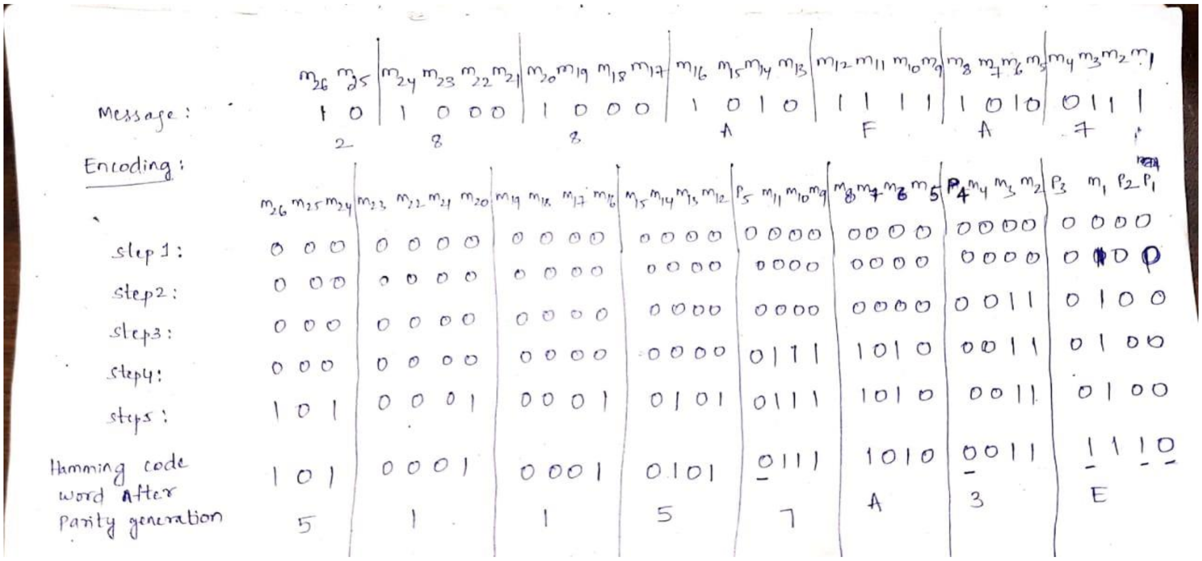
ARM PROJECT REPORT

IMPLEMENTATION OF HAMMING CODE

Objective: Implementation of (31, 26) Hamming code with even parity. Encoding 26-bit message into 31-bit code-word and decoding it back at the receiver. Single-bit error detection and correction is performed.

Hamming code

Encoding:



	m_{26} m_{25}	m_{24} m_{23} m_{22} m_{21}	m_{20} m_{19} m_{18} m_{17}	m_{16} m_{15} m_{14} m_{13}	m_{12} m_{11} m_{10} m_9	m_8 m_7 m_6 m_5	m_4 m_3 m_2 m_1
Message :	1 0	1 0 0 0	1 0 0 0	1 0 1 0	1 1 1 1	1 0 1 0	0 1 1 1
	2	8	8	A	F	A	7

Encoding :

	m_{26} m_{25} m_{24}	m_{23} m_{22} m_{21} m_{20}	m_{19} m_{18} m_{17} m_{16}	m_{15} m_{14} m_{13} m_{12}	P_5 m_{11} m_{10} m_9	m_8 m_7 m_6 m_5	P_4 m_4 m_3 m_2	P_3 m_1 P_2 P_1
step 1 :	0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
step2 :	0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0
step3 :	0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1	0 1 0 0
step4 :	0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 1 1 1	1 0 1 0	0 0 1 1	0 1 0 0
step5 :	1 0 1	0 0 0 1	0 0 0 1	0 1 0 1	0 1 1 1	1 0 1 0	0 0 1 1	0 1 0 0
Hamming code word After Parity generation	1 0 1	0 0 0 1	0 0 0 1	0 1 0 1	0 1 1 1	1 0 1 0	0 0 1 1	1 1 1 0
	5	1	1	5	7	A	3	E

Fig1. Encoding

Message: 26-bit message (m26-m1) 0x288AFA7 is taken as test input.

Step1: Initialize the 31-bit code-word register to 0.

Step2: Mask the message bit m1 with #0x1 value and shift it left by #2 to position m1 in codeword.

Step3: Mask the message bits m2-m4 with #0xE value and shift it left by #3 to position m2-m4 in codeword.

Step4: Mask the message bits m5-m11 with #0x7F0 value and shift it left by #4 to position m5-m11 in codeword.

Step5: Mask the message bits m12-m26with #0x3FFF800 value and shift it left by #5 to position m12-m26 in codeword.

Parity-bits generation:

Number of parity-bits= 31-26=5 (p1-p5)

The following bits of codeword are considered for parity-bits generation

P1-->1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31
P2-->2,3,6,7,10,11,14,15,18,19,22,23,26,27,30,31
P3-->4,5,6,7,12,13,14,15,20,21,22,23,28,29,30,31
P4-->8,9,10,11,12,13,14,15,24,25,26,27,28,29,30,31
P5-->16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31

Depending on the number of 1's in bit positions considering even parity, five parity bits are generated.

The following algorithm is used for parity bit generation.
For P1:

```
R0=Reg with message bits in correct position.
Rin=1
i=0
count=1
while( Rin<31)
{
 Loop:  Y=R0>>Rin
        carry_check();
         i=i+1
         Rin=Rin+1
         If(i<count)
           Goto loop
         Else
           { Rin=Rin+count;
              i=0;
             }
}
If(count_ones=even)
P1=0
Else
P1=1;
End.
Carry_check()
{  if (carry=1)
         Count_ones+=1;}
```

Similarly, to calculate other parity bits, the following values are initialized

P2 ---> Rin=2, Count=2
P3---> Rin=4, Count=4
P4 ---> Rin=8, Count=8
P5 ---> Rin=16, Count=16

To position 5 parity bits in code-word ,left-shift each parity-bit by count-1and OR with R0 to form the 31-bit codeword.

INTRODUCING SINGLE-BIT ERROR IN M1 POSITION:

EOR R0,R0,#4

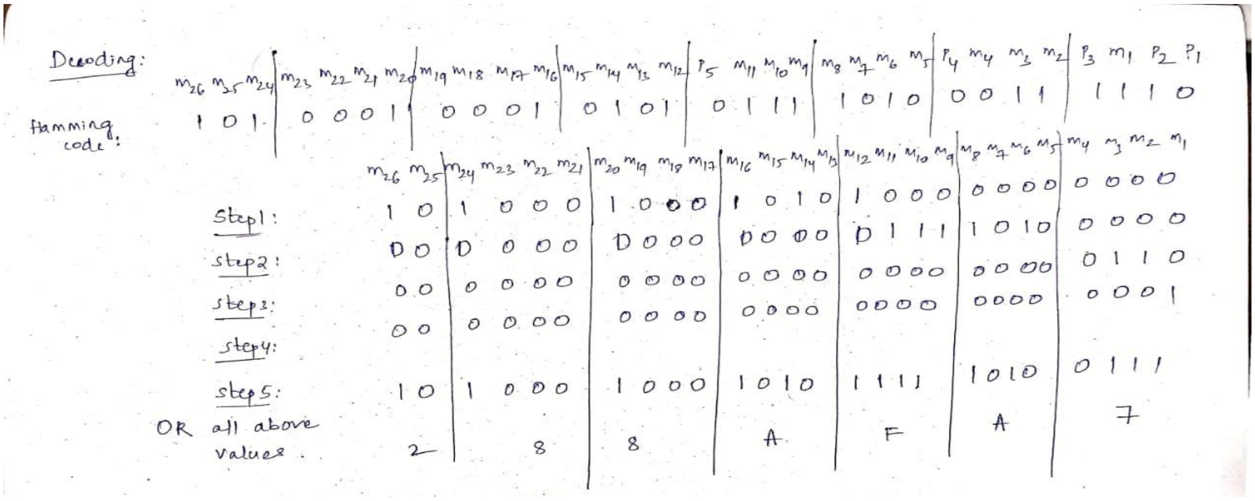
ERROR DETECTION:

Calculation of E5-E1 error parity bits similar to parity-bit generation algorithm.

Calculate the decimal equivalent. It gives the position of error bit in the received code word.

ERROR CORRECTION:
Update a register with #1. Shift it left by (decimal equivalent-1) .
To get the corrected codeword, perform logic EXOR between the above register and received codeword.

DECODING:



Decoding:

	m_{26}	m_{25}	m_{24}	m_{23}	m_{22}	m_{21}	m_{20}	m_{19}	m_{18}	m_{17}	m_{16}	m_{15}	m_{14}	m_{13}	m_{12}	P_5	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	P_4	m_4	m_3	m_2	P_3	m_1	P_2	P_1
Hamming code:	1	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	1	1	1	1	0	1	0	0	0	1	1	1	1	1	0

	m_{26}	m_{25}	m_{24}	m_{23}	m_{22}	m_{21}	m_{20}	m_{19}	m_{18}	m_{17}	m_{16}	m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	m_4	m_3	m_2	m_1
Step1:	1	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
step2:	D	0	D	0	0	0	D	0	0	0	D	0	0	0	D	1	1	1	1	0	1	0	0	0	0	0
step3:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
step4:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
step5:	1	0	1	0	0	0	1	0	0	0	1	0	1	0	1	1	1	1	1	0	1	0	0	1	1	1
OR all above values.	2		8				8				A				F				A				7			

Message: 31-bit corrected codeword 0x51157A3E is taken as test input.

Step1: Mask the corrected codeword m26-m12 with #0x7FFF0000 and right shift by #5 and store it in a register R1.

Step2: Mask the corrected codeword m11-m5 with #0x00007F00 and right shift by #4 and store it in a register R2.

Step3: Mask the corrected codeword m4-m2 with #0x00000070 and right shift by #3 and store it in a register R3.

Step4 Mask the corrected codeword m1 with #0x00000004 and right shift by #2 and store it in a register R4 ..

Step5: Perform logical OR on the registers R1,R2,R3,R4 and store the result in R1.

R1 contain the decoded message.