Hamming Codes For Dummies

Parity bits:

* Always least significant bit
* We tend to deal with even parity bits i.e. the parity bit makes sure there is an even number of 1’s
* This allows us to detect 1 error
* It does not allow us to correct any errors

Hamming code:

* Uses multiple parity bits
* Aim is to be able to correct one error
* For an $$n$$ bit message, we use $$\log\_{2}{n} + 1$$ bits
* The first parity bit covers all positions that (in binary) have a 1 in the least significant position (i.e. 1,3,5,7…)
* The second parity bit covers all positions that (in binary) have a 1 in the 2nd least significant position (i.e. 2,3,6,7,10,11…)
* The nth parity bit covers all positions that (in binary) have a 1 in the nth least significant position.

This example will use LITTLE ENDIAN, i.e. the leftmost bit is the least significant bit We are going to take the letter a (which in binary is 10000110) and add the hamming codes to it.

It’s easiest to make sure that there are $$2^{r} -r -1$$ bits, in this case $$r=4$$ (i.e. we have 4 parity bits), so we’ll pad the message with 3 ones so we get 11 bits, making our new message 10000110000.

Our code word will be length 15 ($$2^{4}-1$$).

NB 0101 has 0 as l.s.b, 1 as 2nd l.s.b, 0 as 3rd l.s.b 1 as 4th l.s.b(or m.s.b)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| L.S.B | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
|  | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| M.S.B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Which Parity bits look at it | 1 | 2 | 1,2 | 3 | 1,3 | 2,3 | 1,2,3 | 4 | 1,4 | 2,4 | 1,2,4 | 3,4 | 1,3,4 | 2,3,4 | 1,2,3,4 |

Layout

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Data |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Bit Type | PB | PB | DB | PB | DB | DB | DB | PB | DB | DB | DB | DB | pad | pad | pad |

Fill in the data bits and the padded bits

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Data |  |  | 1 |  | 0 | 0 | 0 |  | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Bit Type | PB | PB | DB | PB | DB | DB | DB | PB | DB | DB | DB | DB | pad | pad | pad |

First parity bit, only looks at positions that have a 1 in the l.s.p of their binary representation (in orange), there are two 1’s, so to have an even number of 1’s we set the parity bit to 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Data | 0 |  | 1 |  | 0 | 0 | 0 |  | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Bit Type | PB | PB | DB | PB | DB | DB | DB | PB | DB | DB | DB | DB | pad | pad | pad |

Second parity bit, only looks at positions that have a 1 in the 2nd l.s.p of their binary representation (in orange), there are three 1’s, so to have an even number of 1’s we set the parity bit to 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Data | 0 | 1 | 1 |  | 0 | 0 | 0 |  | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Bit Type | PB | PB | DB | PB | DB | DB | DB | PB | DB | DB | DB | DB | pad | pad | pad |

Third parity bit, only looks at positions that have a 1 in the 3rd l.s.p of their binary representation (in orange), there are zero 1’s, so to have an even number of 1’s we set the parity bit to 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Data | 0 | 1 | 1 | 0 | 0 | 0 | 0 |  | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Bit Type | PB | PB | DB | PB | DB | DB | DB | PB | DB | DB | DB | DB | pad | pad | pad |

Fourth parity bit, only looks at positions that have a 1 in the 4th l.s.p of their binary representation (in orange), there are two 1’s, so to have an even number of 1’s we set the parity bit to 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Data | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Bit Type | PB | PB | DB | PB | DB | DB | DB | PB | DB | DB | DB | DB | pad | pad | pad |

This gives us the codeword 011000000110000