Binary Codes For The Decimal Digits

BCD(Binary-Coded Decimal) or 8,4,2,1 Code

- In this code, decimal digits 0 through 9 are represented by their natural binary equivalents using four bits and each decimal digit of a decimal number is represented by this four bit code individually.

DECIMAL	BINARY	BINARY CODE DECIMAL 8 4 2 1
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

Excess-3 Code

- This is another form of BCD code, in which each decimal digit is coded into a 4 bit binary code. The code for each decimal digit is obtained by adding decimal 3 to the natural BCD code of the digit.
- For example:

Decimal Numerals	Excess-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

2,4,2,1 Code

- Represent each decimal digit in binary with respect to weight 2 4 2 1. The 2421 code is the same as that in BCD from 0 to 4; however, it varies from 5 to 9. For example, in this case the bit combination 0100 represents decimal 4; whereas the bit combination 1101 is interpreted as the decimal 7, as obtained from

$$2 \times 1 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 7.$$

Decimal	2 4 2 1 Code
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	1 0 1 1
6	1 1 0 0
7	1 1 0 1
8	1 1 1 0
9	1 1 1 1

8,4,-2,-1 Code

- Represent each decimal digit in binary with respect to weight 8, 4, -2, -1.

Decimal	8 4 -2 -1 Code
0	0 0 0 0
1	0 1 1 1
2	0 1 1 0
3	0 1 0 1
4	0 1 0 0
5	1 0 1 1
6	1 0 1 0
7	1 0 0 1
8	1 0 0 0
9	1 1 1 1

Coding Conversion (Example)

Decimal	2 4 2 1 Code	8 4 -2 -1 Code
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 1 1 1
2	0 0 1 0	0 1 1 0
3	0 0 1 1	0 1 0 1
4	0 1 0 0	0 1 0 0
5	1 0 1 1	1 0 1 1
6	1 1 0 0	1 0 1 0
7	1 1 0 1	1 0 0 1
8	1 1 1 0	1 0 0 0
9	1 1 1 1	1 1 1 1

The Reflected Code/ Gray Code

- Unweighted (not an arithmetic code).
- Only a single bit change from one code number to the next.
- Good for error detection.

Decimal	Binary	Gray Code	Decimal	Binary	Gray code
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

Binary-to-Gray Code Conversion

- Retain most significant bit.
- From left to right, add each adjacent pair of binary code bits to get the next Gray code bit, discarding carries.
- Example: Convert binary number 10110 to Gray code.

1	0	1	1	0	Binary
↓					
1					Gray

1	+	0	1	1	0	Binary
		↓				
1		1				Gray

1	0	+	1	1	0	Binary
			↓			
1	1		1			Gray

1	0	1	+	1	0	Binary
				↓		
1	1	1		0		Gray

1	0	1	1	+	0	Binary
					↓	
1	1	1	0		1	Gray

$$(10110)_2 = (11101)_{Gray}$$

Gray-to-Binary Conversion

- Retain most significant bit.
- From left to right, add each binary code bit generated to the Gray code bit in the next position, discarding carries.
- Example: Convert Gray code 11011 to binary.

1	1	0	1	1	Gray
↓					
1					Binary

1		1	0	1	1	Gray
	+	↓				
1		0				Binary

1	1		0	1	1	Gray
		+	↓			
1	0		0			Binary

1	1	0		1	1	Gray
			+	↓		
1	0	0		1		Binary

1	1	0	1		1	Gray
				+	↓	
1	0	0	1		0	Binary

$$(11011)_{Gray} = (10010)_2$$

Error-detection codes

- **Parity bit:** A parity bit is an extra bit included with a message to make the total number of 1's either odd (odd parity) or even (even parity)
- Say, message: 1 0 1 1 0 _____
- If we use odd parity, P(odd) : 0
- If we use even parity, P(even) : 1

Hamming Code

- When data is transmitted from one location to another there is always the possibility that an error may occur. There are a number of reliable codes that can be used to encode data so that the error can be detected and corrected.
- A Hamming Code can be used to detect and correct one-bit change in an encoded code word. This approach can be useful as a change in a single bit is more probable than a change in two bits or more bits.

Example

- Determine the single error correcting code for ' 10 1 1 1 0 1 1 0 '.

Solution:

The number of data bits in the message, n = 9.

We can obtain the required number of parity bits (p) from the following formula:

$$2^p >= n + p + 1$$

$$\text{if } p=1,\ 2^1 >= 9 + 1 + 1 \text{ (false)}$$

$$\text{if } p=2,\ 2^2 >= 9 + 2 + 1 \text{ (false)}$$

$$\text{if } p=3,\ 2^3 >= 9 + 3 + 1 \text{ (false)}$$

$$\text{if } p=4,\ 2^4 >= 9 + 4 + 1 \text{ (true)}$$

So, p= 4. We have to send total m+n = 9+4=13 bit message.

- **Sending End:** Original message: 10 1 1 1 0 1 1 0

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	13
Values	P1	P2	1	P3	0	1	1	P4	1	0	1	1	0

To find the values of P1,P2, P3 and P4 we have to XOR the binary values of the bit positions holding '1' s.

	0	0	1	1	[3]
	0	1	1	0	[6]
	0	1	1	1	[7]
	1	0	0	1	[9]
	1	0	1	1	[11]
	1	1	0	0	[12]
(XOR)	1	1	0	0	
	P4	P3	P2	P1	

So, the sending message code will be: 0 0 1 1 0 1 1 1 1 0 1 1 0

- **Receiving End:**
- Case 1: If there is no error in transmitting, the receiver will get:

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

to check, receiver will XOR the binary values of the bit positions having 1's in the received message again and the XOR result will be all 0s.

	0	0	1	1	[3]
	0	1	0	0	[4]
	0	1	1	0	[6]
	0	1	1	1	[7]
	1	0	0	0	[8]
	1	0	0	1	[9]
	1	0	1	1	[11]
	1	1	0	0	[12]
(XOR)	0	0	0	0	
So, there is no error in the received message.					

- **Receiving End:**
- Case 2: If there is an error in the 6th position while transmitting, the receiver will get:

0 0 1 1 0 <u>0</u> 1 1 1 0 1 1 0

to check, receiver will XOR the binary values of the bit positions having 1's in the received message again and the XOR result will be the binary value of 6..

	0	0	1	1	[3]
	0	1	0	0	[4]
	0	1	1	1	[7]
	1	0	0	0	[8]
	1	0	0	1	[9]
	1	0	1	1	[11]
	1	1	0	0	[12]
(XOR)	**0**	**1**	**1**	**0**	

So, there is an error in the 6th position in the received message and the receiver will flip the value of 6th position to get correct message:
0 0 1 1 0 <u>1</u> 1 1 1 0 1 1 0