

Number System

Gray Code

- Un-weighted code
- Important Feature:
 - It exhibits one a single bit change from one code word to the next in sequence

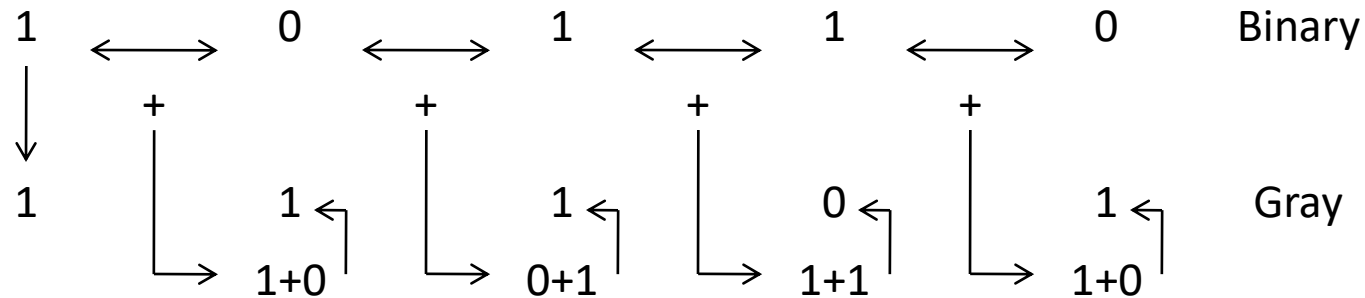
Decimal	Binary	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
12	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

Binary to Gray Code Conversion

- The MSB in the gray code is the same as the corresponding MSB in the binary number
- Going from left to right, add each adjacent pair of binary code bits to get the next gray code bit.
- Discard Carries.

Binary to Gray Code Conversion

- Binary number = 10110



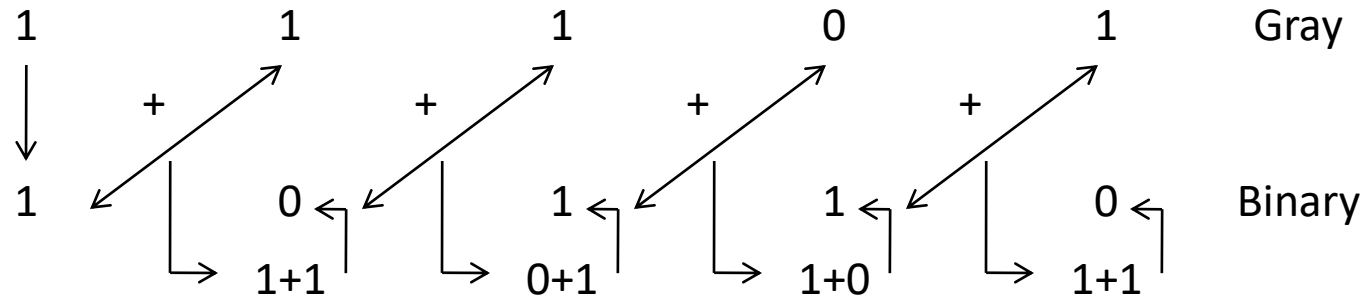
- $(10110)_2 = (11101)_{\text{Gray}}$
- Note: Can perform XOR operation instead of addition. Hence need not think about carry!

Gray to Binary Code Conversion

- MSB in the binary code is the same as the corresponding bit in the gray code.
- Add each binary code bit generated to the next gray code bit in the next adjacent position.
- Discard carries.

Gray to Binary Code Conversion

- Gray code = 11101



- $(11101)_{\text{Gray}} = (10110)_2$
- Note: Can perform XOR operation instead of addition. Hence need not think about carry!

Excess-3 Code

- Un-weighted Code
- Codes are obtained from the corresponding decimal value plus 3 in 4-bit binary
- Self complementing code(9's complement of a number is directly obtained by changing 1s to 0s and 0s to 1s).
 - 9's complement of 4 is 5.
 - 9's complement of 3 is 6

Decimal	BCD	Excess-3 [(BCD+3) in binary]
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0
10	0001 0000	0100 0011

Converting Decimal to Excess-3

- $(23)_{10}$
 - Add both the digits separately by 3
 - $2+3=5$
 - $3+3=6$
 - Convert each corresponding decimal number to equivalent 4 bit binary code
 - 5 – 0101
 - 6 – 0110
 - Answer: $(23)_{10} = (0101\ 0110)_{XS-3}$

Converting Decimal to Excess-3


- $(359.8)_{10}$
 - Add both the digits separately by 3
 - $3+3=6$
 - $5+3=8$
 - $9+3=12$
 - $8+3=11$
 - Convert each corresponding decimal number to equivalent 4 bit binary code
 - $6 - 0110$
 - $8 - 1000$
 - $12 - 1100$
 - $11 - 1011$
 - Answer: $(359.8)_{10} = (0110\ 1000\ 1100.1011)_{XS-3}$

American Standard Code for Information Interchange (ASCII)

- An alphanumeric character set is a set of elements that includes
 - 10 decimal digits
 - 26 letters of alphabets
 - a number of special characters
- Uses 7 bit code and 128 characters
 - Eg: A - 1000001

ASCII

Totally 128
(94+34) characters

- It contains 
 - 94 graphic characters that can be printed
 - 26 uppercase letters(A – Z)
 - 26 lowercase letters(a – z)
 - 10 numerals (0 – 9)
 - 32 special printable characters such as %, *, \$...
 - 34 non-printable characters used for various control functions.

ASCII

- Control Characters
 - Used for routing data and arranging printed text into prescribed format
 - Three types of control characters
 - Format Effectors
 - Characters that affect the layout of printing
 - They include word processor and type writer controls:
 - » Backspace (BS)
 - » Horizontal tabulation (HT)
 - » Carriage return (CR)
 - Information Separators
 - Separate data into divisions such as paragraphs and pages
 - They include:
 - » Record separator(RS)
 - » File separator(FS)

ASCII

- Communication Control Characters
 - Useful during the transmission of text between remote devices so that it can be distinguished from other messages using the same communication channel before it and after it
 - » Start of Text (STX)
 - » End of Text (ETX)
- Mostly computers manipulate an 8-bit quantity as a single quantity. Hence it is stored as one ASCII character per byte
 - Extra bit is used for other purposes depending on the application

ASCII TABLE

Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	\$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL