Experiment No : 07

Aim : Study & implementation of cyclic redundancy check (CRC).

Theory : This is a type of polynomial code in which a bit string is represented in the form of polynomials with coefficients of 0 &1 only. Polynomial arithmetic uses a modulo-2 arithmetic i.e. addition & subtraction are identical to EXOR. For CRC code the sender & receiver should agree upon a generator polynomial G(x). A codeword can be generated for a given data word (message) polynomial M(x) with the help of long division. This technique is more powerful than parity check & checksum error detection. CRC is based on binary division. A sequence of redundant bits called CRC or CRC remainder is appended at the end of the data unit such as byte. The resulting data unit after adding CRC remainder becomes exactly divisible by another predetermined binary number. At the receiver this data is divided by the same binary number. There is no error if this division does not yield any remainder. But a non-zero remainder indicates presence of errors in the received data unit. Such an erroneous data unit is then rejected.

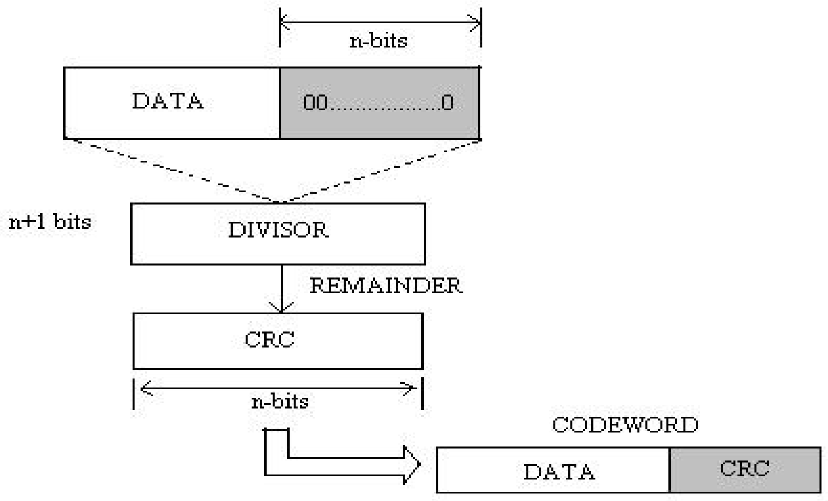
Procedure to obtain CRC:

The redundancy bits used by CRC are derived by following the procedure given below:

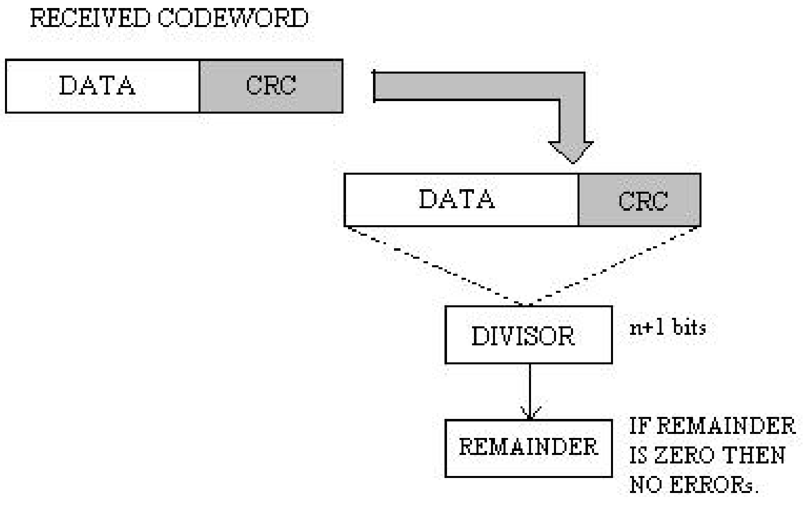
(1) Divide the data unit by a predetermined divisor.
(2) Obtain the remainder. It is the CRC.

Requirements of CRC:

(1) It should have exactly one bit less than divisor.
(2) Appending the CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor.



CRC GENERATOR

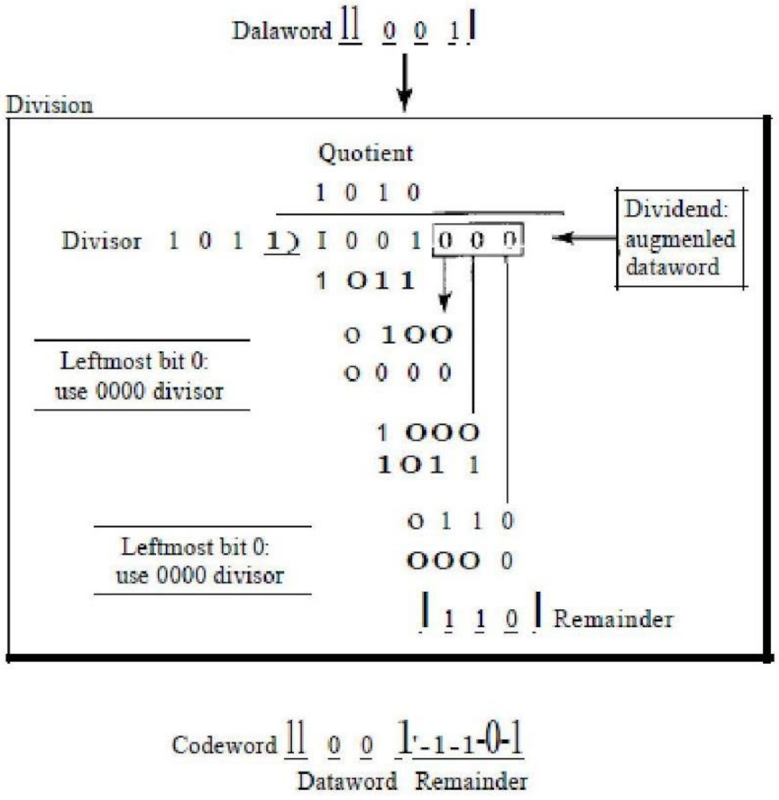


CRC CHECKER

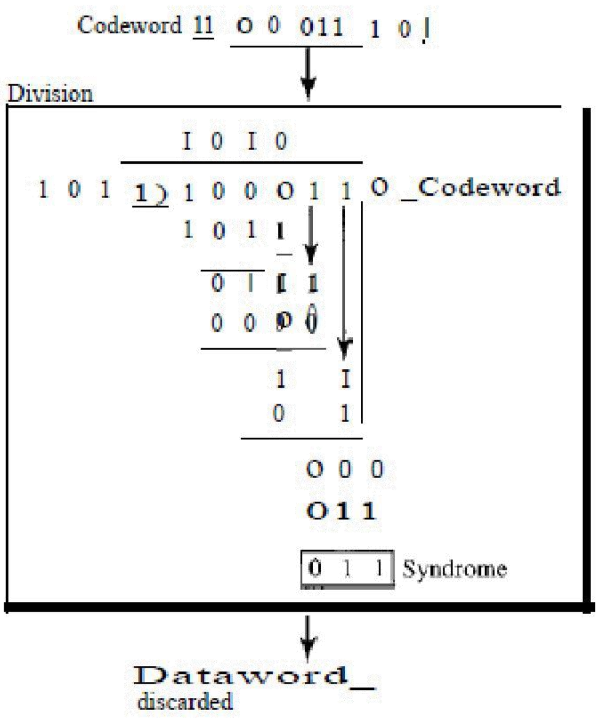
EXAMPLE OF CRC CODE WITH C(7,4)

A CRC code with C(7, 4)

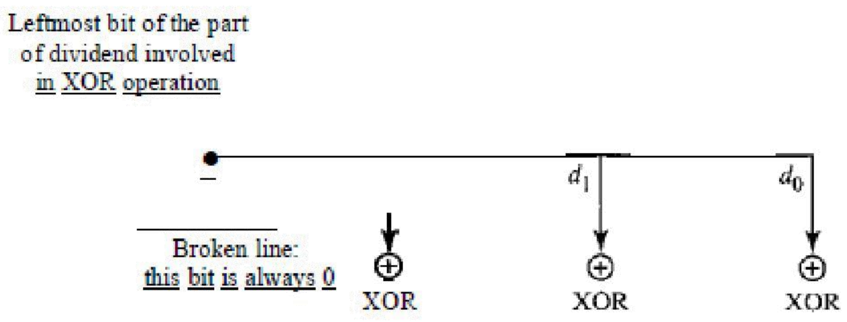
Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111



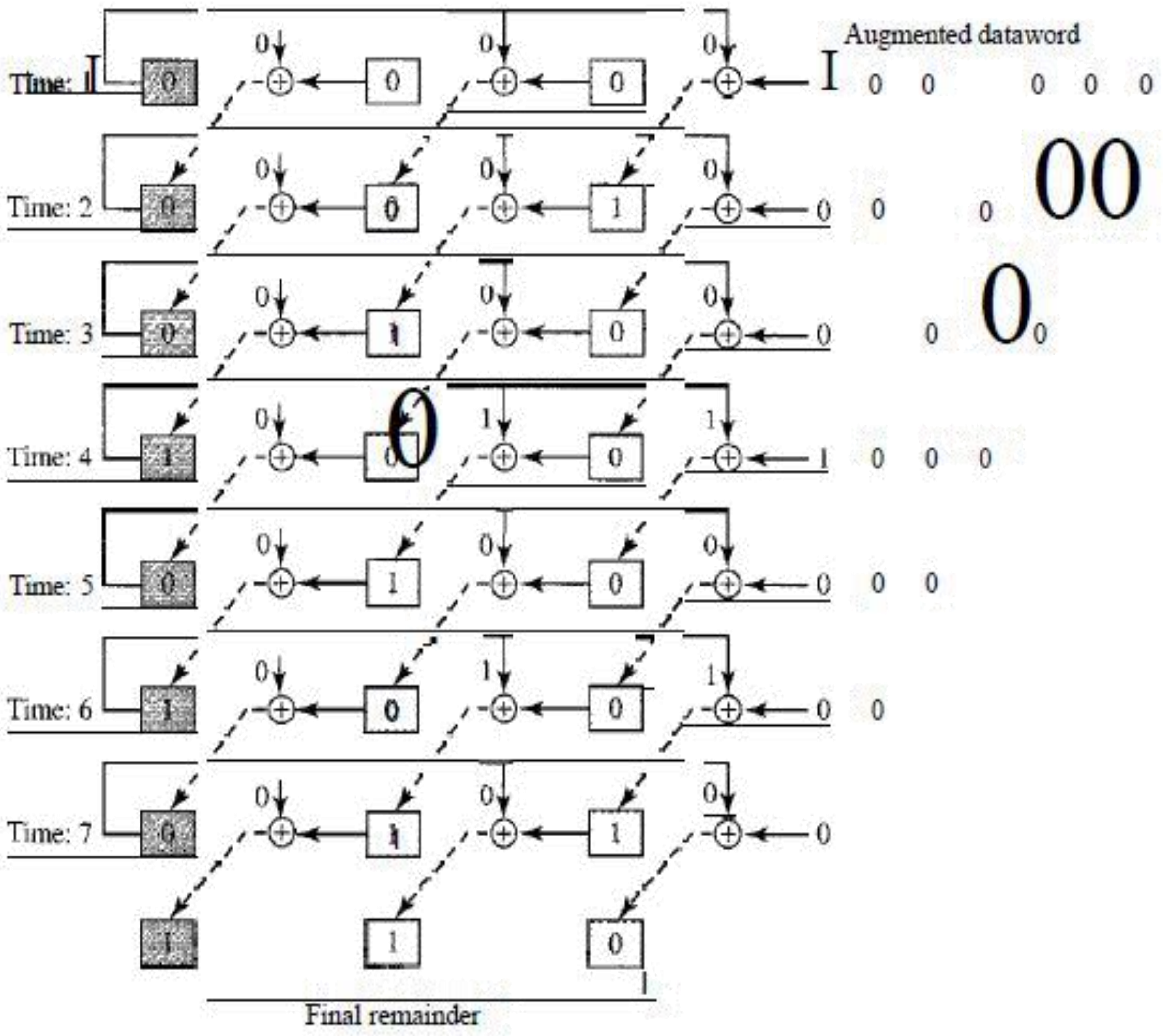
Division in CRC encoder



Division in the CRC decoder



Hardwired design of the divisor in CRC



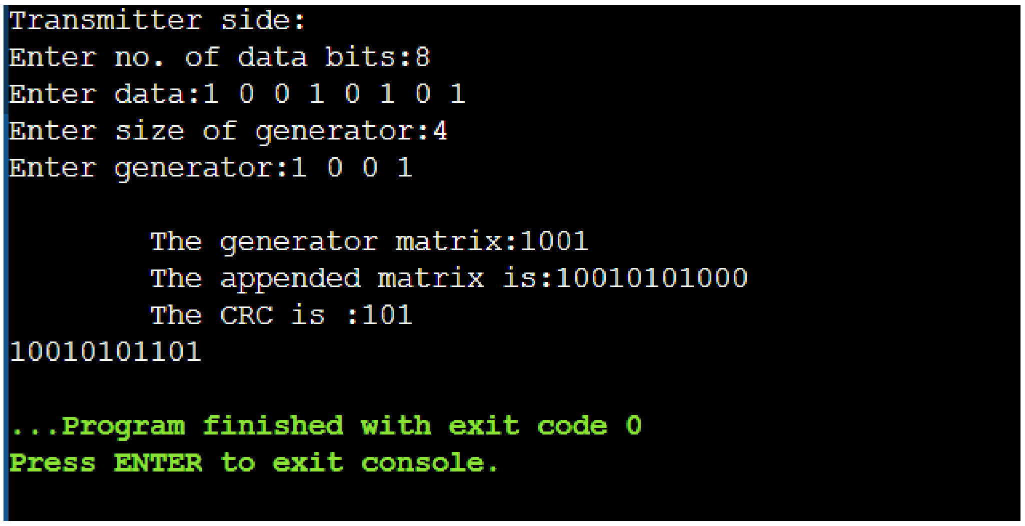
Simulation ofdivision in CRC encoder

Code :

```
#include<stdlib.h>

#include<conio.h>

#include<stdio.h>

using namespace std;

int main()

{

int i,j,n,g,a,arr[20],gen[20],b[20],q[20],s;

printf("Transmitter side:");

printf("\nEnter no. of data bits:");

scanf("%d",&n);

printf("Enter data:");

for(i=0;i< n;i++)

scanf("%d",&arr[i]);


printf("Enter size of generator:"); scanf("%d",&g);

do{

printf("Enter generator:");

for(j=0;j< g;j++)

scanf("%d",&gen[j]);


}

while(gen[0]!=1);

printf("\n\tThe generator matrix:");
```

```
for(j=0;j< g;j++)

printf("%d",gen[j]);

a=n+(g-1);

printf("\n\tThe appended matrix is:"); for(i=0;i< j;++i)

arr[n+i]=0;

for(i=0;i< a;++i)

printf("%d",arr[i]);

for(i=0;i< n;++i)

q[i]= arr[i];


for(i=0;i< n;++i)

{

if(arr[i]==0)

{

for(j=i;j< g+i;++j)

arr[j] = arr[j]^0;

}

else

{

arr[i] = arr[i]^gen[0];

arr[i+1]=arr[i+1]^gen[1];

arr[i+2]=arr[i+2]^gen[2];

arr[i+3]=arr[i+3]^gen[3];

}
}
printf("\n\tThe CRC is :");
for(i=n;i < a;++i)
printf("%d",arr[i]);
s=n+a;
for(i=n;i< s;i++)
q[i]=arr[i];
printf("\n");
for(i=0;i< a;i++)
printf("%d",q[i]);
getch();
}
```

Output :



```
Transmitter side:
Enter no. of data bits:8
Enter data:1 0 0 1 0 1 0 1
Enter size of generator:4
Enter generator:1 0 0 1

        The generator matrix:1001
        The appended matrix is:10010101000
        The CRC is :101
10010101101

...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion : - Hence we have successfully study and implemented the program of cyclic redundancy check (CRC).