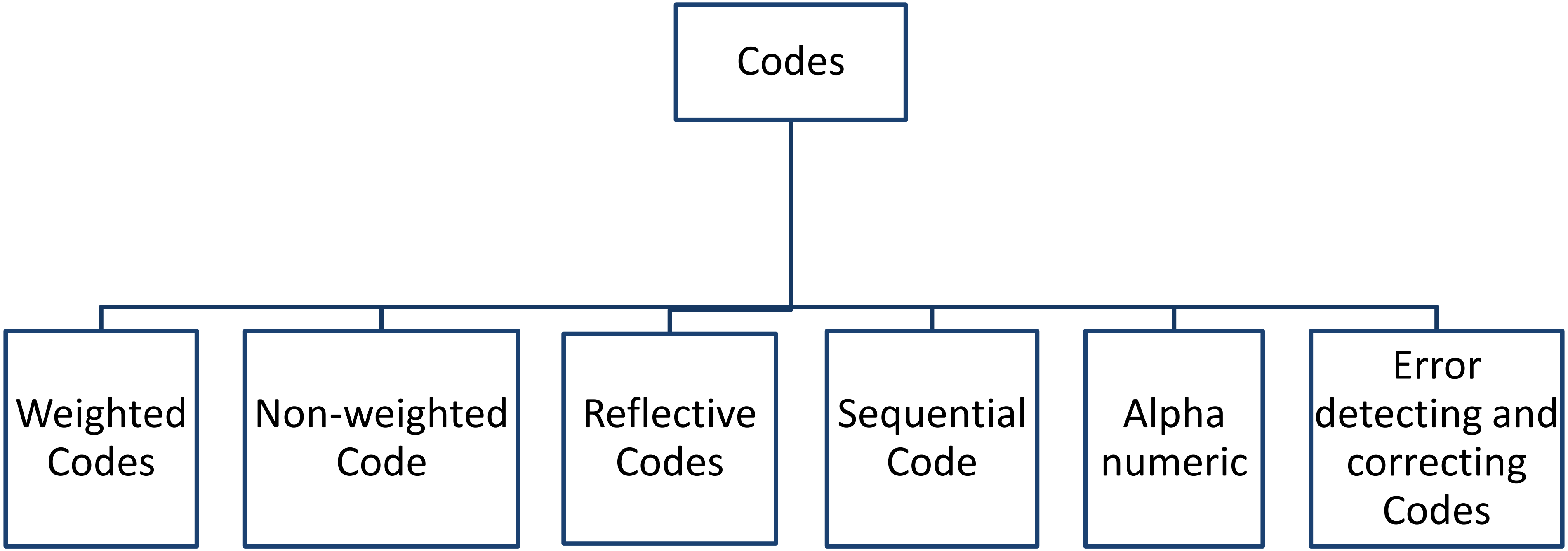
CODES

BCD, XS-3, Gray Code, Alphanumeric Codes (ASCII, EBCDIC), Error detecting and correcting codes (Parity Code, Hamming Code)

Amraja Shivkar

Classification of codes



1. Weighted Codes

- Obey positional weight principle.
- A specific weight is assigned to each position of the number.
- Eg.: Binary, BCD codes

2. Non-weighted Codes

- Do not obey positional weight principle.
- Positional weights are not assigned.
- Eg.: excess-3 code, Gray code

3. Reflective Codes

- A code is said to be reflective when code for 9 is complement of code for 0, code for 8 is complement of code for 1, code for 7 is complement of code for 2, code for 6 is complement of code for 3, code for 5 is complement of code for 4.
- Reflectivity is desirable when 9's complement has to be found.
- Eg.: excess-3 code

4. Sequential Codes

- A code is said to be sequential when each succeeding code is one binary number greater than preceding code.
- Eg.: Binary, XS-3

5. Alphanumeric Codes

- Designed to represent numbers as well as alphabetic characters.
- Capable of representing symbols as well as instructions.
- Eg.: ASCII, EBCDIC

6. Error Detecting and Correcting Codes

- When digital data is transmitted from one system to another, an unwanted electrical disturbance called ‘noise’ may get added to it.
- This can cause an ‘error’ in digital information. That means a 0 can change to 1 or 1 can change to 0.
- To detect and correct such errors special type of codes capable of detecting and correcting the errors are used.
- Eg.: Parity code, Hamming code

BCD(Binary Coded Decimal) Code

- In this code each digit is represented by a 4-bit binary number.
- The positional weights assigned to the binary digits in BCD code are 8-4-2-1 with 1 corresponding to LSB and 8 corresponding to MSB.

Positional Weights	8	4	2	1
	2^3 MSB	2^2	2^1	2^0 LSB

- Other BCD codes like 7-4-2-1, 5-4-2-1 etc also exist.

Conversion from decimal to BCD

- The decimal digits 0 to 9 are converted into BCD, exactly in the same way as binary.

Digital	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Invalid BCD codes:

- With 4 bits we can represent total sixteen numbers (0000 to 1111) but in BCD only first ten codes are used (0000 to 1001)
- Therefore remaining six codes **(1010 to 1111)are invalid in BCD**

Conversion of bigger decimal numbers to BCD:

- Express each decimal digit with its equivalent 4-bit BCD code
- Eg.: Convert $(964)_{10}$ to its equivalent BCD code.

Decimal Number →	9	6	4
Binary Equivalent →	1001	0110	0100

There fore $(964)_{10} = (1001\ 0110\ 0100)_{BCD}$

- Hence smallest number in BCD is 0000 i.e., 0 and largest is 1001 i.e., 9 after which 10 will be expressed by combinations i.e., 0001 0000 and is known as packed BCD

Comparison with Binary:

- Less efficient than binary, since conversion of a decimal number into BCD needs more bits than in binary

Eg., $(22)_{10} = (10110)_2 = (0010\ 0010)_{BCD}$ So BCD uses more bits than binary for the same decimal number.

- BCD arithmetic is more complicated than binary arithmetic.
- BCD – decimal conversion is simpler than Binary – decimal conversion.

Advantages of BCD codes:

- Its similar to decimal number system.
- We need to remember binary equivalents of decimal numbers 0 to 9 only.
- Conversions from decimal to BCD or BCD to decimal is very simple and no calculation is needed.

Disadvantages of BCD codes:

- Less efficient than binary, since conversion of a decimal number into BCD needs more bits than in binary
- BCD arithmetic is more complicated than binary arithmetic.

Convert following decimal numbers to BCD:

(a) 164 (b) 4297 (c) 8065

Convert following BCD codes to decimal equivalent:

(a) 1001 1000 (b) 0001 0100 0110 (c) 0111 0011 0101

Convert following binary numbers to BCD codes: (Hint: convert to decimal first)

(a) 1100 (b) 10001 (c) 1010101

Convert following BCD codes to binary equivalent: (Hint: convert to decimal first)

(a) 0010 1000 (b) 1001 0111 (c) 1000 0000

XS-3 (Excess-3)Code

- Non-weighted code.
- Derived from BCD code (8-4-2-1 code)words by adding (0011)2 or (3)10 to each code word.

$$\text{Decimal} \xrightarrow{\text{Write each digit in 4-bit binary code}} \text{BCD} \xrightarrow{+ (0011)} \text{XS-3}$$

- Therefore Hence smallest number in XS-3 is 0011 i.e., 0 and largest is 1100 i.e., 9

Decimal	BCD	XS-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

- XS-3 is a reflective code since code for 9 is complement of code for 0, code for 8 is complement of code for 1, code for 7 is complement of code for 2, code for 6 is complement of code for 3, code for 5 is complement of code for 4.
- It is a sequential code since each number is 1 binary bit greater than its preceding number.E

Conversion of decimal numbers XS-3 code:

- Eg.: Convert $(964)_{10}$ to its equivalent XS-3 code.

Decimal Number →	9	6	4
XS-3 Equivalent →	1100	1001	0111

Therefore $(964)_{10} = (1100\ 1001\ 0111)_{XS\text{-}3}$

Conversion of XS-3 code to equivalent decimal numbers :

- Eg.: Convert $(0011\ 1010\ 1100)_{XS\text{-}3}$ to its equivalent decimal number.

XS-3 code →	**1010**	**0011**	**1100**
Decimal equivalent →	0	7	9

Therefore $(1010\ 0011\ 1100)_{XS\text{-}3} = (709)_{10}$

Obtain XS-3 equivalent of following numbers:

(a) $(235)_{10}$ (b) $(146)_{10}$ (c) $(0111\ 1000)_{BCD}$ (d) $(1001\ 0011)_{BCD}$

(e) $(101010)_2$ (hint: first convert to decimal)

Gray Code

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

- Non-weighted code.
- It has a very special feature that only one bit will change, each time the decimal number is incremented, therefore also called unit distance code.

Binary and Gray conversions:

- For Gray to binary or binary to Gray conversions let's understand rules for Ex-OR

(Ex-OR is represented by symbol $\oplus$)

Rules for EX-OR:

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 0$$

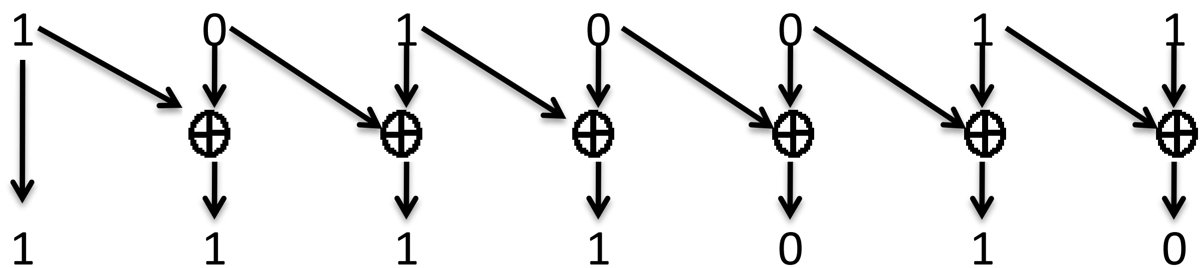
Conversion from Binary to Gray code:

Step 1: Write MSB of given Binary number as it is.

Step 2: Ex-OR this bit with next bit of that binary number and write the result.

Step 3: Ex-OR each successive sum until LSB of that binary number is reached.

- Eg.: Convert $(1010011)_2$ to its equivalent Gray code.



Therefore $(1010011)_2 = (1111010)_{Gray}$

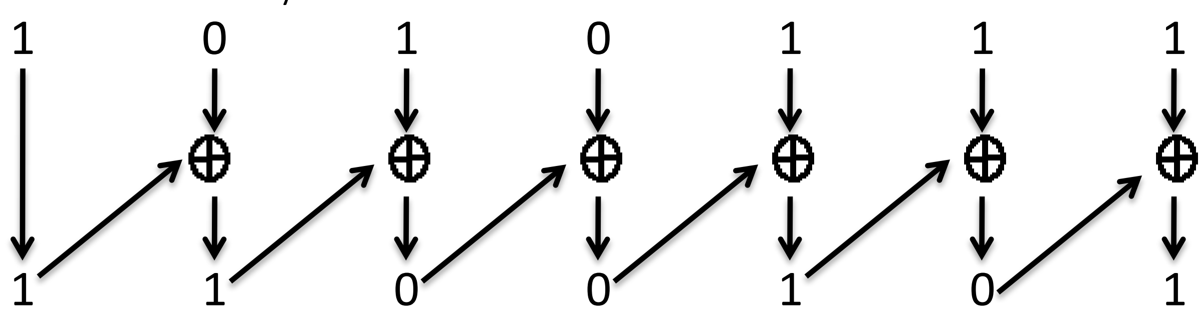
Conversion from Gray to Binary:

Step 1: Write MSB of given Binary number as it is.

Step 2: Ex-OR this bit with next bit of that binary number and write the result.

Step 3: Continue this process until LSB of that binary number is reached.

- Eg.: Convert $(1010111)_{Gray}$ to its equivalent Binary number.



Therefore $(1010111)_{Gray} = (1100101)_2$

Alphanumeric Codes

- A binary bit can represent only two symbols ‘0’ and ‘1’. But it is not enough for communication between two computers because there we need many more symbols for communication.
- These symbols are required to represent

- 26 alphabets with capital and small letters
- Numbers from 0 to 9
- Punctuation marks and other symbols

- Alphanumeric codes represent numbers and alphabetic characters. They also represent other characters such as punctuation symbols and instructions for conveying information.
- Therefore instead of using only single binary bits, a group of bits is used as a code to represent a symbol.

The ASCII code

b_7 →					0	0	0	0	1	1	1	1
b_6 →					0	0	1	1	0	0	1	1
b_5 →					0	1	0	1	0	1	0	1
Bits	b_4 ↓	b_3 ↓	b_2 ↓	b_1 ↓	Column → / Row ↓	0	1	2	3	4	5	6	7
	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
	0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
	0	0	1	0	2	STX	DC2	"	2	B	R	b	r
	0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
	0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
	0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
	0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
	0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
	1	0	0	0	8	BS	CAN	(	8	H	X	h	x
	1	0	0	1	9	HT	EM	)	9	I	Y	i	y
	1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
	1	0	1	1	11	VT	ESC	+	;	K	[	k	{
	1	1	0	0	12	FF	FC	,	<	L	\	l	|
	1	1	0	1	13	CR	GS	-	=	M	]	m	}
	1	1	1	0	14	SO	RS	.	>	N	^	n	~
	1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Encode the following in ASCII code:

1. We the people

W	**1010111**
e	**1100101**
	0100000
t	**1110100**
h	**1101000**
e	**1100101**
	0100000
P	**1010000**
e	**1100101**
o	**1101111**
p	**1100001**
l	**1101100**
e	**1100101**

ASCII- (American Standard Code for Information Interchange)

- Universally accepted alphanumeric code.
- Used in most computers and other electronic equipments. Most computer keyboards are standardized with ASCII.
- When a key is pressed, its corresponding ASCII code is generated which goes to the computer.
- Contains 128 characters and symbols.
- Since $128 = 2^7$ hence we need 7 bits to write 128 characters. Therefore ASCII is a 7 bit code.
- Can be represented in 8 bits by considering MSB = 0 always.
- Hence we have ASCII codes from 0000 0000 to 0111 1111 in binary or from 00 to 7F in hexadecimal.
- The first 32 characters are non-graphic control commands (never displayed or printed) eg., null, escape
- The remaining characters are graphic symbols (can be displayed and printed). This includes alphabets (capital and small), punctuation signs and commonly used symbols.
- So ASCII code consists of 94 printable characters, 32 non printable control commands and "Space" and "Delete" characters = 128 characters

---

Using ASCII table obtain ASCII code word for

(a) DEL (b) % (c) W (d) g (e) &

EBCDIC-(Extended Binary Coded Decimal Interchange Code)

- 8-bit code.
- Total 256 characters are possible, however all are not used.
- There is no parity bit used to check error in this code set.

Using code table obtain EBCDIC code word for

(a) NUL (b) & (c) m (d) SP (e) -

EBCDIC Code Table

B8 →					0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	B7 →				0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
		B6 →			0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
			B5 →		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
B4 ↓	B3 ↓	B2 ↓	B1 ↓	HEX-0 / HEX-1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	NUL	DLE	DS		SP	&	-									0
0	0	0	1	1	SOH	SBA	SOS				/		a	j			A	J		1
0	0	1	0	2	STX	EUA	FS	SYN					b	k	s		B	K	S	2
0	0	1	1	3	ETX	IC							c	l	t		C	L	T	3
0	1	0	0	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0	1	0	1	5	PT	NL	LF	RS					e	n	v		E	N	V	5
0	1	1	0	6	LC		ETB	UC					f	o	w		F	O	W	6
0	1	1	1	7	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1	0	0	0	8		CAN							h	q	y		H	Q	Y	8
1	0	0	1	9		EM							i	r	z		I	R	Z	9
1	0	1	0	A	SMM	CC	SM		¢	!	¦	:								
1	0	1	1	B	VT				.	$	,	#								
1	1	0	0	C	FF	DUP		RA	<	*	%	@								
1	1	0	1	D	CR	SF	ENQ	NAK	(	)	_	'								
1	1	1	0	E	SO	FM	ACK		+	;	>	=								
1	1	1	1	F	SI	ITB	BEL	SUB	|	¬	?	"								

Error detecting and correcting codes

- When a digital information is transmitted, it may not be received correctly by the receiver.
- The error is caused due to electrical disturbance of circuit it is also called noise.
- This noise may force ‘1’ to change to ‘0’ or vice versa.
- This error has to be detected and corrected.

Parity:

- For detection of error an extra bit (parity bit) is attached to code.
- For example: If a 7 bit data (1010110) is to be transmitted then it can be transmitted as 8 bit word (01010110) i.e., even parity code word or as (11010110)i.e., odd parity code word.
- Where parity is decided by extra MSB (parity bit) which is introduced in original data.
- If total number of ‘1’s in transmitted/ received word is even then parity is even.
- If total number of ‘1’s in transmitted/ received word is odd then parity is odd.

BCD code				BCD code with even parity					BCD code with odd parity				
N_4	N_3	N_2	N_1	P	N_4	N_3	N_2	N_1	P	N_4	N_3	N_2	N_1
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	1	1	0	0	0	1	0	0	0	0	1
0	0	1	0	1	0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1	1	0	0	1	1
0	1	0	0	1	0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1	1	0	1	0	1
0	1	1	0	0	0	1	1	0	1	0	1	1	0
0	1	1	1	1	0	1	1	1	0	0	1	1	1
1	0	0	0	1	1	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	1	1	1	0	0	1

Extra

Encode the following in ASCII code with even parity and represent it in hexadecimal code:

1. People

Character	ASCII code with even parity	Hexadecimal code
P	0101 0000	50
e	0110 0101	65
o	0110 1111	6F
p	0111 0001	71
l	0110 1100	6C
e	0110 0101	65

Detection of error by parity code

- Suppose that a 7 bit data (1011010) is to be transmitted with even parity.
- Hence it is transmitted as (01011010) where MSB is parity bit which is kept 0 in order to maintain even parity of transmitted word.
- If it is received as (01011010) i.e., without error then parity still remains even. Hence, it is declared as correct word.
- If it is received as (01111010) i.e., with 1 error then parity becomes odd. Hence, it is declared as incorrect word.
- But the drawback of this code is if data is received with 2 errors , say as (0110010) then parity still remains even and declared as correct word even in spite of being incorrect.
- Also it cannot detect where exactly the error has occurred.

	P	Message bits							Parity	Receivers decision
Transmitted Code	0	1	0	1	1	0	1	0	Even	-
Received Code (with 0 error)	0	1	0	1	1	0	1	0	Even	Correct word
Received Code (with 1 error)	0	1	**1**	1	1	0	1	0	Odd	Correct word
Received Code (with 2 errors)	0	1	**1**	1	0	0	1	0	Even	Correct word

Hamming Code

- It is a linear block code.
- It is an error correcting code
- The 7-bit Hamming code is commonly used, but this concept can be extended to any number of bits.

N_7	N_6	N_5	P_4	N_3	P_2	P_1
7	6	5	4	3	2	1

N → number/data bits   P → Parity bits

- Parity bits are introduced at each 2^n bit where n = 0, 1, 2, 3…
- 1^{st} parity bit is at $\mathbf{2^0 = 1}$ i.e., **1^{st} place** and denoted by $\mathbf{P_1}$
- 2^{nd} parity bit is at $\mathbf{2^1 = 2}$ i.e., **2^{nd} place** and denoted by $\mathbf{P_2}$
- 3^{rd} parity bit is at $\mathbf{2^2 = 2}$ i.e., 4^{th} **place** and denoted by $\mathbf{P_4}$
- 4^{th} parity bit will be at $2^3 = 8$ i.e., 8^{th} place. But since we have only 7 bit code it cannot have this parity bit. So 7 bit Hamming code has only 3 parity bits P_1, P_2, P_4.

A bit word 1 0 1 1 is transmitted. Construct the even parity 7-bit Hamming Code for this data

- **Step 1:** fill the data bits in their respective places (N_7, N_6, N_5, N_3) leaving parity bit places empty as shown

N_7	N_6	N_5	P_4	N_3	P_2	P_1
1	0	1		1		
7	6	5	4	3	2	1

- **Step 2:** Decide P_1:

P_1 checks parity of bit 1, 3, 5, 7 that means it checks on P_1, N_3, N_5, N_7

N_7	N_6	N_5	P_4	N_3	P_2	P_1
1	0	1		1		**1**
7	6	5	4	3	2	1

Set P_1 = 1 to have even parity for Bits 1,3,5,7

- **Step 3:** Decide P_2:

P_2 checks parity of bit 2, 3, 6, 7 that means it checks on P_2, N_3, N_6, N_7

N_7	N_6	N_5	P_4	N_3	P_2	P_1
1	0	1		1	**0**	**1**
7	6	5	4	3	2	1

Set P_2 = 0 to have even parity for Bits 1,3,6,7

- **Step 4:** Decide P_4:

P_4 checks parity of bit 4, 5, 6, 7 that means it checks on P_4, N_5, N_6, N_7

N_7	N_6	N_5	P_4	N_3	P_2	P_1
1	0	1	**0**	1	**0**	**1**
7	6	5	4	3	2	1

Set P_4 = 1 to have even parity for Bits 1,3,5,7

Hence the required 7 bit Hamming code is 1 0 1 0 1 0 1