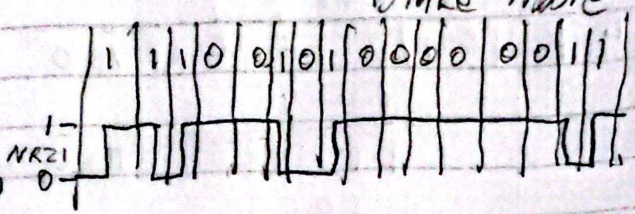
COMP 4320 HW 2

Blake Moore

1\. Sequence: 1110 0101 0000 0011
4B/5B: 11100 01011 11110 10101
NRZI: 10111 10010 10100 11001
(changes at 1 and holds over 0)



2\. 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0
Result: 1 1 0 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 0 1 1 0
Stuffed: 11th 20th 30th

3\. Two-dimensional parity allows detection of 3-bit errors. Using the example to the right we try covering up the error in either the row or column parity bit by flipping a third bit. It will hide one, but not both. So all 3-bit errors are caught.

1	1	1	0	1	1	0	1
1	1	0	1	0	1	0	0
0	1	1	1	1	1	0	1
0	1	1	0	1	0	0	1
1	1	0	0	0	1	0	1
0	0	1	0	1	0	1	1
1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1

If we flip the first two bits of the first two rows, the parity bits and corresponding parity byte remains the same. So, it does not catch all 4-bit errors. The general circumstance is when 4-bit errors balance the parity bits.

0	0	1	0	1	1	0	1
0	0	0	1	0	1	0	0
0	1	1	1	1	1	0	1
0	1	1	0	1	0	0	1
1	1	0	0	0	1	0	1
0	0	1	0	1	0	1	1
1	1	0	0	0	0	0	0
0	0	0	1	0	0	1	0

4\. A.

```
          11111100
1001 / 111 000 11 000
       1001
       0111 00 11 000
        1001
       00111 011 000
         1001
       000 111 11 000
          1001
       000011 01 000
            1001
       0000010 0 000
             100 1
       00000000 100
```

Message: 111 000 11 100

2-D parity tells us the location (ex. (1,2)) of the error, allowing for correction in the 1-bit error scenario.

If we have a 2-bit error, we only detect it, we do not get the location, so we cannot correct it.

4. B. 1001)011000 11 100

1001

11110011 100

1001

10111011 100

1001

10011111 100

1001

10001101 100

1001

10000100 100

1001

10000000000

Remainder = 100

Since the remainder does not equal zero, there is error.