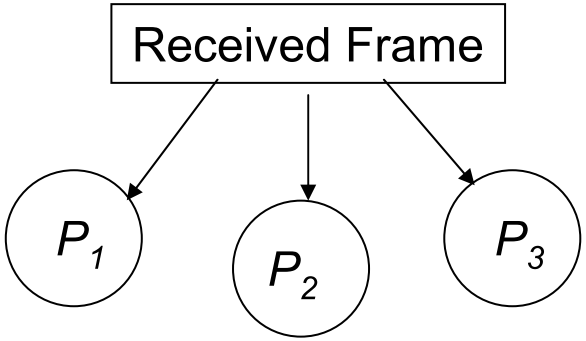
Channel coding

$\boldsymbol{P_e}$: bit error probability, or more commonly bit error rate (BER).

$\boldsymbol{P_1}$: The probability that a transmitted frame contains no error.

$\boldsymbol{P_2}$: Probability that, with an error detection algorithm in use, a transmitted frame arrives with one or more undetected errors.

$\boldsymbol{P_3}$: Probability that, with an error detection algorithm in use, a frame arrives with one or more detected bit errors but no undetected bit errors.



$$P_1=(1 - P_e)^k$$

$$P_2=1 - P_1$$

Channel coding

- The probability that a frame arrives with no bit errors decreases when the probability of a single bit error increases.
- The probability that a frame arrives with no bit error decreases with increasing frame length.

Example (in the book)

Consider that the BER on a 64 kpbs channel should be less than $\mathbf{10^{-6}}$

Suppose now that we have the requirement that on average one frame with undetected bit error should occur per day on a continuously used 64 kpbs channel.

A frame length of 1000 bits is assumed. The total amount of frames transmitted on one day is then:

$(64000*3600*24)/1000 = 5.529*10^6$

The maximum frame error rate then becomes:

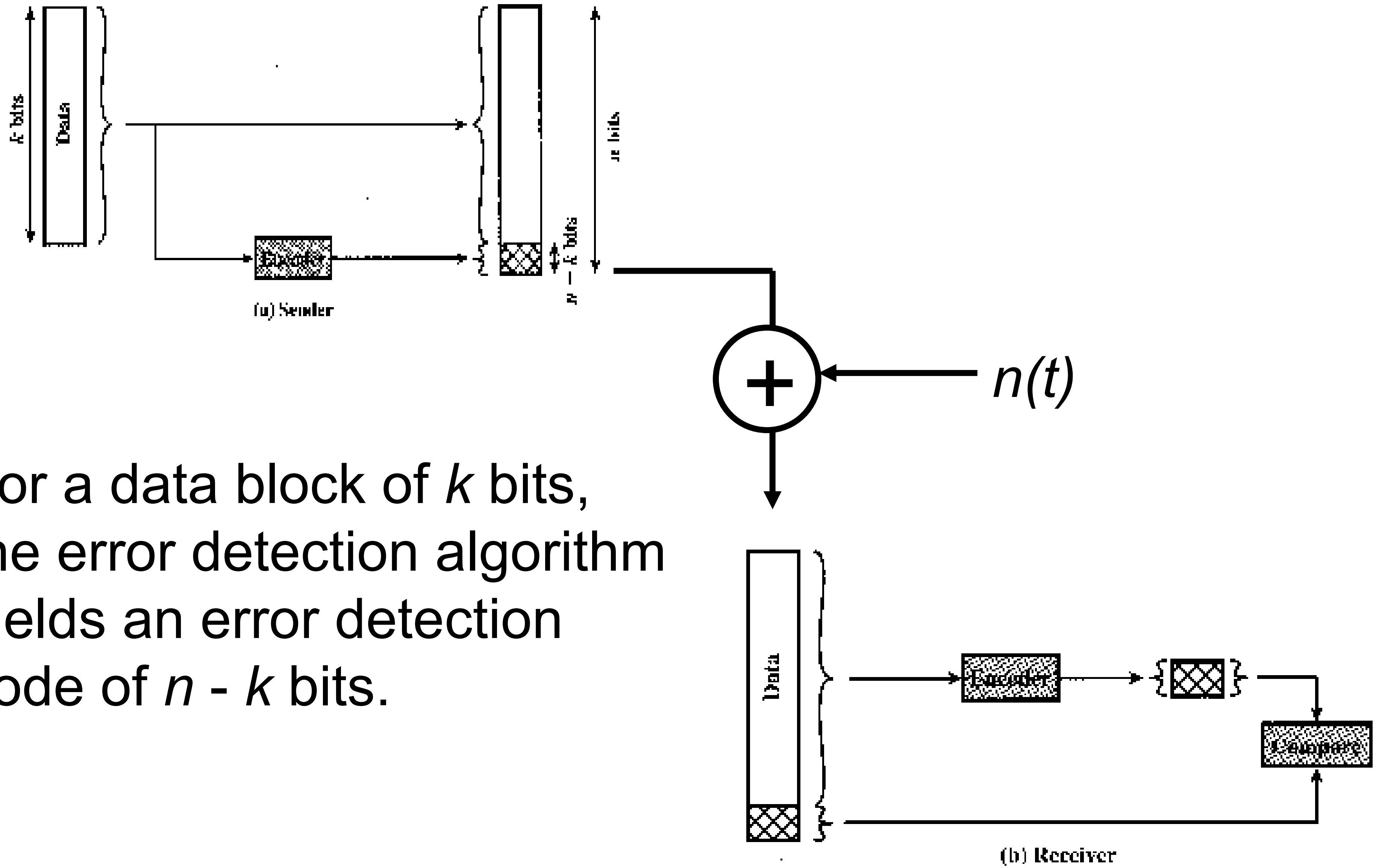
$1/(5.529*10^6) = 0.18*10^{-6}$

The actual frame error rate is however:

$P_1 = (0.999999)^{1000} = 0.999$

which is about three orders of magnitude too large to meet our requirements.

Channel coding



(a) Sender

(b) Receiver

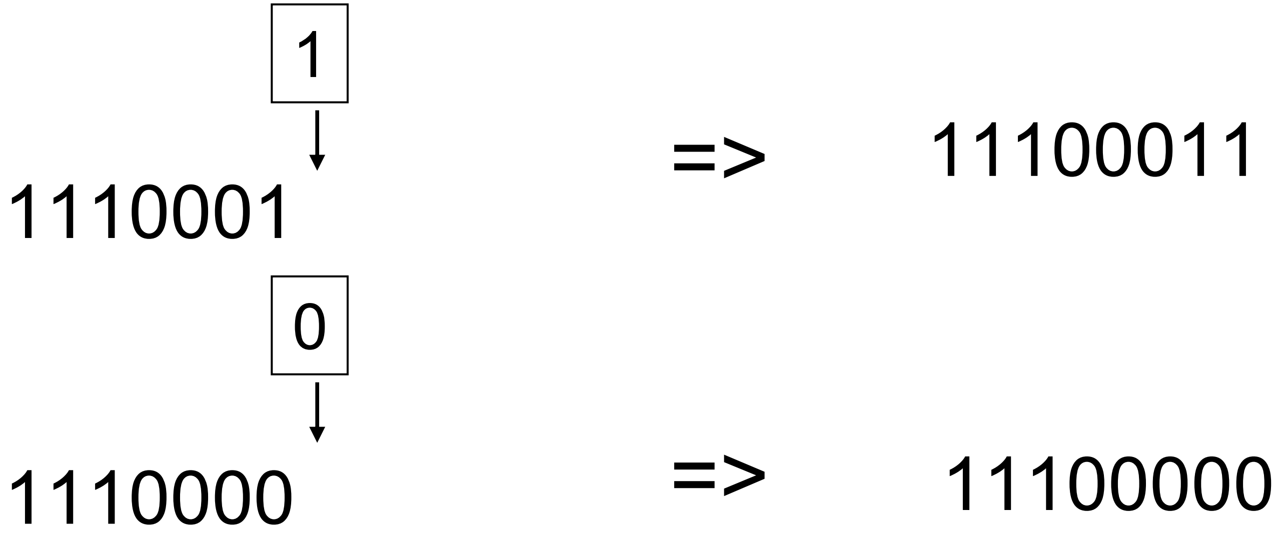
For a data block of *k* bits, the error detection algorithm yields an error detection code of *n* - *k* bits.

Parity check

- The parity check appends a parity bit to the end of a block of data.
- A typical example is character transmission, in which a parity is attached to each 7-bit character.
- The value of this bit is selected so that the character has an even number of 1s (***even parity***) or an odd number of 1s (***odd parity***).

Error detection

Parity check (odd parity)



1

1110001 => 11100011

0

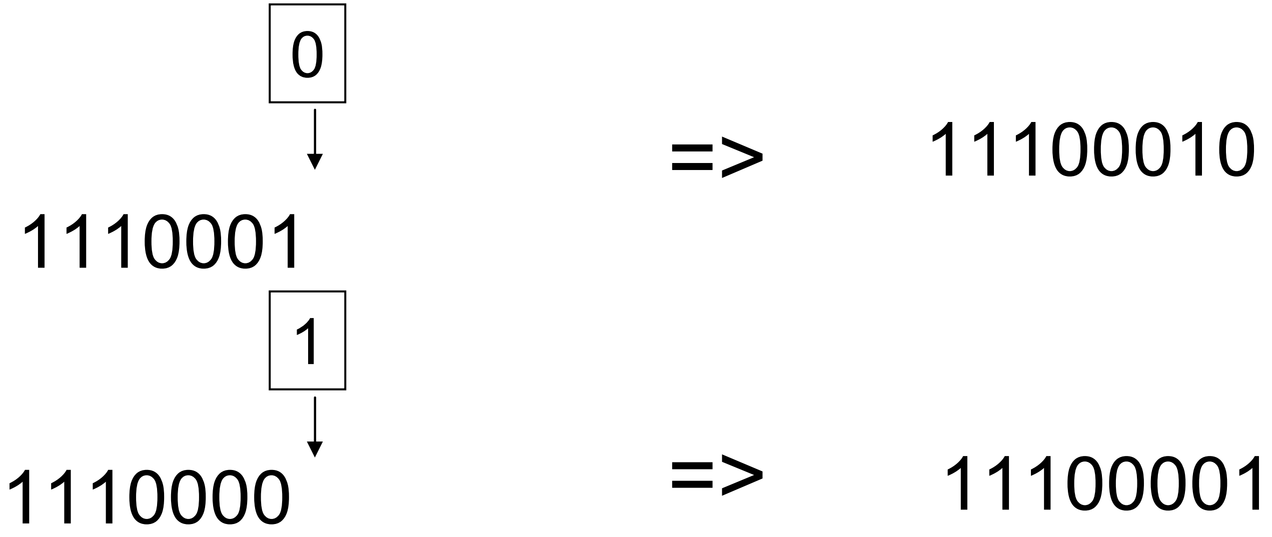
1110000 => 11100000

The receiver examines the received characters and, if the total numbers of 1s is odd it assumes that no error has occurred.

If one or any odd numbers of bits are inverted during the transmission, then the receiver will detect an error, e.g. 11$\underline{0}$00011, 11$\underline{0}$00000.

However, if two (or any even number) of bits are inverted an undetected error occurs.

Parity check (even parity)



The receiver examines the received characters and, if the total numbers of 1s is even it assumes that no error has occurred.

If one or any odd numbers of bits are inverted during the transmission, then the receiver will detect an error, e.g. 11000010, 11000001.

However, if two (or any even number) of bits are inverted an undetected error occurs.

Cyclic redundancy check (CRC)

- The most common error detecting codes is the cyclic redundancy check (CRC).
- Given a *k* bit block of bits, the transmitter generates an (n - k) bit sequence, known as the frame check sequence (FCS).
- The resulting frame consists of *n* bits, that is exactly devisable by some predetermined number.
- The receiver then divides the incoming frame by that number and, if there is no remainder, it assumes that there was no error.

CRC

Three different way to present the same thing:

- Modulo 2 arithmetic
- Polynomials
- Digital logic

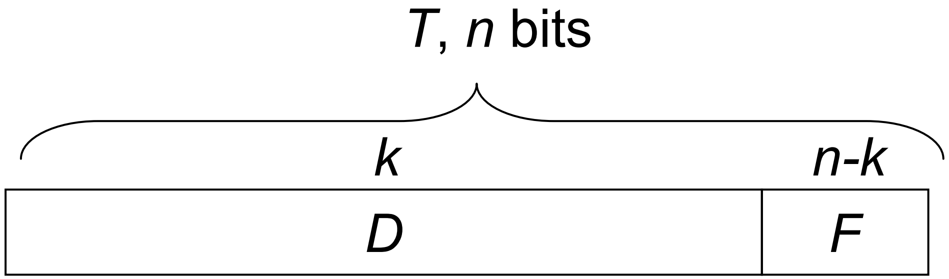
Definitions:

T = *n* bit frame to be transmitted.

D = *k* bit block of data, the first *k* bits of *T.*

F = *n* - *k* bit FCS, the last *n* – *k* bits of *T.*

P pattern of *n* - *k* bits, predetermined divisor.



Modulo 2 arithmetic

In coding theory many mathematical operations is performed in finite number systems with certain algebraic constructions.

A simple but often used arithmetic is binary arithmetic:

+	0	1
0	0	1
1	1	0

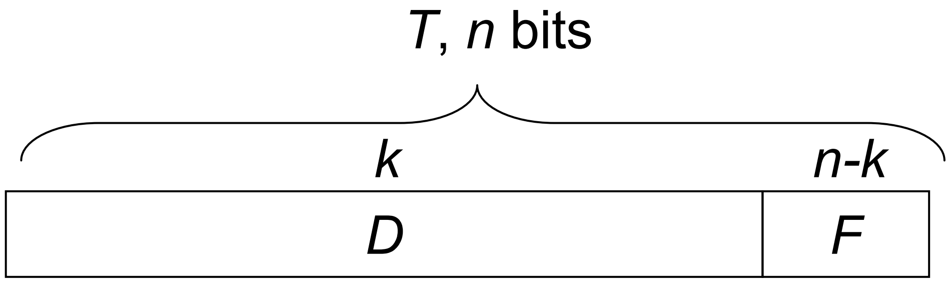
*	0	1
0	0	0
1	0	1

Example:

$$\begin{array}{r} 1111\\ +\ \underline{\ 1010}\\ 0101\end{array} \qquad \begin{array}{r} 1111\\ -\underline{\ 0101}\\ 1010\end{array} \qquad \begin{array}{r} 11001\\ \underline{\text{�}\quad\ \ 11}\\ 11001\\ \underline{11001}\\ 101011\end{array}$$

Observe that in binary arithmetic is ‘1’ both positive and negative, this means that 1+1 and 1-1 the same and that 1*1=1. For addition modulo 2 the symbol ⬑ is used.

CRC – Modulo 2 arithmetic



We would like *T*/*P* to have no remainder, where *P* is the predefined pattern.

Mathematical description:

$$T = 2^{n-k} D + F$$

By multiplying *D* by 2^{n-k}, *D* is ***shifted to the left*** by *n – k* and padded with zeros. Adding *F*, ***concatenates*** *D* and *F*, which is *T.*

CRC – Modulo 2 arithmetic

T should be exactly devisable by P, lets start with:

$$\frac{2^{n-k}D}{P} = Q + \frac{R}{P}$$

There is a ***quotient,*** *Q,* and a ***remainder,*** *R*. Because division is modulo 2, the remainder will always be at least one bit shorter than the divisor. The remainder is used as FCS, then:

$$T = 2^{n-k}D + R$$

This remainder, *R*, must satisfy our condition that the *T*/*P* has no remainder:

$$\frac{T}{P} = \frac{2^{n-k}D + R}{P} = \frac{2^{n-k}D}{P} + \frac{R}{P}$$

CRC – Modulo 2 arithmetic

By substituting the previous equation we get:

$$\frac{T}{P} = Q + \frac{R}{P} + \frac{R}{P}$$

A binary number added to itself modulo 2 is zero thus:

$$\frac{T}{P} = Q + \frac{R + R}{P} = Q$$

There is no remainder and thus *T* is exactly divisible by *P*.

The FCS is then easily generated at the transmitter, divide $2^{n-k}D$ by *P* and use the *n* - *k* bit remainder as the FCS.

On reception, the receiver will divide *T* by *P* and will get no remainder if there have been no errors.

CRC – Example (page 208)

Given:

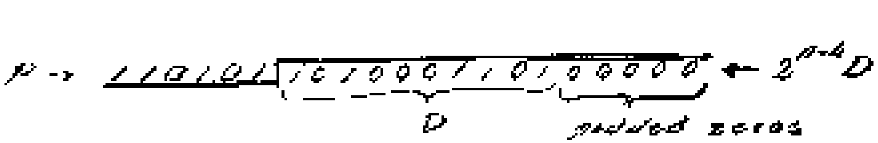
Message, D = 1010001101 (10 bits)

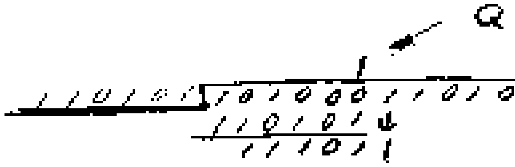
Pattern, P = 110101 (6 bits)

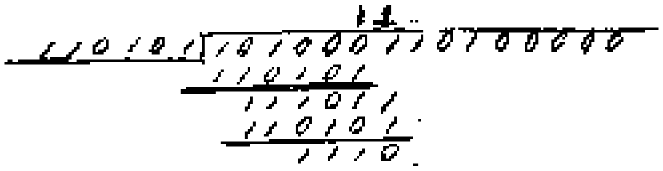
FCS, R = to be calculated

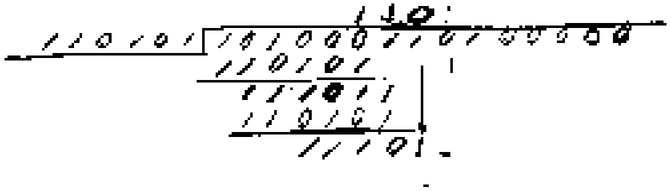
Thus, n = 15 and k = 10 and $n - k = 5$

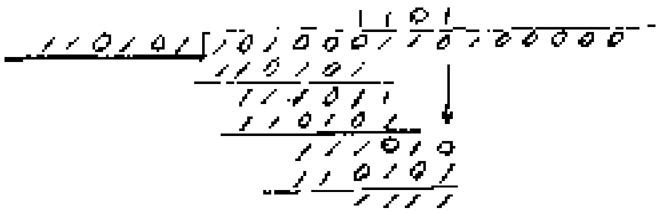
=>D is multiplied with 2^{n-k}, shifted 5 steps to the left, equal to: 101000110100000

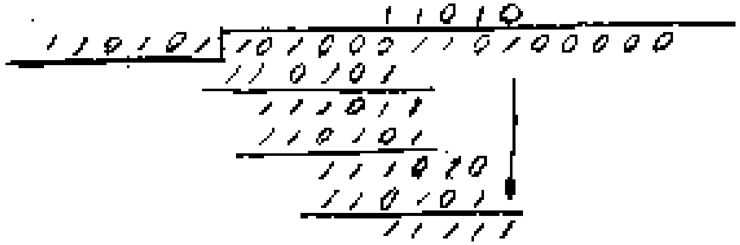

P →
110101
101000110100000
← 2^{n-k}D
D
padded zeros

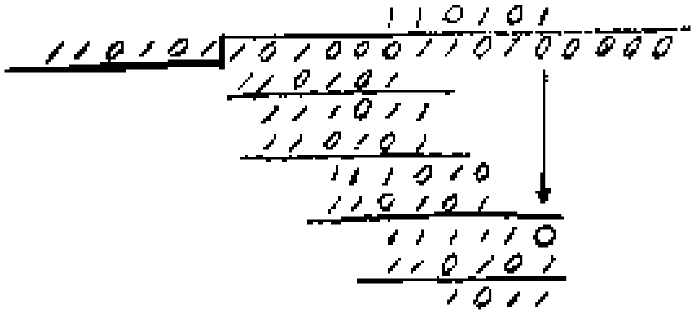

Q
1

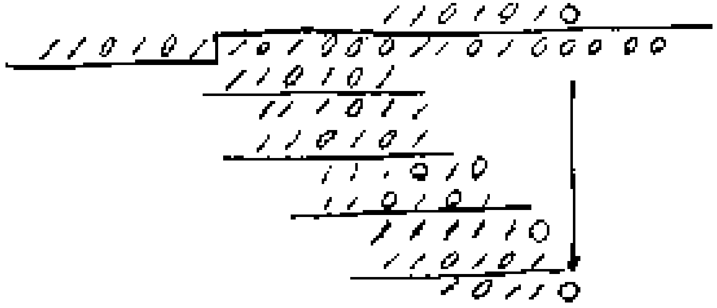

11

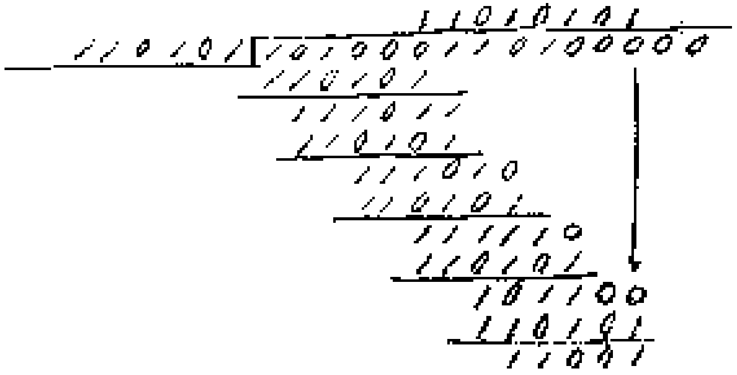

110


1101


11010


110101


1101010


110101101

```
              110101011
110101 | 101000110100000
         110101
          111011
          110101
            111010
            110101
              111110
              110101
                101100
                110101
                 110010
                 110101
                    0111
```

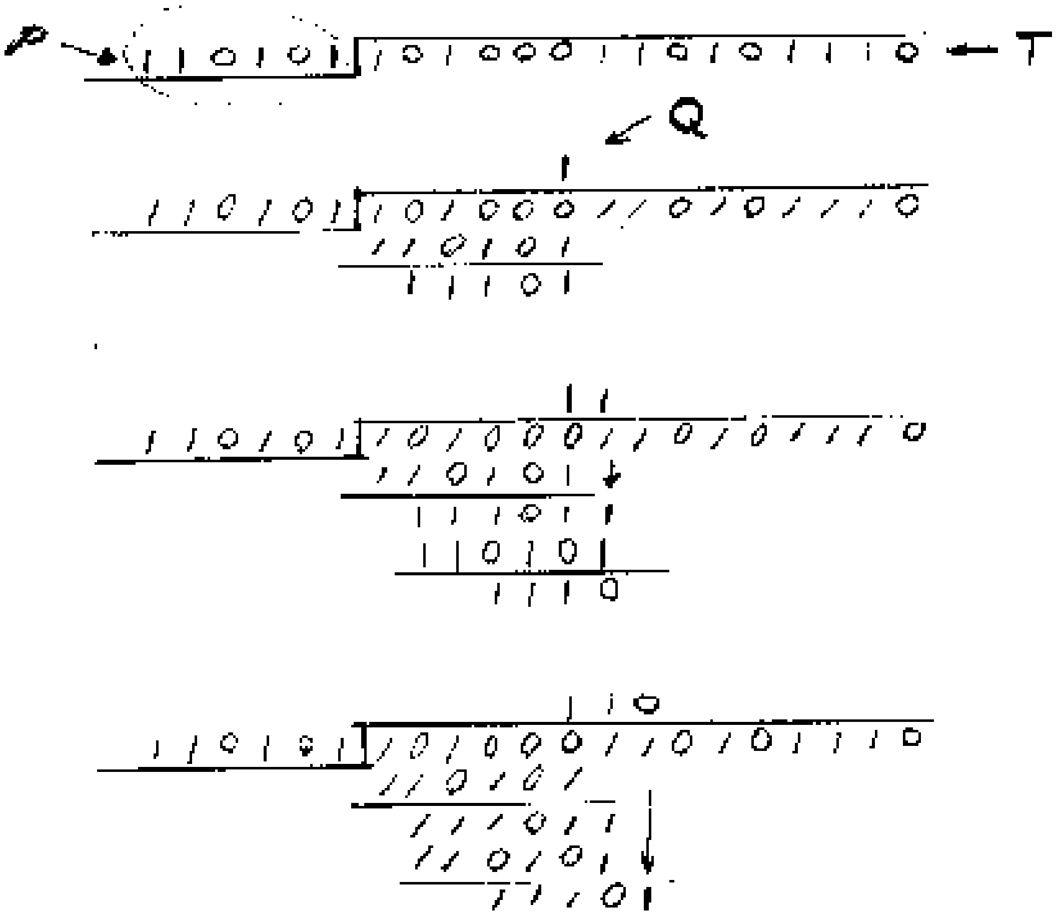
```
              1101010110
110101 | 101000110100000
         110101
          111011
          110101
            111010
            110101
              111110
              110101
                101100
                110101
                 110010
                 110101
                   01110
                   \___/
                     R
```

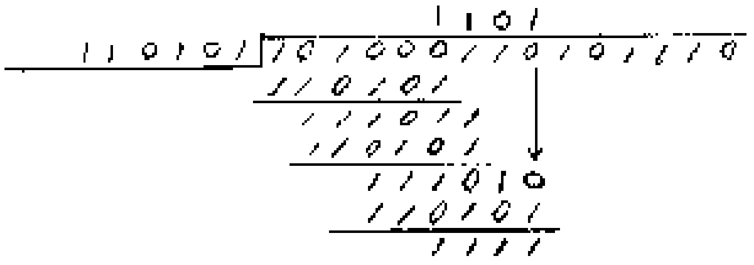
The reminder is added to $2^{n-k}D$, $T = 2^{n-k}+R$

```
T ⇒ + 101000110100000
                01110
      ---------------
      101000110101110
```

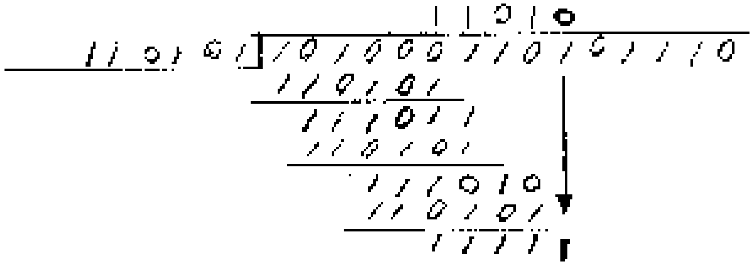
CRC – Example

If there are no errors, the receiver receives *T* intact. The received frame is then divided by *P*.

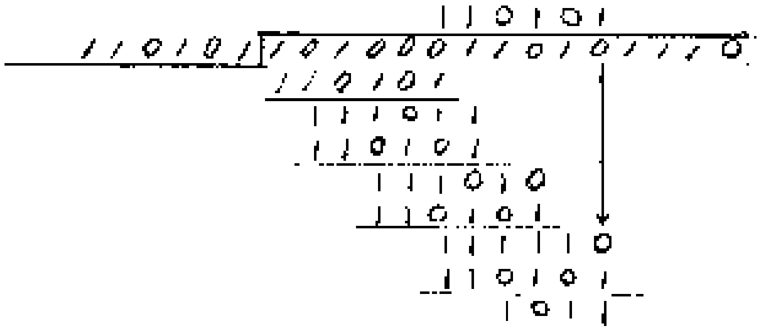




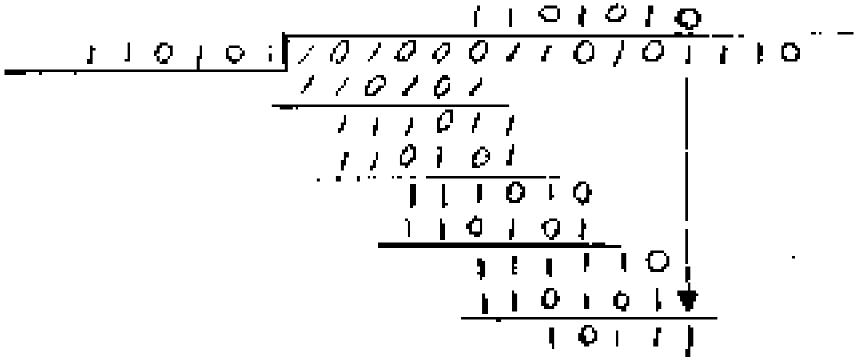
```
              1101
110101|101000110101110
       110101
        111011
        110101
          111010
          110101
            1111
```



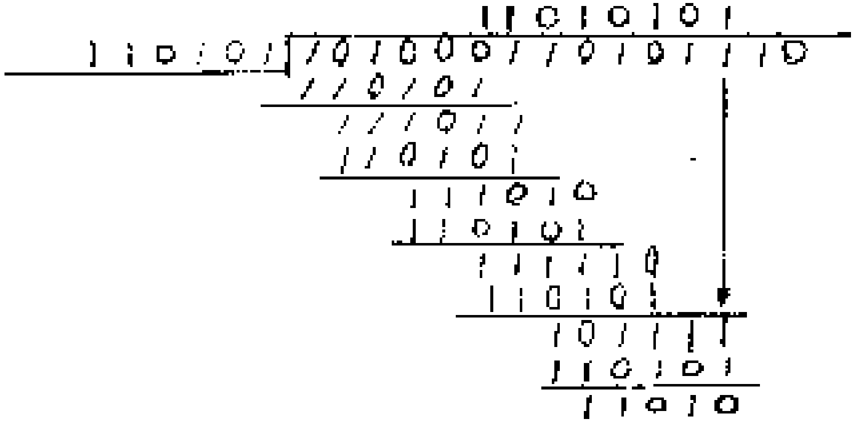
```
              11010
110101|101000110101110
       110101
        111011
        110101
          111010
          110101
            11111
```



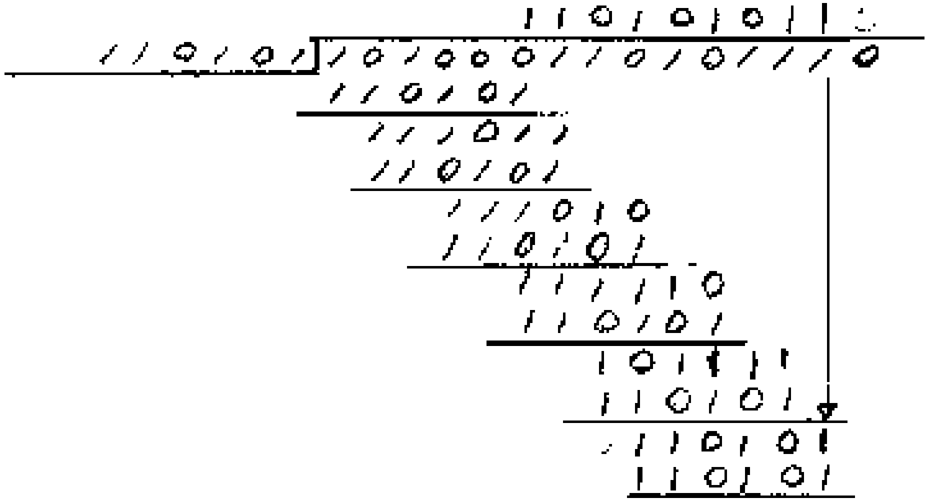
```
              110101
110101|101000110101110
       110101
        111011
        110101
          111010
          110101
            111110
            110101
              1011
```

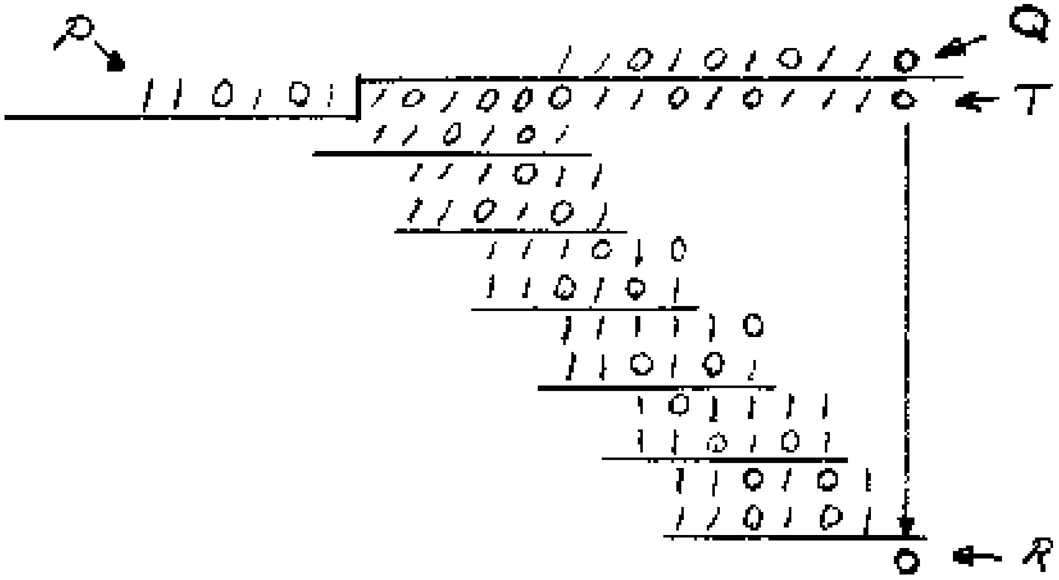


```
              1101010
110101|101000110101110
       110101
        111011
        110101
          111010
          110101
            111110
            110101
              10111
```



```
              11010101
110101|101000110101110
       110101
        111011
        110101
          111010
          110101
            111110
            110101
              101111
              110101
               11010
```




P
Q
T
R

CRC

The pattern, *P*, is chosen to be one bit longer than the desired FCS.

The exact bit pattern chosen depends on the type of errors expected.

At minimum, both the high- and low order bits of *P* must be ’1’.

An error result in the **inversion** of a bit, this is equivalent to taking the **XOR** of the bit and 1 (modulo 2 addition of ’1’ to the bit): **0+1=1**, **1+1=0**.

The errors in a frame can be represented by an *n* bit field with ’1’s in each error position

The resulting frame, T_r, can be expressed as:

$$T_r = T \oplus E$$

where *T*= transmitted frame, *E* = error pattern with 1s in position where errors occur and T_r= received frame

CRC

- If there is an error ($E \neq 0$), the receiver will fail to detect the error if T_r is divisible by P which is equivalent to E divisible by P.
- Intuitively this seems as an unlikely occurrence

CRC – polynomials

- A second way of viewing the CRC process is to express all values as polynomials in a dummy variable X, with binary coefficients.
- The coefficients correspond to the bits in the binary number.
- Arithmetic operations are again modulo 2.
- The CRC process is now expressed as:

$$\frac{X^{n-k}D(X)}{P(X)} = Q(X) + \frac{R(X)}{P(X)} \qquad T(X) = X^{n-k}D(X) + R(X)$$

CRC – polynomials – Example

Using the previous example, in polynomial form

D = 1010001101 => $D(X) = X^9+X^7+X^3+X^2+1$

P = 110101 => P(X) = $X^5+X^4+X^2+1$

R = 01110 => $R(X) = X^3+X^2+X$

CRC – polynomials – Example

$$
\begin{array}{rl}
 & X^9+X^8+X^6+X^4+X^2+X \qquad \leftarrow Q(X) \\
P(X) \longrightarrow X^5+X^4+X^2+1 & \overline{)\,X^{14} \qquad X^{12} \qquad X^8+X^7+ \qquad X^5} \qquad \leftarrow 2^3D(X) \\
 & \underline{X^{14}+X^{13}+ \quad X^{11}+ \quad X^9} \\
 & \quad X^{13}+X^{12}+X^{11}+ \quad X^9+X^8 \\
 & \quad \underline{X^{13}+X^{12}+ \quad X^{10}+ \quad X^8} \\
 & \qquad X^{11}+X^{10}+X^9+ \quad X^7 \\
 & \qquad \underline{X^{11}+X^{10}+ \quad X^8+ \quad X^6} \\
 & \qquad\quad X^9+X^8+X^7+X^6+X^5 \\
 & \qquad\quad \underline{X^9+X^8+ \quad X^6+ \quad X^4} \\
 & \qquad\qquad X^7+ \quad X^5+X^4 \\
 & \qquad\qquad \underline{X^7+X^6+ \quad X^4+ \quad X^2} \\
 & \qquad\qquad\quad X^6+X^5+ \qquad X^2 \\
 & \qquad\qquad\quad \underline{X^6+X^5+ \quad X^3+ \quad X} \\
 & \qquad\qquad\qquad X^3+X^2+X \qquad \leftarrow R(X)
\end{array}
$$

CRC – polynomials

An error *E(x)* will only be undetectable if it is divisible by *P(X).*

All of the following errors are not divisible by a suitably chosen *P(x)* and hence are detectable:

- All single errors, if *P(x)* has more than one nonzero term.
- All double bit errors, as long as *P(x)* has a factor with three terms.

- All single errors, if *P(x)* has more than one nonzero term.
- All double bit errors, as long as *P(x)* has a factor with three terms

CRC – polynomials

- Any odd number of errors, as long as *P(x)* contains a factor (*X*+1).
- Any burst error for which the length of the burst is less than or equal to *n* - *k* (less than or equal to the length of the FCS).
- A fraction of an error burst of length *n* - *k* +1
- A fraction of an error burst of length greater than *n* - *k*+1, where the fraction equals 1 - $2^{-(n-k)}$.
- If all error pattern are considered equally likely, then for a burst error of length *r* +1, the probability of an undetected is $1/2^{r-1}$.
- For a longer burst the probability is $1/2^{r}$, where *r* is the length of the FCS.

CRC – polynomials

Four versions of widely used polynomials patterns, P(x) is:

CRC-12 = $X^{12}+X^{11}+X^{3}+X^{2}+X+1$

CRC-16 = $X^{16}+X^{15}+X^{2}+1$

CRC-CCITT = $X^{16}+X^{12}+X^{5}+1$

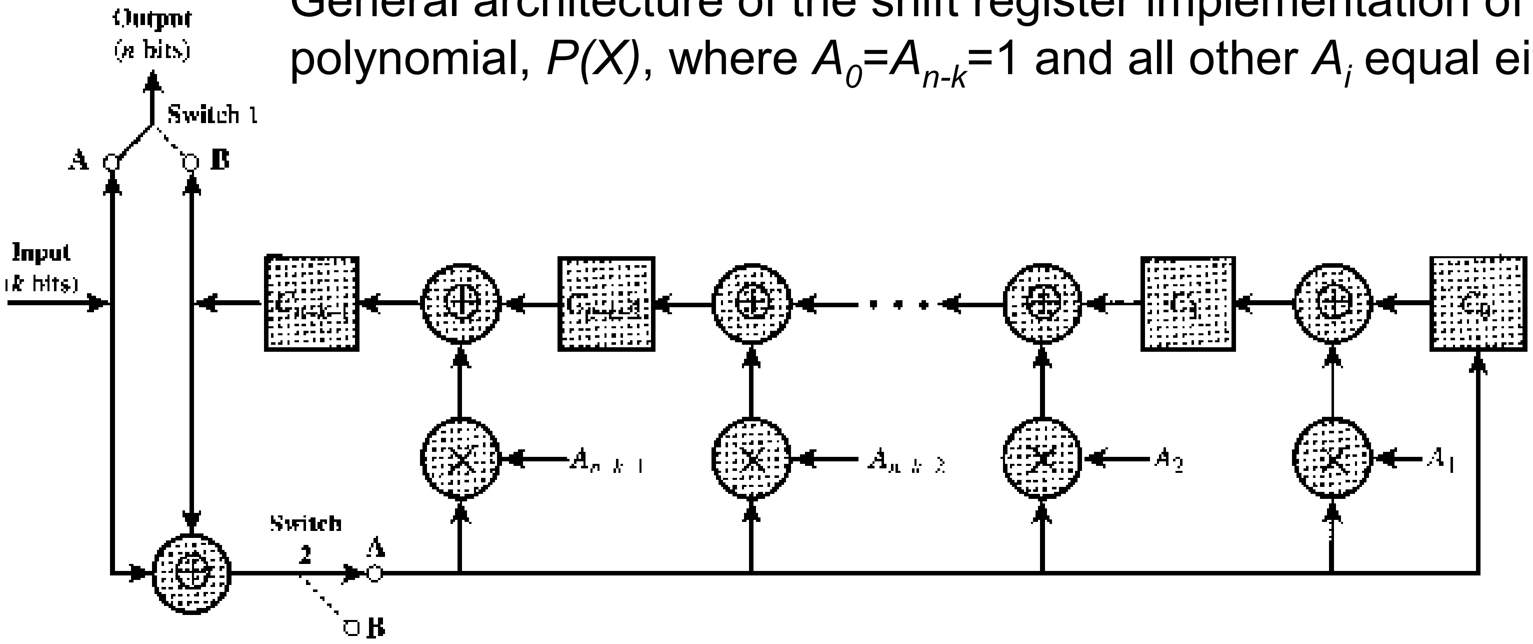
CRC-32 = $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$

CRC – digital logic

The CRC process can be implemented, as a dividing circuit consisting of XOR gates and a shift register.

1. The ***shift register*** contains *n* - *k* bits, equal to the length of the FCS.
2. There are up to *n* - *k* ***XOR gates***.
3. The ***presence or absence of a gate corresponds to the polynomial***, *P(x)*, excluding the terms ***1*** and $\boldsymbol{X^{n-k}}$.

General architecture of the shift register implementation of a CRC for the polynomial, *P(X)*, where A_0=A_{n-k}=1 and all other A_i equal either '0' or '1'.



$$P(X) = \sum_{i=0}^{n-k} A_i X^i$$

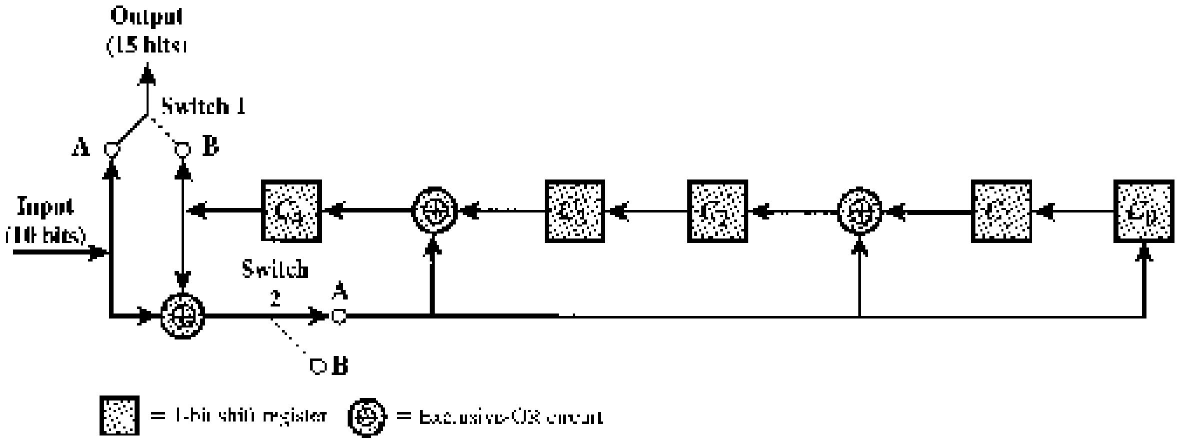
CRC – digital logic – example

Data D=1010001101=>

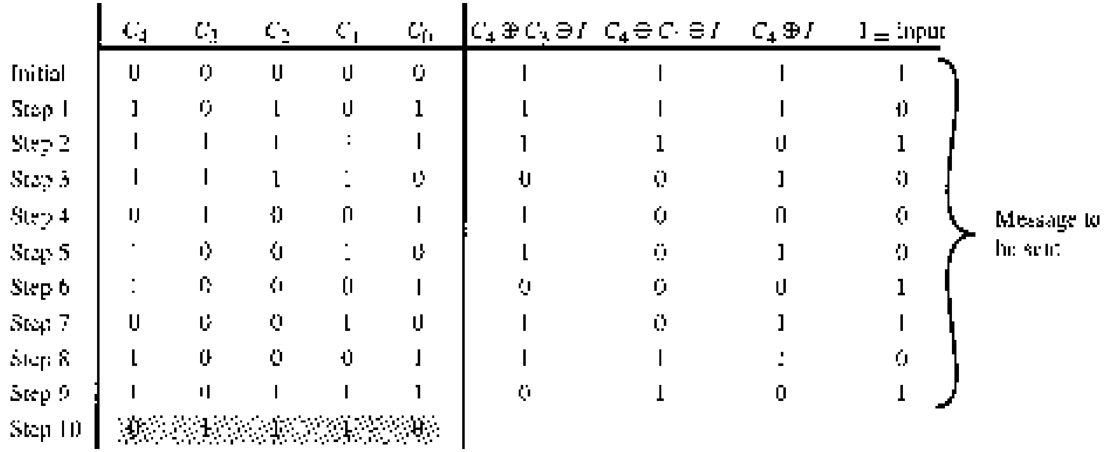
$D(X) = X^9+X^7+X^3+X^2+1$

Divisor P =110101=>

$P(X) = X^5+X^4+X^2+1$



(a) Shift register implementation



	C_4	C_3	C_2	C_1	C_0	$C_4 \oplus C_3 \oplus I$	$C_4 \oplus C_1 \oplus I$	$C_4 \oplus I$	I = input
Initial	0	0	0	0	0	1	1	1	1
Step 1	1	0	1	0	1	1	1	1	0
Step 2	1	1	1	1	1	1	1	0	1
Step 3	1	1	1	1	0	0	0	1	0
Step 4	0	1	0	0	1	1	0	0	0
Step 5	1	0	0	1	0	1	0	1	0
Step 6	1	0	0	0	1	0	0	0	1
Step 7	0	0	0	1	0	1	0	1	1
Step 8	1	0	0	0	1	1	1	1	0
Step 9	1	0	1	1	1	0	1	0	1
Step 10	0	1	1	1	0				

Message to be sent

(b) Example with input of 1010001101

CRC – digital logic – example

- To produce the proper output two switches are used.
- The input data bits are fed in with both switches in position A.
- The result is that for the 10 first steps, the input bits are fed into the shift register and also used as output bits.
- After the last data bit is processed, the shift register contains the remainder (FCS).
- As soon as the last data bit is provided to the shift register , both switches are set to the B position.
- This has two effects (1) all of the XOR gates becomes pass through (no bits are changed), and (2) as the shifting process continuous, the 5 CRC bits are output.