MAJOR PROJECT – EC498

IMPLEMENTATION OF MAJORITY LOGIC DECODING OF BLOCK CODES

Team Members

Akash A (191EC102)
Navratan (191EC133)

Mentor

Professor John D’Souza

National Institute of Technology, Karnataka Surathkal
NH66, Srinivasnagar Surathkal
Mangalore Karnataka – 575025

Current Progress:

- Learnt about the importance of linear and cyclic block codes. Understood how every linear block code can be represented by its generator matrix G or its parity check matrix H. Read about cyclic codes and how any cyclic code can be represented by its generating polynomial g(X).

- Learnt that the encoding and syndrome calculation for cyclic codes can be done conveniently as compared to other linear block codes by polynomial division circuits. Studied about the functioning of a polynomial division circuit created using Linear Feedback Shift Registers (LFSRs).

- Implemented encoders using the LFSR configuration for three different (n, k) cyclic codes. Tested the results for each of the possible information vectors using C and matched the outputs using random tests carried out on Verilog.

- Learnt about the underlying mathematics behind Majority Logic Decoding where parity check sums orthogonal on an error digit are formed and the last error digit can be found using majority logic.

- Read about the Type II configuration of the MLD and understood the mathematics behind the type I decoder where the parity check sums are formed from the syndrome digits instead of the received vector. Understood the advantages offered by each.

- Implemented both Type I and Type II MLDs in both C and Verilog. Tested both of these decoder configurations for all possible combinations of information and error vectors using C. Matched the test results for random tests in Verilog.

- Understood the importance of using Multi Step MLDs single there exist very few codes that are completely orthogonalizable in a single steps. Learnt about how two or more levels of majority logic could be used to perform error correction by forming parity check sums orthogonal to two or more digits instead of a single digit.

- Implemented a two step Majority Logic Decoder on a (15, 5) code available in the text book in both C and Verilog. Tested for all possible combinations of information and error vectors in C and matched the results for random test cases in Verilog.

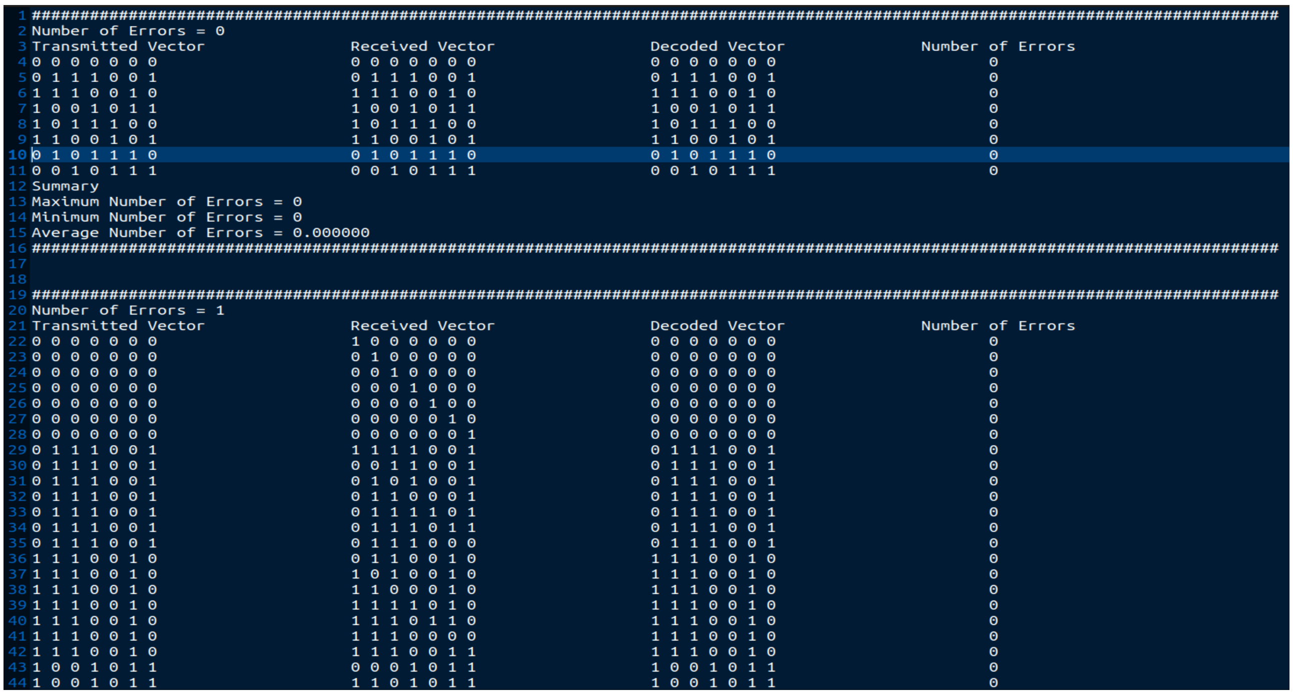
- Currently working on understanding the classes of codes that are majority logic decodable. Attempting to understand the underlying mathematics behind these codes and showing that they are single step or multi step majority logic decodable.

C Implementation:

The C code was implemented to perform both encoding and decoding of a cyclic code that is majority logic decodable. The functions were written in a way to emulate the process occuring in the hardware so that the results could be correctly matched. Three different codes were tried: a (7, 4) single error correcting code, a (15, 7) double error correcting code and a (15, 5) triple error correcting code. The (15, 5) code is not single step but two step majority logic decodable with six parity check sums. The result of the C code is a file showing the transmitted code vector, the received vector after channel-induced errors and the final decoded vector after MLD process. This is done for each of the possible combinations of information and error vectors. The results of the C code are as shown below

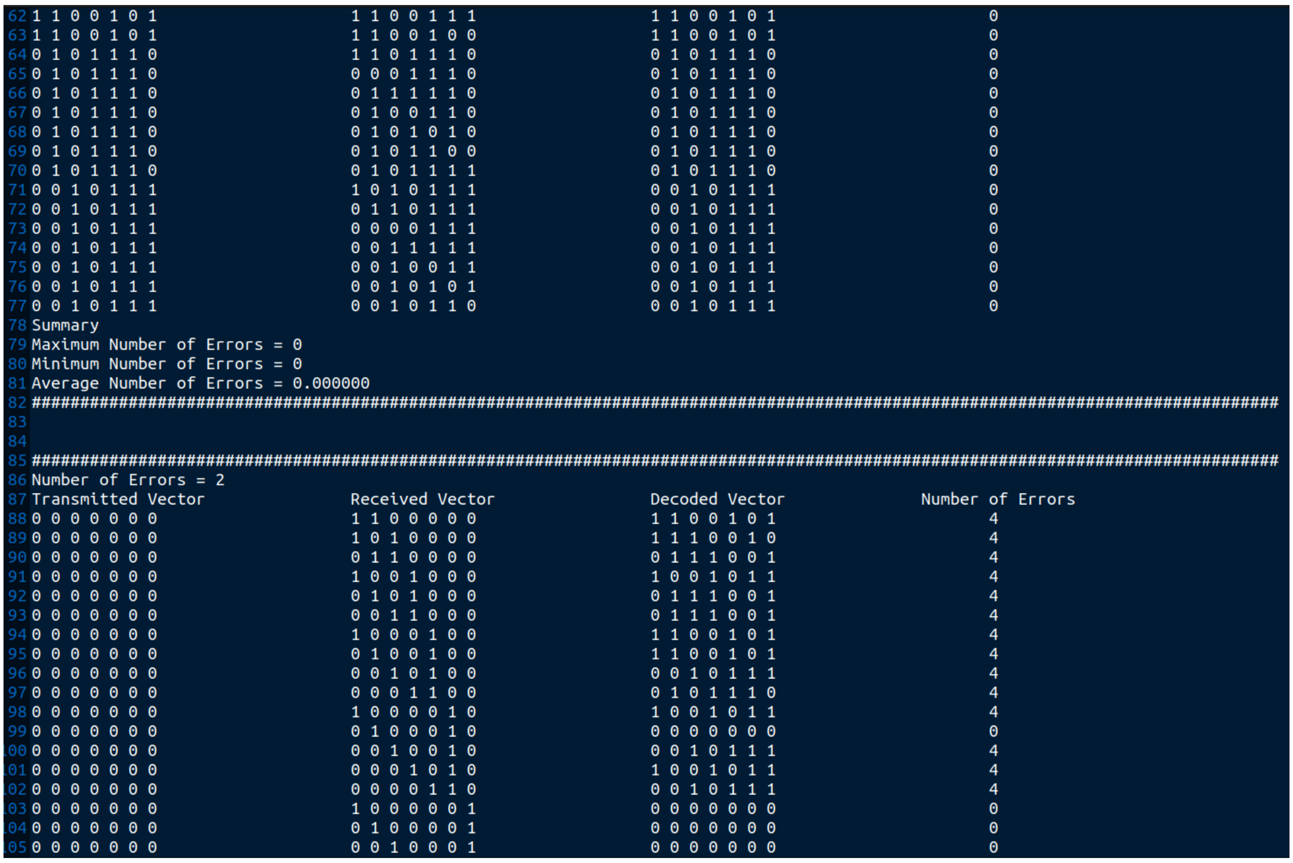
1. (7, 3) Cyclic Code => $g(X) = 1 + X^2 + X^3 + X^4$

Snapshots of C Implementation Results:



```
##################################################################################################################################
Number of Errors = 0
Transmitted Vector               Received Vector                Decoded Vector              Number of Errors
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 1 0 0 1                    0 1 1 1 0 0 1                  0 1 1 1 0 0 1                      0
1 1 1 0 0 1 0                    1 1 1 0 0 1 0                  1 1 1 0 0 1 0                      0
1 0 0 1 0 1 1                    1 0 0 1 0 1 1                  1 0 0 1 0 1 1                      0
1 0 1 1 1 0 0                    1 0 1 1 1 0 0                  1 0 1 1 1 0 0                      0
1 1 0 0 1 0 1                    1 1 0 0 1 0 1                  1 1 0 0 1 0 1                      0
0 1 0 1 1 1 0                    0 1 0 1 1 1 0                  0 1 0 1 1 1 0                      0
0 0 1 0 1 1 1                    0 0 1 0 1 1 1                  0 0 1 0 1 1 1                      0
Summary
Maximum Number of Errors = 0
Minimum Number of Errors = 0
Average Number of Errors = 0.000000
##################################################################################################################################


##################################################################################################################################
Number of Errors = 1
Transmitted Vector               Received Vector                Decoded Vector              Number of Errors
0 0 0 0 0 0 0                    1 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 0 0 0                  0 0 0 0 0 0 0                      0
0 0 0 0 0 0 0                    0 0 1 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 0                  0 0 0 0 0 0 0                      0
0 0 0 0 0 0 0                    0 0 0 0 1 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 0 0 0 0 0 0                      0
0 0 0 0 0 0 0                    0 0 0 0 0 0 1                  0 0 0 0 0 0 0                      0
0 1 1 1 0 0 1                    1 1 1 1 0 0 1                  0 1 1 1 0 0 1                      0
0 1 1 1 0 0 1                    0 0 1 1 0 0 1                  0 1 1 1 0 0 1                      0
0 1 1 1 0 0 1                    0 1 0 1 0 0 1                  0 1 1 1 0 0 1                      0
0 1 1 1 0 0 1                    0 1 1 0 0 0 1                  0 1 1 1 0 0 1                      0
0 1 1 1 0 0 1                    0 1 1 1 1 0 1                  0 1 1 1 0 0 1                      0
0 1 1 1 0 0 1                    0 1 1 1 0 1 1                  0 1 1 1 0 0 1                      0
0 1 1 1 0 0 1                    0 1 1 1 0 0 0                  0 1 1 1 0 0 1                      0
1 1 1 0 0 1 0                    0 1 1 0 0 1 0                  1 1 1 0 0 1 0                      0
1 1 1 0 0 1 0                    1 0 1 0 0 1 0                  1 1 1 0 0 1 0                      0
1 1 1 0 0 1 0                    1 1 0 0 0 1 0                  1 1 1 0 0 1 0                      0
1 1 1 0 0 1 0                    1 1 1 1 0 1 0                  1 1 1 0 0 1 0                      0
1 1 1 0 0 1 0                    1 1 1 0 1 1 0                  1 1 1 0 0 1 0                      0
1 1 1 0 0 1 0                    1 1 1 0 0 0 0                  1 1 1 0 0 1 0                      0
1 1 1 0 0 1 0                    1 1 1 0 0 1 1                  1 1 1 0 0 1 0                      0
1 0 0 1 0 1 1                    0 0 0 1 0 1 1                  1 0 0 1 0 1 1                      0
1 0 0 1 0 1 1                    1 1 0 1 0 1 1                  1 0 0 1 0 1 1                      0
```



```
1 1 0 0 1 0 1                    1 1 0 0 1 1 1                  1 1 0 0 1 0 1                      0
1 1 0 0 1 0 1                    1 1 0 0 1 0 0                  1 1 0 0 1 0 1                      0
0 1 0 1 1 1 0                    1 1 0 1 1 1 0                  0 1 0 1 1 1 0                      0
0 1 0 1 1 1 0                    0 0 0 1 1 1 0                  0 1 0 1 1 1 0                      0
0 1 0 1 1 1 0                    0 1 1 1 1 1 0                  0 1 0 1 1 1 0                      0
0 1 0 1 1 1 0                    0 1 0 0 1 1 0                  0 1 0 1 1 1 0                      0
0 1 0 1 1 1 0                    0 1 0 1 0 1 0                  0 1 0 1 1 1 0                      0
0 1 0 1 1 1 0                    0 1 0 1 1 0 0                  0 1 0 1 1 1 0                      0
0 1 0 1 1 1 0                    0 1 0 1 1 1 1                  0 1 0 1 1 1 0                      0
0 0 1 0 1 1 1                    1 0 1 0 1 1 1                  0 0 1 0 1 1 1                      0
0 0 1 0 1 1 1                    0 1 1 0 1 1 1                  0 0 1 0 1 1 1                      0
0 0 1 0 1 1 1                    0 0 0 0 1 1 1                  0 0 1 0 1 1 1                      0
0 0 1 0 1 1 1                    0 0 1 1 1 1 1                  0 0 1 0 1 1 1                      0
0 0 1 0 1 1 1                    0 0 1 0 0 1 1                  0 0 1 0 1 1 1                      0
0 0 1 0 1 1 1                    0 0 1 0 1 0 1                  0 0 1 0 1 1 1                      0
0 0 1 0 1 1 1                    0 0 1 0 1 1 0                  0 0 1 0 1 1 1                      0
Summary
Maximum Number of Errors = 0
Minimum Number of Errors = 0
Average Number of Errors = 0.000000
##################################################################################################################################


##################################################################################################################################
Number of Errors = 2
Transmitted Vector               Received Vector                Decoded Vector              Number of Errors
0 0 0 0 0 0 0                    1 1 0 0 0 0 0                  1 1 0 0 1 0 1                      4
0 0 0 0 0 0 0                    1 0 1 0 0 0 0                  1 1 1 0 0 1 0                      4
0 0 0 0 0 0 0                    0 1 1 0 0 0 0                  0 1 1 1 0 0 1                      4
0 0 0 0 0 0 0                    1 0 0 1 0 0 0                  1 0 0 1 0 1 1                      4
0 0 0 0 0 0 0                    0 1 0 1 0 0 0                  0 1 1 1 0 0 1                      4
0 0 0 0 0 0 0                    0 0 1 1 0 0 0                  0 1 1 1 0 0 1                      4
0 0 0 0 0 0 0                    1 0 0 0 1 0 0                  1 1 0 0 1 0 1                      4
0 0 0 0 0 0 0                    0 1 0 0 1 0 0                  1 1 0 0 1 0 1                      4
0 0 0 0 0 0 0                    0 0 1 0 1 0 0                  0 0 1 0 1 1 1                      4
0 0 0 0 0 0 0                    0 0 0 1 1 0 0                  0 1 0 1 1 1 0                      4
0 0 0 0 0 0 0                    1 0 0 0 0 1 0                  1 0 0 1 0 1 1                      4
0 0 0 0 0 0 0                    0 1 0 0 0 1 0                  0 0 0 0 0 0 0                      0
0 0 0 0 0 0 0                    0 0 1 0 0 1 0                  0 0 1 0 1 1 1                      4
0 0 0 0 0 0 0                    0 0 0 1 0 1 0                  1 0 0 1 0 1 1                      4
0 0 0 0 0 0 0                    0 0 0 0 1 1 0                  0 0 1 0 1 1 1                      4
0 0 0 0 0 0 0                    1 0 0 0 0 0 1                  0 0 0 0 0 0 0                      0
0 0 0 0 0 0 0                    0 1 0 0 0 0 1                  0 0 0 0 0 0 0                      0
0 0 0 0 0 0 0                    0 0 1 0 0 0 1                  0 0 0 0 0 0 0                      0
```

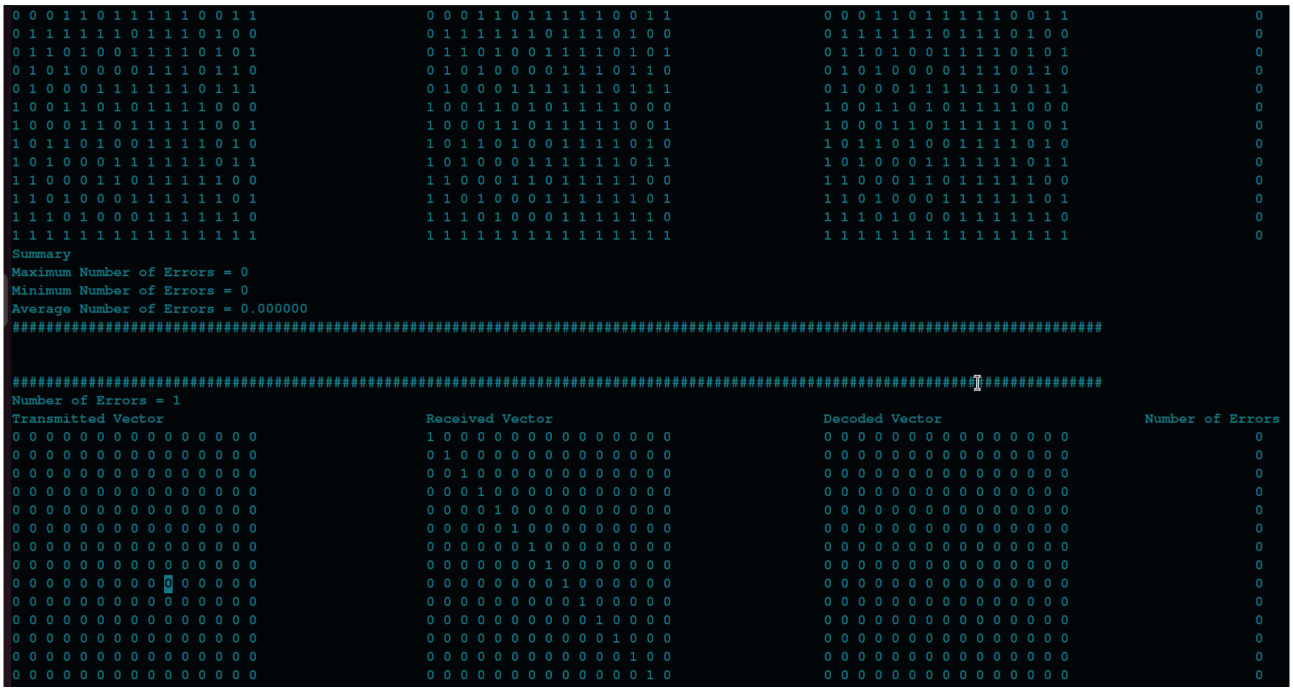
Summary of Results:

# Errors in Received Vector	Maximum # of Errors in Decoded Vector	Minimum # of Errors in Decoded Vector	Average # of Errors in Decoded Vector
0	0	0	0
1	0	0	0
2	4	0	2.667
3	4	0	3.886
4	4	4	4
5	4	4	4
6	4	4	4
7	4	4	4

Inference: It was observed that the minimum distance of the code was 4. Thus, if 4 or more errors occur, the received vector has closer Hamming distance to another code word. Further, 0 or 1 errors are properly corrected.

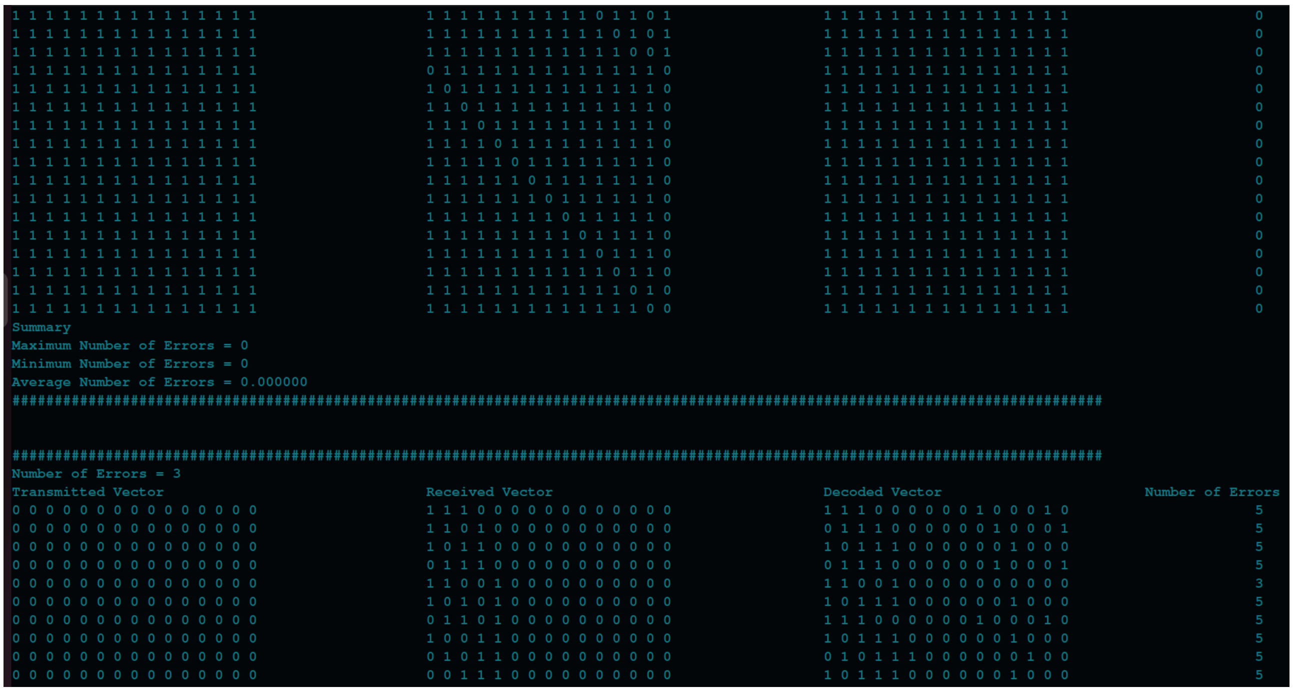
2. (15, 7) Cyclic Code => $g(X) = 1 + X^4 + X^6 + X^7 + X^8$

Snapshots of Results:



```
0 0 0 1 1 0 1 1 1 1 1 0 0 1 1                  0 0 0 1 1 0 1 1 1 1 1 0 0 1 1                 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1                    0
0 1 1 1 1 1 1 0 1 1 1 0 1 0 0                  0 1 1 1 1 1 1 0 1 1 1 0 1 0 0                 0 1 1 1 1 1 1 0 1 1 1 0 1 0 0                    0
0 1 1 0 1 0 0 1 1 1 1 0 1 0 1                  0 1 1 0 1 0 0 1 1 1 1 0 1 0 1                 0 1 1 0 1 0 0 1 1 1 1 0 1 0 1                    0
0 1 0 1 0 0 0 0 1 1 1 0 1 1 0                  0 1 0 1 0 0 0 0 1 1 1 0 1 1 0                 0 1 0 1 0 0 0 0 1 1 1 0 1 1 0                    0
0 1 0 0 0 1 1 1 1 1 1 0 1 1 1                  0 1 0 0 0 1 1 1 1 1 1 0 1 1 1                 0 1 0 0 0 1 1 1 1 1 1 0 1 1 1                    0
1 0 0 1 1 0 1 0 1 1 1 1 0 0 0                  1 0 0 1 1 0 1 0 1 1 1 1 0 0 0                 1 0 0 1 1 0 1 0 1 1 1 1 0 0 0                    0
1 0 0 0 1 1 0 1 1 1 1 1 0 0 1                  1 0 0 0 1 1 0 1 1 1 1 1 0 0 1                 1 0 0 0 1 1 0 1 1 1 1 1 0 0 1                    0
1 0 1 1 0 1 0 0 1 1 1 1 0 1 0                  1 0 1 1 0 1 0 0 1 1 1 1 0 1 0                 1 0 1 1 0 1 0 0 1 1 1 1 0 1 0                    0
1 0 1 0 0 0 1 1 1 1 1 1 0 1 1                  1 0 1 0 0 0 1 1 1 1 1 1 0 1 1                 1 0 1 0 0 0 1 1 1 1 1 1 0 1 1                    0
1 1 0 0 0 1 1 0 1 1 1 1 1 0 0                  1 1 0 0 0 1 1 0 1 1 1 1 1 0 0                 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0                    0
1 1 0 1 0 0 0 1 1 1 1 1 1 0 1                  1 1 0 1 0 0 0 1 1 1 1 1 1 0 1                 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1                    0
1 1 1 0 1 0 0 0 1 1 1 1 1 1 0                  1 1 1 0 1 0 0 0 1 1 1 1 1 1 0                 1 1 1 0 1 0 0 0 1 1 1 1 1 1 0                    0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    0
Summary
Maximum Number of Errors = 0
Minimum Number of Errors = 0
Average Number of Errors = 0.000000
############################################################################################################################


############################################################################################################################
Number of Errors = 1
Transmitted Vector                             Received Vector                               Decoded Vector                      Number of Errors
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                  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 0 0 0 0 0 0 0 0 0 0 0 0 0 0                  0 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 0 0 0 0 0 0 0 0 0 0 0 0 0                  0 0 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 0 0 0 0 0 0 0 0 0 0 0 0                  0 0 0 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 0 0 0 0 0 0 0 0 0 0 0                  0 0 0 0 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 0 0 0 0 0 0 0 0 0 0                  0 0 0 0 0 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 0 0 0 0 0 0 0 0 0                  0 0 0 0 0 0 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 0 0 0 0 0 0 0 0                  0 0 0 0 0 0 0 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 0 0 0 0 0 0 0                  0 0 0 0 0 0 0 0 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 0 0 0 0 0 0                  0 0 0 0 0 0 0 0 0 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 0 0 0 0 0                  0 0 0 0 0 0 0 0 0 0 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 0 0 0 0                  0 0 0 0 0 0 0 0 0 0 0 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 0 0 0                  0 0 0 0 0 0 0 0 0 0 0 0 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 0 0                  0 0 0 0 0 0 0 0 0 0 0 0 0 1 0                 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    0
```



```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 0 1 1 0 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 0 1 0 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 1 0 0 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    0 1 1 1 1 1 1 1 1 1 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 0 1 1 1 1 1 1 1 1 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 0 1 1 1 1 1 1 1 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 0 1 1 1 1 1 1 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 0 1 1 1 1 1 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 0 1 1 1 1 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 0 1 1 1 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 0 1 1 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 0 1 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 0 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 0 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 0 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 1 0 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                      0
Summary
Maximum Number of Errors = 0
Minimum Number of Errors = 0
Average Number of Errors = 0.000000
#################################################################################################################################


#################################################################################################################################
Number of Errors = 3
Transmitted Vector                               Received Vector                                Decoded Vector                        Number of Errors
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 1 1 0 0 0 0 0 0 0 0 0 0 0 0                  1 1 1 0 0 0 0 0 0 1 0 0 0 1 0                      5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 1 0 1 0 0 0 0 0 0 0 0 0 0 0                  0 1 1 1 0 0 0 0 0 0 1 0 0 0 1                      5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 0 1 1 0 0 0 0 0 0 0 0 0 0 0                  1 0 1 1 1 0 0 0 0 0 0 1 0 0 0                      5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    0 1 1 1 0 0 0 0 0 0 0 0 0 0 0                  0 1 1 1 0 0 0 0 0 0 1 0 0 0 1                      5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 1 0 0 1 0 0 0 0 0 0 0 0 0 0                  1 1 0 0 1 0 0 0 0 0 0 0 0 0 0                      3
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 0 1 0 1 0 0 0 0 0 0 0 0 0 0                  1 0 1 1 1 0 0 0 0 0 0 1 0 0 0                      5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    0 1 1 0 1 0 0 0 0 0 0 0 0 0 0                  1 1 1 0 0 0 0 0 0 1 0 0 0 1 0                      5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 0 0 1 1 0 0 0 0 0 0 0 0 0 0                  1 0 1 1 1 0 0 0 0 0 0 1 0 0 0                      5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    0 1 0 1 1 0 0 0 0 0 0 0 0 0 0                  0 1 0 1 1 1 0 0 0 0 0 0 1 0 0                      5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    0 0 1 1 1 0 0 0 0 0 0 0 0 0 0                  1 0 1 1 1 0 0 0 0 0 0 1 0 0 0                      5
```

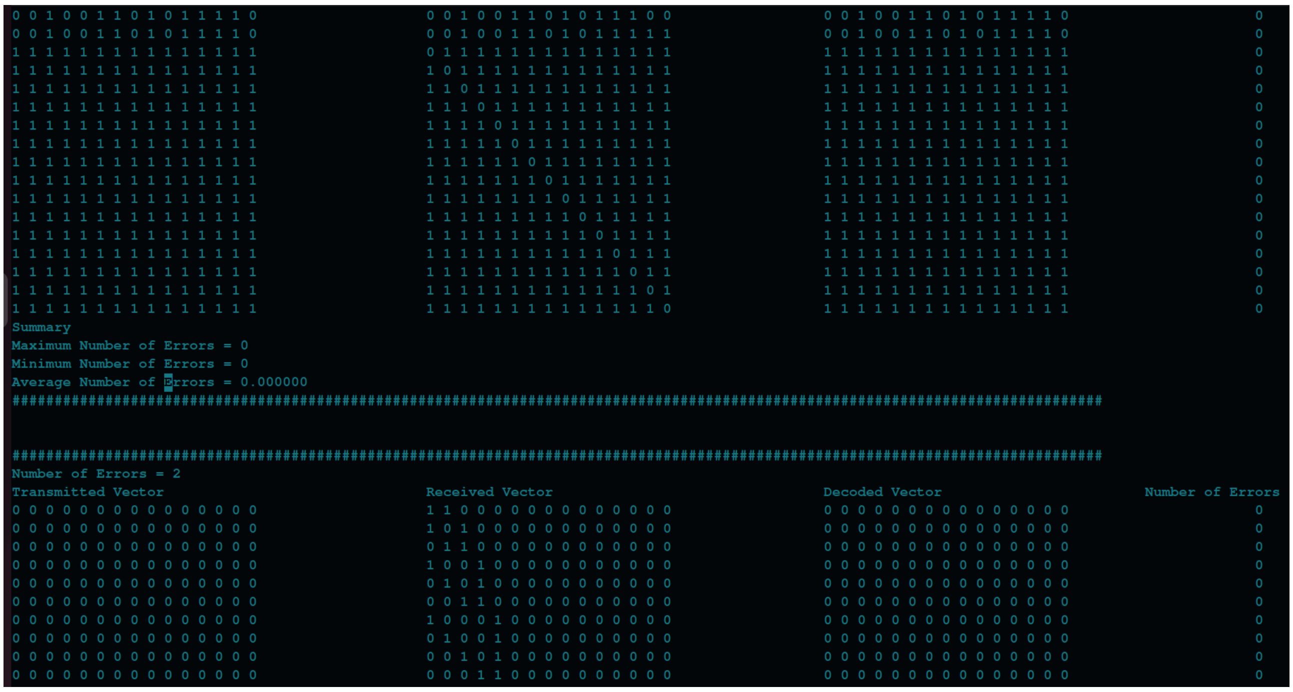
Summary of Results:

# of Errors in Received Vector	Maximum # of Errors in Decoded Vector	Minimum # of Errors in Decoded Vector	Average # of Errors in Decoded Vector
0	0	0	0
1	0	0	0
2	0	0	0
3	7	0	4.224
4	8	0	5.400
5	9	3	5.926
6	10	3	6.562
7	10	4	7.186
8	11	5	7.814
9	12	5	8.438
10	12	6	9.074
11	15	7	9.600
12	15	8	10.776
13	15	15	15
14	15	15	15
15	15	15	15

Inferences: Any occurences of 2 or fewer errors is corrected. There are certain cases when number of errors is large where all the bits are flipped. For n >= 3, the average number of errors after decoding, is greater than that before decoding. This could be a disadvantage as the code could prove dangerous in noisy channels with high number of burst errors.

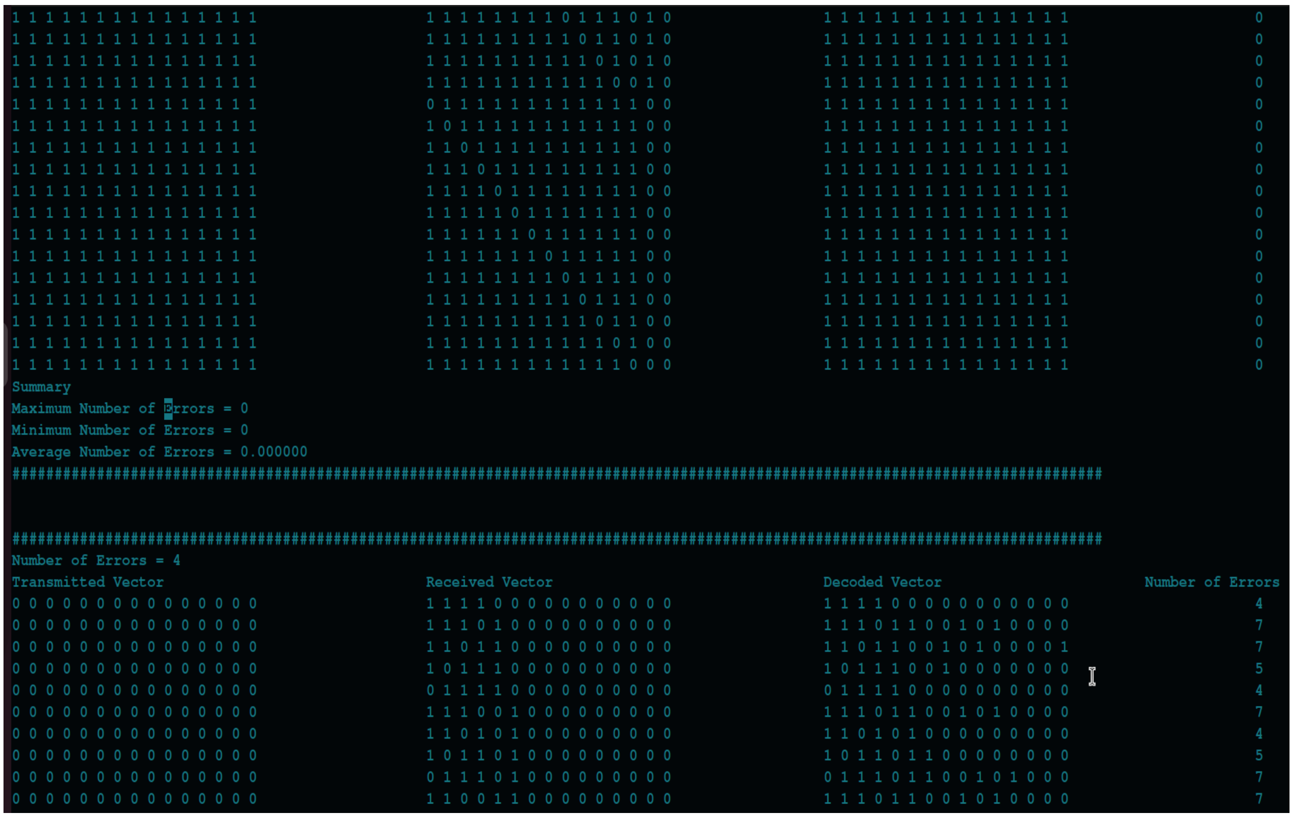
3. (15, 5) Two Step MLD => $g(X) = 1 + X + X^2 + X^4 + X^5 + X^8 + X^{10}$

Snapshot of Results:



```
0 0 1 0 0 1 1 0 1 0 1 1 1 1 0                    0 0 1 0 0 1 1 0 1 0 1 1 1 0 0                  0 0 1 0 0 1 1 0 1 0 1 1 1 1 0                       0
0 0 1 0 0 1 1 0 1 0 1 1 1 1 0                    0 0 1 0 0 1 1 0 1 0 1 1 1 1 1                  0 0 1 0 0 1 1 0 1 0 1 1 1 1 0                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    0 1 1 1 1 1 1 1 1 1 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 0 1 1 1 1 1 1 1 1 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 0 1 1 1 1 1 1 1 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 0 1 1 1 1 1 1 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 0 1 1 1 1 1 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 0 1 1 1 1 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 0 1 1 1 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 0 1 1 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 0 1 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 0 1 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 0 1 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 0 1 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 1 0 1 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 1 1 0 1                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 1 1 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
Summary
Maximum Number of Errors = 0
Minimum Number of Errors = 0
Average Number of Errors = 0.000000
##################################################################################################################################


##################################################################################################################################
Number of Errors = 2
Transmitted Vector                               Received Vector                                Decoded Vector                         Number of Errors
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 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 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 0 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 0 0 0 0 0 0 0 0 0 0 0 0                    0 1 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 0 0 0 0 0 0 0 0 0 0 0 0                    1 0 0 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 0 0 0 0 0 0 0 0 0 0 0                    0 1 0 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 0 0 0 0 0 0 0 0 0 0 0                    0 0 1 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 0 0 0 0 0 0 0 0 0 0 0                    1 0 0 0 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 0 0 0 0 0 0 0 0 0 0                    0 1 0 0 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 0 0 0 0 0 0 0 0 0 0                    0 0 1 0 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 0 0 0 0 0 0 0 0 0 0                    0 0 0 1 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
```



```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 0 1 1 1 0 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 0 1 1 0 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 0 1 0 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 0 0 1 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    0 1 1 1 1 1 1 1 1 1 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 0 1 1 1 1 1 1 1 1 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 0 1 1 1 1 1 1 1 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 0 1 1 1 1 1 1 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 0 1 1 1 1 1 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 0 1 1 1 1 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 0 1 1 1 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 0 1 1 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 0 1 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 0 1 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 0 1 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 0 1 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                    1 1 1 1 1 1 1 1 1 1 1 1 0 0 0                  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1                       0
Summary
Maximum Number of Errors = 0
Minimum Number of Errors = 0
Average Number of Errors = 0.000000
##################################################################################################################################


##################################################################################################################################
Number of Errors = 4
Transmitted Vector                               Received Vector                                Decoded Vector                         Number of Errors
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 1 1 1 0 0 0 0 0 0 0 0 0 0 0                  1 1 1 1 0 0 0 0 0 0 0 0 0 0 0                       4
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 1 1 0 1 0 0 0 0 0 0 0 0 0 0                  1 1 1 0 1 1 0 0 1 0 1 0 0 0 0                       7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 1 0 1 1 0 0 0 0 0 0 0 0 0 0                  1 1 0 1 1 0 0 1 0 1 0 0 0 0 1                       7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 0 1 1 1 0 0 0 0 0 0 0 0 0 0                  1 0 1 1 1 0 0 1 0 0 0 0 0 0 0                       5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    0 1 1 1 1 0 0 0 0 0 0 0 0 0 0                  0 1 1 1 1 0 0 0 0 0 0 0 0 0 0                       4
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 1 1 0 0 1 0 0 0 0 0 0 0 0 0                  1 1 1 0 1 1 0 0 1 0 1 0 0 0 0                       7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 1 0 1 0 1 0 0 0 0 0 0 0 0 0                  1 1 0 1 0 1 0 0 0 0 0 0 0 0 0                       4
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 0 1 1 0 1 0 0 0 0 0 0 0 0 0                  1 0 1 1 0 1 1 0 0 0 0 0 0 0 0                       5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    0 1 1 1 0 1 0 0 0 0 0 0 0 0 0                  0 1 1 1 0 1 1 0 0 1 0 1 0 0 0                       7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0                    1 1 0 0 1 1 0 0 0 0 0 0 0 0 0                  1 1 1 0 1 1 0 0 1 0 1 0 0 0 0                       7
```

Summary of Results:

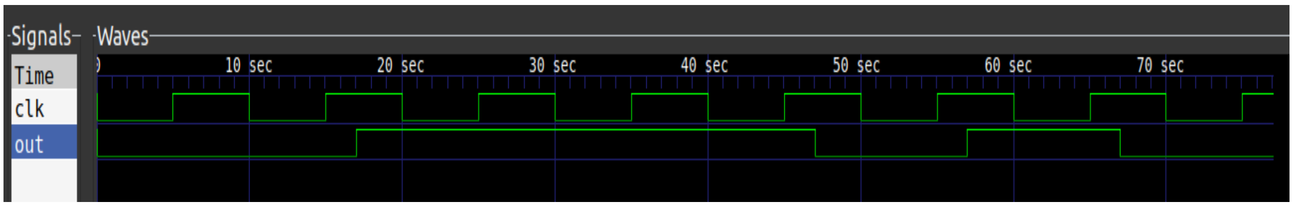
# of Errors in Received Vector	Maximum # of Errors in Decoded Vector	Minimum # of Errors in Decoded Vector	Average # of Errors in Decoded Vector
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	8	4	5.653
5	10	4	6.230
6	11	4	6.756
7	11	4	7.457
8	11	4	7.453
9	11	4	8.244
10	11	5	8.770
11	11	7	9.347
12	15	15	0
13	15	15	0
14	15	15	0
15	15	15	0

Inference: The code can correct 3 or fewer errors.

Verilog Implementation:

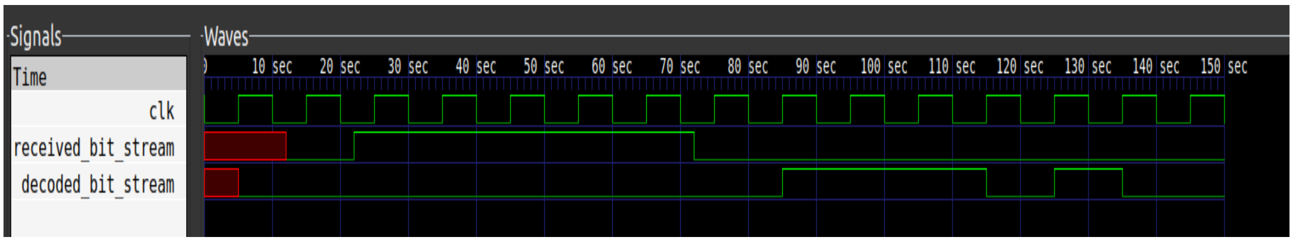
The same three codes were implemented using Verilog where both Type I and Type II decoders were used for the single step majority logic decodable codes and Type II decoder for two step majority logic decodable codes. Currently, the test bench written only tests one code vector at a time. This is because both the encoder and decoder take several clock cycles to process a single code word for which time, the succeeding code word must be maid to wait. We propose to solve this issue by implementing parallellism in the encoder and decoder so that the code words do not have the need to wait. The Verilog files were simulated using Icarus Verilog and GTKWave was used to view the waveforms. The results obtained from the Verilog simulation are as shown below:

1. (7, 4) encoder => Information Vector = (1 1 0); Expected Code Word = (0 1 0 1 1 1 0)



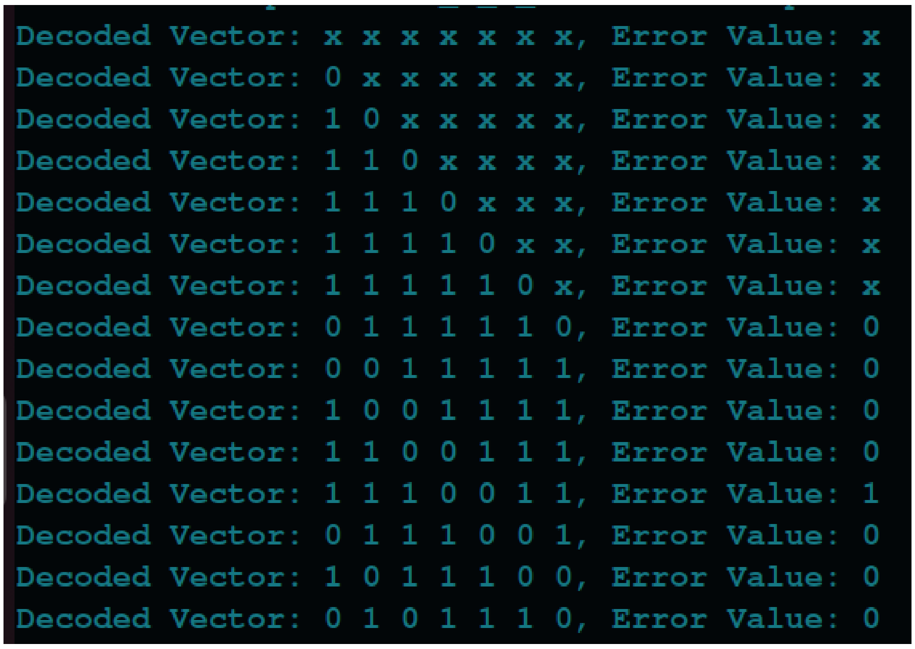
The successive bits at the output of the encoder implemented to appear at an interval of 10 sec starting from 7 sec. Based on this rule, we observe the encoded vector to be (0 1 1 1 0 1 0). We note that this the exact reverse of the expected codeword to be generated. However, this will work correctly since at the receiver, due to shift register configuration, the last bit appears in the first register.

2. (7, 4) Type I Decoder: We now introduce a single error in the encoded vector at an arbitrary position. Let the received vector be (0 1 1 1 1 1 0). After decoding, we must get back the encoded vector. The results of the Type I decoder based on syndrome computation is as shown below:



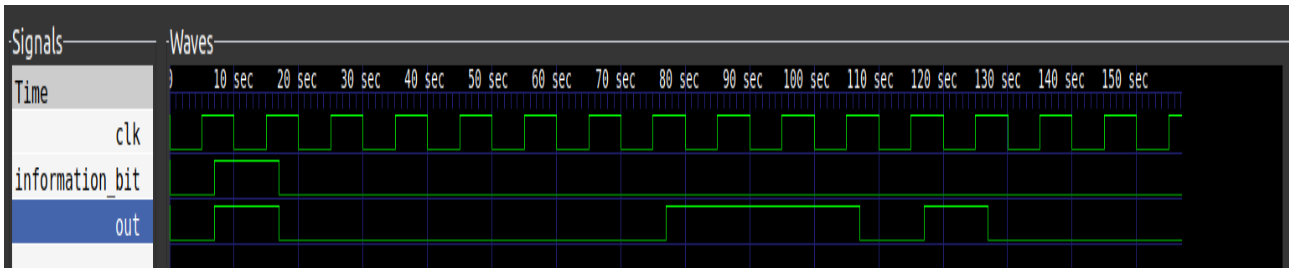
The decoded vector is obtained every 10 sec after 75 sec i.e after the first syndrome has been computed. Observing the above wave form, we can observe the decoded bit stream to be (0 1 1 1 0 1 0). This is the exact reverse of the code word that we had transmitted. Thus, we can infer that the decoding process has occured correctly.

3. (7, 4) Type II Decoder: The same erroneous received vector was given to the Type II decoder to get the exact same results. The results are as shown below:



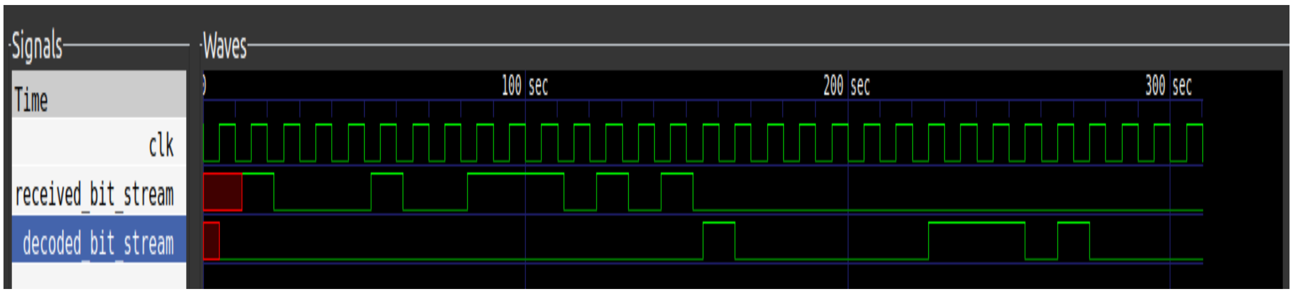
```
Decoded Vector: x x x x x x x, Error Value: x
Decoded Vector: 0 x x x x x x, Error Value: x
Decoded Vector: 1 0 x x x x x, Error Value: x
Decoded Vector: 1 1 0 x x x x, Error Value: x
Decoded Vector: 1 1 1 0 x x x, Error Value: x
Decoded Vector: 1 1 1 1 0 x x, Error Value: x
Decoded Vector: 1 1 1 1 1 0 x, Error Value: x
Decoded Vector: 0 1 1 1 1 1 0, Error Value: 0
Decoded Vector: 0 0 1 1 1 1 1, Error Value: 0
Decoded Vector: 1 0 0 1 1 1 1, Error Value: 0
Decoded Vector: 1 1 0 0 1 1 1, Error Value: 0
Decoded Vector: 1 1 1 0 0 1 1, Error Value: 1
Decoded Vector: 0 1 1 1 0 0 1, Error Value: 0
Decoded Vector: 1 0 1 1 1 0 0, Error Value: 0
Decoded Vector: 0 1 0 1 1 1 0, Error Value: 0
```

4. (15, 7) encoder => Information Vector = (0 0 0 0 0 0 1);
Expected Code Word = (0 0 0 1 0 1 1 1 0 0 0 0 0 0 1)



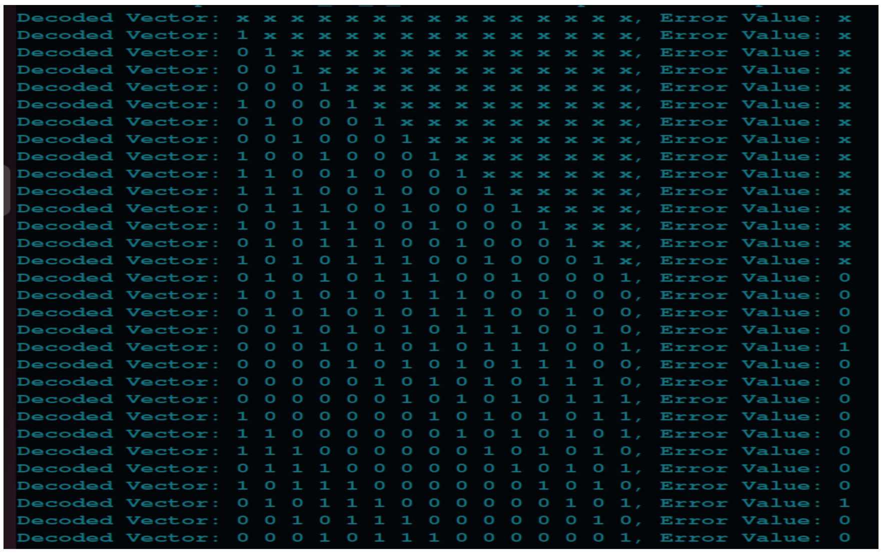
Similar to previous case, successive bits appear at an interval of 10 sec starting at 7 sec. The output of the encoder is then observed to be (1 0 0 0 0 0 0 1 1 1 0 1 0 0 0) which is the reverse of the expected codeword. Thus, the encoder is operating correctly.

5. (15, 7) Type I Decoder: We now introduce two errors in the above code word. Let the received vector be (0 1 0 1 0 1 1 1 0 0 1 0 0 0 1). Shown below is the result after passing through the Type I MLD:



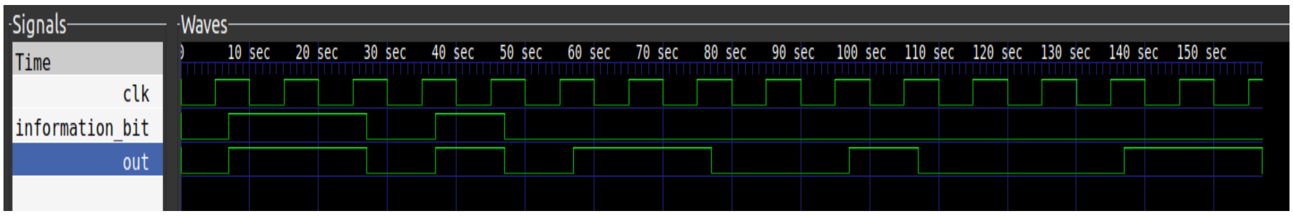
For a 15 bit code, the decoded bit stream starts at 155 seconds at intervals of 10 sec i.e. after the first syndrome is computed. Based on this idea, the decoded bit stream is inferred as (1 0 0 0 0 0 0 1 1 1 0 1 0 0 0). This is the reverse of the required codeword and thus, the decoder is inferred to operate correctly.

6. (15, 7) Type II Decoder: The same test case as the type I decoder was used and the results obtained are as follows:



```
Decoded Vector: x x x x x x x x x x x x x x x, Error Value: x
Decoded Vector: 1 x x x x x x x x x x x x x x, Error Value: x
Decoded Vector: 0 1 x x x x x x x x x x x x x, Error Value: x
Decoded Vector: 0 0 1 x x x x x x x x x x x x, Error Value: x
Decoded Vector: 0 0 0 1 x x x x x x x x x x x, Error Value: x
Decoded Vector: 1 0 0 0 1 x x x x x x x x x x, Error Value: x
Decoded Vector: 0 1 0 0 0 1 x x x x x x x x x, Error Value: x
Decoded Vector: 0 0 1 0 0 0 1 x x x x x x x x, Error Value: x
Decoded Vector: 1 0 0 1 0 0 0 1 x x x x x x x, Error Value: x
Decoded Vector: 1 1 0 0 1 0 0 0 1 x x x x x x, Error Value: x
Decoded Vector: 1 1 1 0 0 1 0 0 0 1 x x x x x, Error Value: x
Decoded Vector: 0 1 1 1 0 0 1 0 0 0 1 x x x x, Error Value: x
Decoded Vector: 1 0 1 1 1 0 0 1 0 0 0 1 x x x, Error Value: x
Decoded Vector: 0 1 0 1 1 1 0 0 1 0 0 0 1 x x, Error Value: x
Decoded Vector: 1 0 1 0 1 1 1 0 0 1 0 0 0 1 x, Error Value: x
Decoded Vector: 0 1 0 1 0 1 1 1 0 0 1 0 0 0 1, Error Value: 0
Decoded Vector: 1 0 1 0 1 0 1 1 1 0 0 1 0 0 0, Error Value: 0
Decoded Vector: 0 1 0 1 0 1 0 1 1 1 0 0 1 0 0, Error Value: 0
Decoded Vector: 0 0 1 0 1 0 1 0 1 1 1 0 0 1 0, Error Value: 0
Decoded Vector: 0 0 0 1 0 1 0 1 0 1 1 1 0 0 1, Error Value: 1
Decoded Vector: 0 0 0 0 1 0 1 0 1 0 1 1 1 0 0, Error Value: 0
Decoded Vector: 0 0 0 0 0 1 0 1 0 1 0 1 1 1 0, Error Value: 0
Decoded Vector: 0 0 0 0 0 0 1 0 1 0 1 0 1 1 1, Error Value: 0
Decoded Vector: 1 0 0 0 0 0 0 1 0 1 0 1 0 1 1, Error Value: 0
Decoded Vector: 1 1 0 0 0 0 0 0 1 0 1 0 1 0 1, Error Value: 0
Decoded Vector: 1 1 1 0 0 0 0 0 0 1 0 1 0 1 0, Error Value: 0
Decoded Vector: 0 1 1 1 0 0 0 0 0 0 1 0 1 0 1, Error Value: 0
Decoded Vector: 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0, Error Value: 0
Decoded Vector: 0 1 0 1 1 1 0 0 0 0 0 0 1 0 1, Error Value: 1
Decoded Vector: 0 0 1 0 1 1 1 0 0 0 0 0 0 1 0, Error Value: 0
Decoded Vector: 0 0 0 1 0 1 1 1 0 0 0 0 0 0 1, Error Value: 0
```

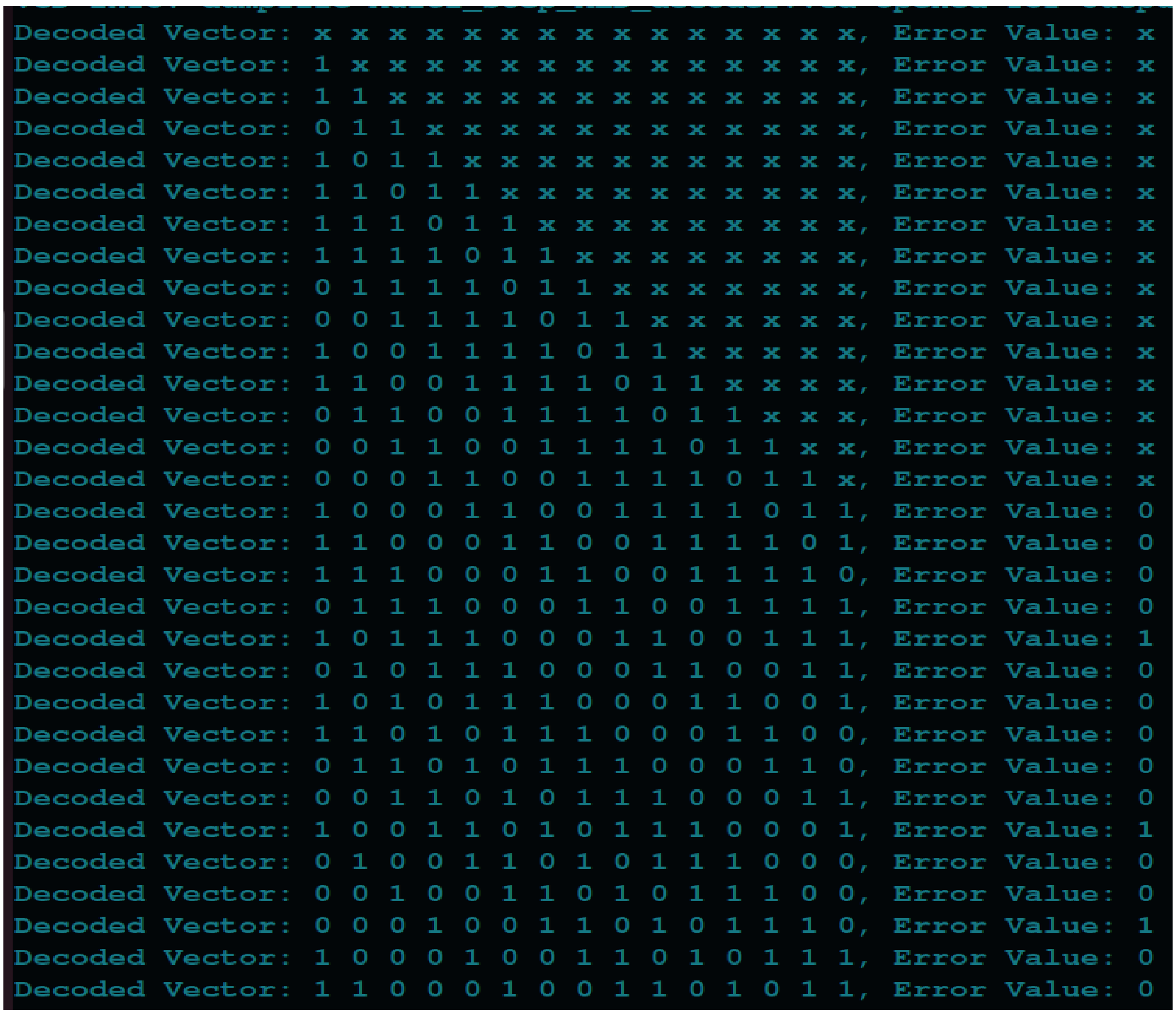
7. Multi Step MLD => Information Vector = (0 1 0 1 1)
Expected Code Word = (1 1 0 0 0 1 0 0 1 1 0 1 0 1 1)



The encoded bit stream as discussed earlier appears at intervals of 10 seconds starting from 7 seconds. The encoded bit stream is inferred to be (1 1 0 1 0 1 1 0 0 1 0 0 0 1 1). This is the reverse of the expected code word and thus, the encoder is operating correctly.

8. Multi Step MLD => Type II Decoder

Let us introduce 3 errors at arbitrary locations in the above code word. Let the received code word be (1 0 0 0 1 1 0 0 1 1 1 1 0 1 1). The output from the decoder is as shown below:



```
Decoded Vector: x x x x x x x x x x x x x x x, Error Value: x
Decoded Vector: 1 x x x x x x x x x x x x x x, Error Value: x
Decoded Vector: 1 1 x x x x x x x x x x x x x, Error Value: x
Decoded Vector: 0 1 1 x x x x x x x x x x x x, Error Value: x
Decoded Vector: 1 0 1 1 x x x x x x x x x x x, Error Value: x
Decoded Vector: 1 1 0 1 1 x x x x x x x x x x, Error Value: x
Decoded Vector: 1 1 1 0 1 1 x x x x x x x x x, Error Value: x
Decoded Vector: 1 1 1 1 0 1 1 x x x x x x x x, Error Value: x
Decoded Vector: 0 1 1 1 1 0 1 1 x x x x x x x, Error Value: x
Decoded Vector: 0 0 1 1 1 1 0 1 1 x x x x x x, Error Value: x
Decoded Vector: 1 0 0 1 1 1 1 0 1 1 x x x x x, Error Value: x
Decoded Vector: 1 1 0 0 1 1 1 1 0 1 1 x x x x, Error Value: x
Decoded Vector: 0 1 1 0 0 1 1 1 1 0 1 1 x x x, Error Value: x
Decoded Vector: 0 0 1 1 0 0 1 1 1 1 0 1 1 x x, Error Value: x
Decoded Vector: 0 0 0 1 1 0 0 1 1 1 1 0 1 1 x, Error Value: x
Decoded Vector: 1 0 0 0 1 1 0 0 1 1 1 1 0 1 1, Error Value: 0
Decoded Vector: 1 1 0 0 0 1 1 0 0 1 1 1 1 0 1, Error Value: 0
Decoded Vector: 1 1 1 0 0 0 1 1 0 0 1 1 1 1 0, Error Value: 0
Decoded Vector: 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1, Error Value: 0
Decoded Vector: 1 0 1 1 1 0 0 0 1 1 0 0 1 1 1, Error Value: 1
Decoded Vector: 0 1 0 1 1 1 0 0 0 1 1 0 0 1 1, Error Value: 0
Decoded Vector: 1 0 1 0 1 1 1 0 0 0 1 1 0 0 1, Error Value: 0
Decoded Vector: 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0, Error Value: 0
Decoded Vector: 0 1 1 0 1 0 1 1 1 0 0 0 1 1 0, Error Value: 0
Decoded Vector: 0 0 1 1 0 1 0 1 1 1 0 0 0 1 1, Error Value: 0
Decoded Vector: 1 0 0 1 1 0 1 0 1 1 1 0 0 0 1, Error Value: 1
Decoded Vector: 0 1 0 0 1 1 0 1 0 1 1 1 0 0 0, Error Value: 0
Decoded Vector: 0 0 1 0 0 1 1 0 1 0 1 1 1 0 0, Error Value: 0
Decoded Vector: 0 0 0 1 0 0 1 1 0 1 0 1 1 1 0, Error Value: 1
Decoded Vector: 1 0 0 0 1 0 0 1 1 0 1 0 1 1 1, Error Value: 0
Decoded Vector: 1 1 0 0 0 1 0 0 1 1 0 1 0 1 1, Error Value: 0
```

Thus, the final decoded vector is exactly the same as the code word. This tells us that the decoder is working correctly.