THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXP - 2

CN - C32 - 2103164

AIM - Implementation of Hamming Code for Error correction and detection

Theory :- The Hamming Code is a widely used for error correction and detection in digital communication. The theory behind the Hamming Code involves adding parity bits to data stream to detect and correct error. It works as follows :-

Data Encoding :- For input message of 'K' bits, 'r' parity bits are added at positions that are powers of 2 (1, 2, 4, 8... etc). The positions of these parity bits are calculated by using binary representation. Each parity bit is responsible for checking subset of data bits.

Parity Calculation :- Each parity bit calculates its value based on data bits that it covers. For eg, parity bit at position 1 checks all data bits that have a 1 in their least significant bit and so on. Parity bits are calculated using even parity, which means total numbers of 1s in group of bits should be even.

Error Detection :- During transmission, if a bit get flipped due to noise, the parity bits will no longer match their calculated values. This discrepancy indicates an error

Error Correction :- If an error is detected, the erroneous bits position can be determined by looking at parity bits

that cover that position. By using the pattern of incorrect parity bits, the erroneous bits position ca identified and corrected.

Decoding:- The received data is compared to parity If any discrepancies are found, error detection correction are performed based on parity bits positi

Its limitations, is it cannot correct more tha two errors or detect error in parity bits the

Problem :- A 7 bit Hamming code is received as Assume Even parity & state whether the recei is correct or wrong, if wrong locate the bit in

Solution:- Received HC :-

D7	D6	D5	P4	D3	P2	P1
1	0	1	1	0	1	1

Detecting errors :-

Step 1: Analyzing bits 1, 3, 5, 7

We have $P_1 D_3 D_5 D_7 \Rightarrow 1011$ { odd pa error e

We put $P_1 = 1$

EN-C32-2103164

Step 2: analyzing bit 2, 3, 6 & 7

P_2 D_3 D_6 D_7

$\Rightarrow$ 1 0 0 1 { Even parity No Errors }

$\Rightarrow P_2 = 0$.

Step 3:- Analyzing bits 4, 5, 6 & 7

we have P_4 D_5 D_6 D_7

$\Rightarrow$ 1 1 0 1 { odd parity error Exists }

we put $P_4 = 1$

<u>Correcting error</u>

error word E = | P_4 | P_2 | P_1 |
| --- | --- | --- |
| 1 | 0 | 1 |

decimal value $E = 5$ which shows that 5th bit is in error

We write the correct world by simply inverting the 5th bit

$\therefore$ Correct word = 1 0 0 1 0 1 1

27/08/23

```
option=int(input('Press 1 for generating hamming code  \nPress 2 for finding error in hamming code\n\t
Enter your choice:--\n'))
while(option>0 and option<3):  # GENERATE HAMMING CODE


    if(option == 1):
        print('Enter the data bits')
        d=input()
        data=list(d)
        data.reverse()
        c,ch,j,r,h=0,0,0,0,[]

        while ((len(d)+r+1)>(pow(2,r))):
            r=r+1

        for i in range(0,(r+len(data))):
            p=(2**c)

            if(p==(i+1)):
                h.append(0)
                c=c+1

            else:
                h.append(int(data[j]))
                j=j+1

        for parity in range(0,(len(h))):
            ph=(2**ch)
            if(ph==(parity+1)):
                startIndex=ph-1
                i=startIndex
                toXor=[]

                while(i<len(h)):
```

```
            block=h[i:i+ph]
            toXor.extend(block)
            i+=2*ph

        for z in range(1,len(toXor)):
            h[startIndex]=h[startIndex]^toXor[z]
        ch+=1

    h.reverse()
    print('Hamming code generated would be:- ', end="")
    print(int(''.join(map(str, h))))



elif(option==2): # DETECT ERROR IN RECEIVED HAMMING CODE
    print('Enter the hamming code received')
    d=input()
    data=list(d)
    data.reverse()
    c,ch,j,r,error,h,parity_list,h_copy=0,0,0,0,0,[],[],[]

    for k in range(0,len(data)):
        p=(2**c)
        h.append(int(data[k]))
        h_copy.append(data[k])
        if(p==(k+1)):
            c=c+1

    for parity in range(0,(len(h))):
        ph=(2**ch)
        if(ph==(parity+1)):

            startIndex=ph-1
```

```
        i=startIndex
        toXor=[]

        while(i<len(h)):
            block=h[i:i+ph]
            toXor.extend(block)
            i+=2*ph

        for z in range(1,len(toXor)):
            h[startIndex]=h[startIndex]^toXor[z]
        parity_list.append(h[parity])
        ch+=1
parity_list.reverse()
error=sum(int(parity_list) * (2 ** i) for i, parity_list in enumerate(parity_list[::-1]))

if((error)==0):
    print('There is no error in the hamming code received')

elif((error)>=len(h_copy)):
    print('Error cannot be detected')

else:
    print('Error is in',error,'bit')

    if(h_copy[error-1]=='0'):
        h_copy[error-1]='1'

    elif(h_copy[error-1]=='1'):
        h_copy[error-1]='0'
        print('After correction hamming code is:- ')
    h_copy.reverse()
    print(int(''.join(map(str, h_copy))))
```

```
else:
    print('Option entered does not exist')


option=int(input('Press 1 for generating hamming code  \nPress 2 for finding error in hamming code\n\t Enter your choice:--\n'))
```

OUTPUT

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS D:\CN> python -u "d:\CN\parCheck.py"
Press 1 for generating hamming code
Press 2 for finding error in hamming code
         Enter your choice:--
1
Enter the data bits
101101
Hamming code generated would be:- 1011100100
Press 1 for generating hamming code
Press 2 for finding error in hamming code
         Enter your choice:--
2
Enter the hamming code received
1001100100
Error is in 8 bit
1011100100
Press 1 for generating hamming code
Press 2 for finding error in hamming code
         Enter your choice:--
3
PS D:\CN>
```