ECOLE SUPÉRIEURE POLYTECHNIQUE INTERNATIONALE PRIVÉE DE SFAX

Digital Circuits & Computer Architecture

Dr. Eng. Yasmine KOUBAA
PhD in electrical engineering

Academic year 2023-204

Outlines

1. Introduction

2. Number systems

2.1. Decimal Number System (base 10)

2.2. Binary Number System (base 2)

2.3. Octal Number System (base 8)

2.4. Hexadecimal Number System (base 16)

3. Conversion between number systems

3.1. Binary, octal and hexadecimal to decimal conversion

3.2. Decimal to binary conversion

3.3. Binary-octal-hexadecimal and decimal-octal-hexadecimal conversion

4. Binary addition

5. Signed Numbers (1's & 2's - Complement), Negative Number Representation

6. Binary codes

6.1. Gray code

6.2. BCD code (Binary Coded Decimal)

6.3. Excess 3 code

6.4. Error Detecting Codes

1. Introduction

- Machines interpret numbers differently from humans.
- Whenever we type something on a device, those letters convert into certain numbers which only the computer can understand.
- The number system is an essential part of computer technology enabling computers to perform all functions in just a few seconds,

2. Number systems

- There are several number systems which we normally use in modern computing and digital electronics, such as decimal (base 10), binary (base 2), octal (base 8) and hexadecimal (base 16),
- Amongst them we are most familiar with the decimal number system.
- These systems are classified according to the values of the base (B) of the number system.

2.1.Polynomial représentation

- In general, we can express any number in any base or radix "B." Any number with base B, having n digits to the left and m digits to the right of the decimal point, can be expressed as:

$$\enclose{circle}{a_n}B^{n-1} + a_{n-1}B^{n-2} + a_{n-2}B^{n-3} + \cdots + a_2B^1 + a_1B^0 + b_1B^{-1} + b_2B^{-2} + .. + \enclose{circle}{b_m}B^{-m}$$

where a_n is the digit in the nth position. The coefficient $\boldsymbol{a_n}$ is defined as the **MSD (**Most Significant Digit) and $\boldsymbol{b_m}$ is defined as the **LSD** (Least Significant Digit).

2. Number systems

2.1. Decimal Number System (base 10)

- The most common system for number representation is the **decimal.** It’s used in finances, engineering and biology, almost everywhere we see and use numbers.
- With a decimal system we have **10 different digits :**

DECIMAL SYMBOLS									
0	1	2	3	4	5	6	7	8	9

- As you can see there are 10 symbols from 0 to 9. With these symbols we can construct all the numbers in the decimal system, such as $(2023)_{10}$ $(208)_{10}$

2. Number systems

Examples:

$$(5462)_{10} = 5X10^3 + 4X10^2 + 6X10^1 + 2X10^0$$

$$(239.537)_{10} = 2X10^2 + 3X10^1 + 9X10^0 + 5X10^{-1} + 3X10^{-2} + 7X10^{-3}$$

We can keep in mind these characteristics of the decimal numbers system:

- it’s using 10 symbols
- can be decomposed in factors containing powers of 10
- it’s the most common number representation system

2. Number systems

2.2. Binary Number System (base 2)

- A binary system has only 2 different digits: 0 and 1. So to deal with a binary number system is quite easier than a decimal system.

BINARY SYMBOLS	
0	1

- In a digital world, we can think in binary nature, e.g., a light can be either off or on. There is no state in between these two.
- So we generally use the binary system when we deal with the digital world : all the operations are done using two levels of voltage: high and low. Each level of voltage is assigned to a value/symbol: HIGH for 1 and LOW for 0. For a microcontroller which is supplied with +5V the 1 (high) will be represented by +5 V and the 0 (low) by 0 V.

2. Number systems

- All the decimal numbers we can think of can be represented into binary symbols i.e., 0 and 1.

Decimal (base 10)	Binary (base 2)
$(2510)_{10}$	$(11001)_{2}$

- Usually to distinguish between a decimal or binary number, we must specify the base to which we are referring to. The base is described as a subscript after the last character of the number
- By specifying the base of the number we eliminate the probability of confusion, because the same representation (e.g. 11) can mean different things for different bases.

$$(11)_{10} \neq (11)_{2}$$

- To specify a binary number we use the prefix 0b.
  Example: 0b1100

2. Number systems

01110011

binary digit

8 binary digits = 1 Byte

Example:
1000111100001111
16 binary digits = 2 Bytes

The characteristics of a binary system are:

- it’s using 2 symbols
- can be decomposed in factors containing powers of 2
- it’s used in computers, microcontrollers

2. Number systems

2.3. Octal Number System (base 8)

- In an octal number system there are 8 digits:

OCTAL SYMBOLS							
0	1	2	3	4	5	6	7

- $(107)_8$ is an octal number and the subscript after the last character of the number clearly indicate the base 8

- Any octal number cannot have any digit greater than 7.
  Example :
  109 is it an octal number?
  109 isn't an octal number

2. Number systems

2.4. Hexadecimal Number System (base 16)

- Hexadecimal number system has 16 digits 0 to 9 and the rest of the six digits are specified by letter symbols as A,B, C, D, E, and F :

Hexadecimal symbols															
0	1	2	3	4	5	6	7	8	9	A (10)	B (11)	C (12)	D (13)	E (14)	F (15)

- Common practice is to use the prefix “0x” in order to distinguish from the decimal notation. Example: 0x105E9.

- The characteristics of a hexadecimal number representation system are:
  - It’s using 16 symbols
  - Can be decomposed in factors containing powers of 16
  - It’s used in computers, microcontrollers

2. Number systems

The table below summaries the characteristics of the above mentioned number representation systems:

System	Nomber of symbols	Symbols	Prefix	Example
Decimal	10	0,1,2,3,4,5,6,7,8,9	None	147
Binary	2	0,1	0b	11011001
Octal	8	0,1,2,3,4,5,6,7	None	41
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	0X	7DE1

Decimal numbers	4 bits binary number	Hexadecimal number	3 bits binary number	Octal number
0	0000	0	000	0
1	0001	1	001	1
2	0010	2	010	2
3	0011	3	011	3
4	0100	4	100	4
5	0101	5	101	5
6	0110	6	110	6
7	0111	7	111	7
8	1000	8		10
9	1001	9		11
10	1010	A		12
11	1011	B		13
12	1100	C		14
13	1101	D		15
14	1110	E		16
15	1111	F		17

3. Conversion between number systems

- It is often required to convert a number in a particular number system to any other number system, e.g., it may be required to convert a decimal number to binary or octal or hexadecimal.
- The reverse is also true, i.e., a binary number may be converted into decimal and so on.

3.1. Binary, octal and hexadecimal to decimal conversion

- Converting a number N from any number system B to a decimal number is given by polynomial representation.

- Each of the binary, octal, or hexadecimal number system is a positional number system, i.e., each of the digits in the number systems has a positional weight as in the case of the decimal system.

3. Conversion between number systems

Examples:

Binary → Decimal

$(101)_2 = (\qquad)_{10}$

$(11010)_2 = (\qquad)_{10}$

$(1111)_2 = (\qquad)_{10}$

Octal → Decimal

$(12)_8 = (\qquad)_{10}$

$(107)_8 = (\qquad)_{10}$

$(200)_8 = (\qquad)_{10}$

Hexadecimal → Decimal

$(15)_{16} = (\qquad)_{10}$

$(FF)_{16} = (\qquad)_{10}$

$(2C0)_{16} = (\qquad)_{10}$

3. Conversion between number systems

3.2. Decimal to binary conversion

- To convert a **number in decimal to a number in binary** we have to **divide the decimal number** by **2** repeatedly, **until the quotient of zero is obtained**.
- This method of repeated division by 2 is called the ‘double-dabble’ method.
- The remainders are noted down for each of the division steps. Then the column of the remainder is read in reverse order i.e., from bottom to top order.
- We try to show the method with examples:

Decimal → Binary

$(43)_{10} = (\qquad\qquad)_2$

$(9)_{10} = (\qquad\qquad)_2$

$(24)_{10} = (\qquad\qquad)_2$

$(33)_{10} = (\qquad\qquad)_2$

$(256)_{10} = (\qquad\qquad)_2$

3. Conversion between number systems

Conversion from a Binary to Octal Number and Vice Versa

- We know that the maximum digit in an octal number system is 7, which can be represented as $(111)_2$ in a binary system. Hence, **starting from the LSB**, we **group three digits** at a time and replace them by the decimal equivalent of those groups and we get the final octal number.

- Since at the time of grouping of three digits starting from the LSB, we find that a group cannot be completed, so we complete the group by **adding 0s to the MSB side**.

Binary ⟷ Octal

$(100101)_2 = (\qquad)_8$

$(10010110)_2 = (\qquad)_8$

3. Conversion between number systems

Conversion from a Binary to Hexadecimal Number and Vice Versa

- We know that the maximum digit in a hexadecimal system is 15, which can be represented by $(1111)_2$ in a binary system. Hence, **starting from the LSB**, we **group four digits** at a time and replace them with the hexadecimal equivalent of those groups and we get the final hexadecimal number

- Since at the time of grouping of four digits starting from the LSB, we find that a group cannot be completed, so we complete the group by **adding 0s to the MSB side**.

Binary ⟷ Hexadecimal

$(11010011)_2 = (\qquad)_{16}$

$(1111011011)_2 = (\qquad)_{16}$

3. Conversion between number systems

Conversion from an Octal to Hexadecimal Number and Vice Versa

- **To convert an octal number into a hexadecimal number** the following steps are to be followed:
  (*i*) First convert the octal number to its binary equivalent (as already discussed above).
  (*ii*) Then form groups of 4 bits, starting from the LSB.
  (*iii*) Then write the equivalent hexadecimal number for each group of 4 bits.

- For **converting a hexadecimal number into an octal number** the following steps are to be followed:
  (*i*) First convert the hexadecimal number to its binary equivalent.
  (*ii*) Then form groups of 3 bits, starting from the LSB.
  (*iii*) Then write the equivalent octal number for each group of 3 bits.

3. Conversion between number systems

Examples:

Octal ⟷ Hexadecimal

$(25)_8 = (\quad\quad\quad)_{16}$

$(6401)_8 = (\quad\quad\quad)_{16}$

3. Conversion between number systems

3.3. Binary-octal-hexadecimal and decimal-octal-hexadecimal conversion

Decimal to octal conversion

- To convert a number in decimal to a number in octal we have to **divide the decimal number by 8** repeatedly, until the quotient of zero is obtained.
- This method of repeated division by 8 is called ‘octal-dabble.’
- The remainders are noted down for each of the division steps. Then the column of the remainder is read from bottom to top order, just as in the case of the double-dabble method

Decimal to hexadecimal conversion

- The same steps are repeated to convert a number in decimal to a number in hexadecimal. Only here we have to **divide the decimal number by 16 repeatedly**, until the quotient of zero is obtained.
- This method of repeated division by 16 is called ‘hexdabble.’
- The remainders are noted down for each of the division steps. Then the column of the remainder is read from bottom to top

3. Conversion between number systems

Examples:

Decimal ⟷ Octal ⟷ Hexadecimal

$(68)_{10} = (\qquad\qquad)_8 = (\qquad\qquad)_{16}$

$(45)_{10} = (\qquad\qquad)_8 = (\qquad\qquad)_{16}$

4. Binary addition

- The binary addition operation works similarly to the base 10 decimal system, except that it is a base 2 system.
- The binary system consists of only two digits, 1 and 0. Most of the functionalities of the computer system use the binary number system.
- The binary code uses the digits 1's and 0's to make certain processes turn off or on.

Rules of binary addition

0 + 0 = 0
0 + 1 = 1
1 + 0 =1
1 + 1 = 01
1+1+1=11

4. Binary addition

Example:

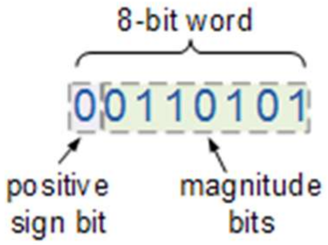
100 + 11 = (111)2
1111 + 1 = (10000)2
1001 + 111 = (10000)2
1111 + 1011 = (11010)2
11011+1111 = (101010)2

5. Signed Numbers (1's & 2's - Complement), Negative Number Representation

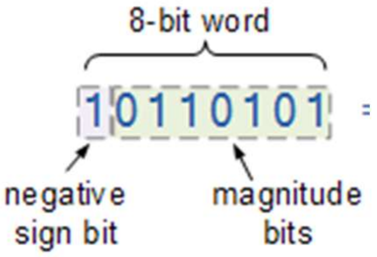
Signed Numbers

- For signed binary numbers the most significant bit (MSB) is used as the sign bit.

- If the sign bit is “0”, this means the number is positive in value.



- If the sign bit is “1”, then the number is negative in value.



5. Signed Numbers (1's & 2's - Complement), Negative Number Representation

1’s complement of a binary number is another binary number obtained by transforming the 0 bit to 1 and the 1 bit to 0.

	0	1	1	1
1’s complement	1	0	0	0

	1	1	0	0
1’s complement	0	0	1	1

2’s complement of a binary number is 1, added to the 1’s complement of the binary number.

Binary number	0	1	1	1
1’s Complement	1	0	0	0
+1	0	0	0	1
2’s complement	1	0	0	1

5. Signed Numbers (1's & 2's - Complement), Negative Number Representation

In the case of signed numbers 101 ≠5

Sign bit

2’s complement is used to encode signed numbers:

- Step 1: Encode the number in binary base (base 2)
- Step 2 : Apply the 1’s Complement to the natural binary number
- Step 3 : Add 1 to the 1’s Complement

Exemple : $(-5)_{10}= (?)_2$

Binary number	**0**	**1**	**0**	**1**
1’s Complement	1	0	1	0
+1	0	0	0	1
2’s complement	1	0	1	1

Sign bit

(-5)= (1011)

6. Binary codes

a → 1100001

→

110000000
001100011
001100011
001100110

→

110000001
001100011
001100011
001100110
000000111

6. Binary codes

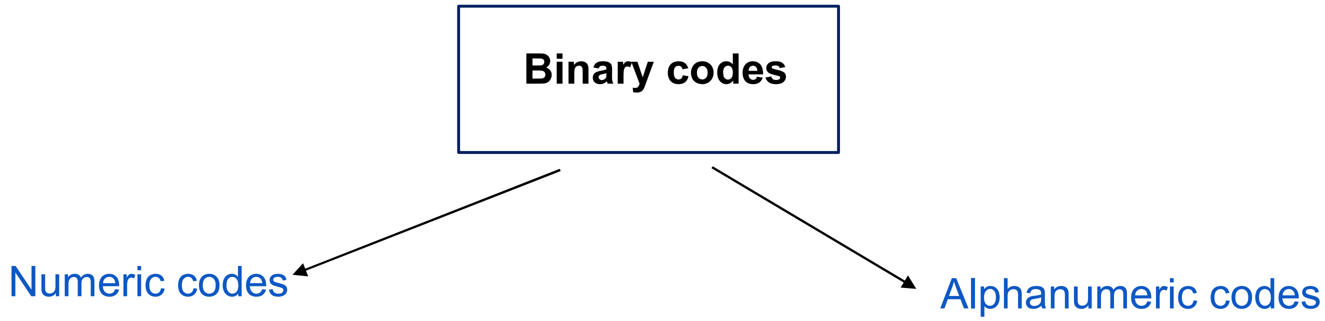
Encoding

110000000
001100011
001100011
001100110

Decoding

- Encoding and decoding are carried out using binary codes

6. Binary codes



Binary codes

Numeric codes

Alphanumeric codes

<u>non-weighted</u>

- Gray code
- Excess-3

<u>weighted</u>

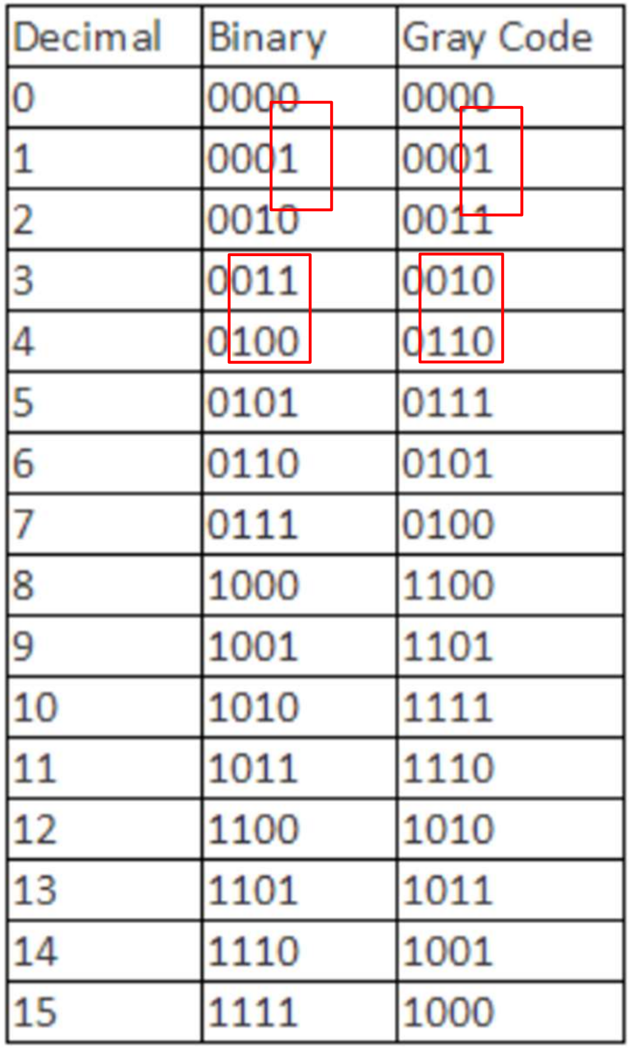
- BCD 8-4-2-1
- 2-4-2-2
- 3-3-2-1
- 8-4-(-2)-(-1)
- 6-3-1-(-1)

- Ascii (7 bits)

6. Binary codes

6.1. Gray code

- It is the non-weighted code. That means there are no specific weights assigned to the bit position.
- It has a very special feature that, only **one bit** will change each time the decimal number is incremented.

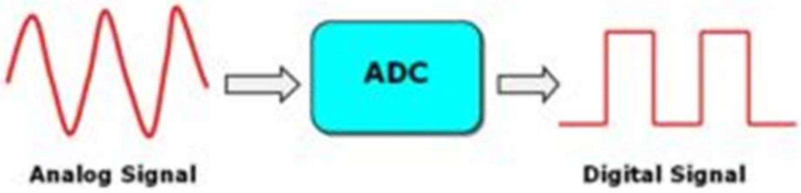


Decimal	Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

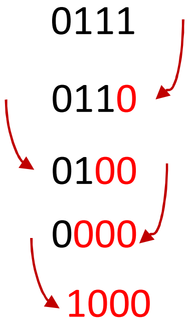
6. Binary codes

Applications of gray code

- Gray code is used in digital signal transmission because it minimizes the occurrence of errors.
- Gray code is preferable to straight binary code in angle measuring devices. Using Gray code almost eliminates the possibility of misreading an angle, which is likely if the angle is represented in binary form (base 2).



Analog To Digital Conversion

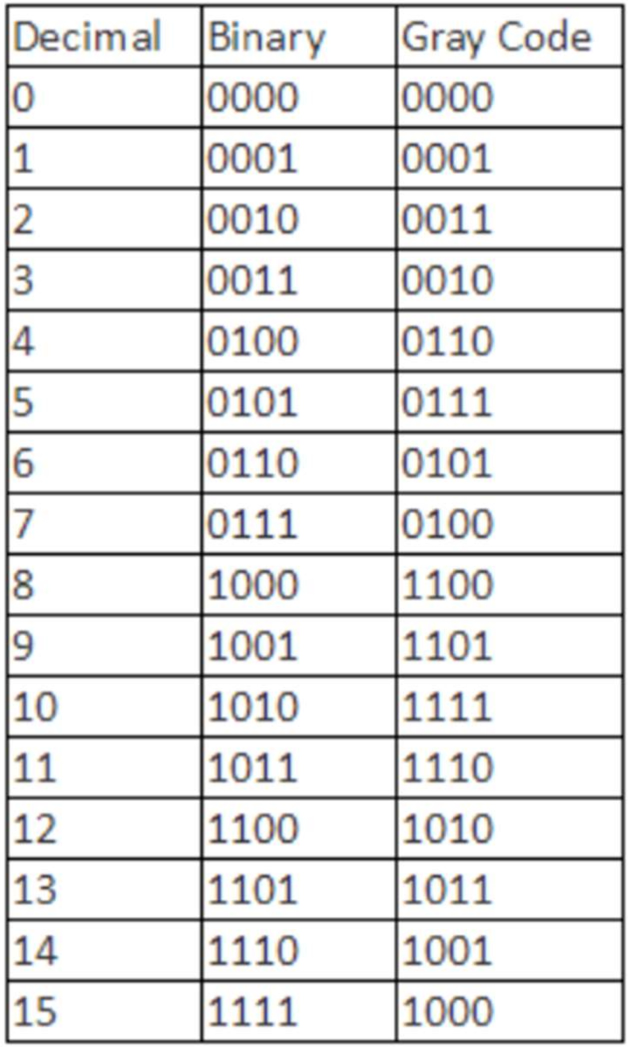

0111

0110

0100

0000

1000



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

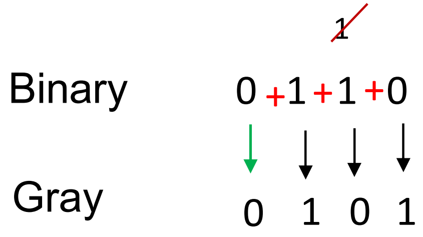
6. Binary codes

Convert Binary number to Gray number

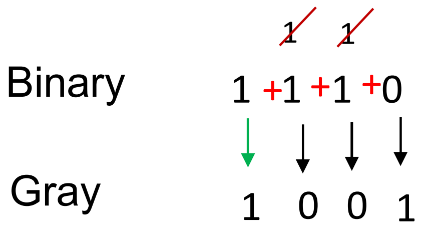
1. Start with the most significant bit (MSB) of the binary number. The MSB of the Gray code equivalent is the same as the MSB of the given binary number.
2. The second most significant bit, adjacent to the MSB, in the Gray code number is obtained by adding the MSB and the second MSB of the binary number and ignoring possible carry. That is, if the most significant bit and the bit adjacent to it are both "1", the corresponding Gray code bit would be a "0".
3. The third most significant bit, adjacent to the second MSB, of the Gray code number is obtained by adding the second MSB and the third MSB to the binary number and ignoring possible carry.
4. The process continues until the LSB of the Gray code number is obtained by adding the LSB and the nearest upper adjacent bit of the binary number.

Example:

$(6)_{10}=(0110)_2$



$(14)_{10}=(1110)_2$



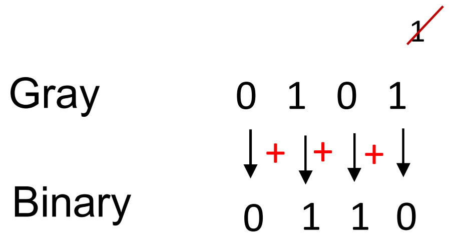
6. Binary codes

Conversion Gray number to Binary number

1. Start with the most significant bit (MSB). The MSB of the binary number is the same as the MSB of the Gray code number.
2. The bit next to the MSB (the second MSB) of the binary number is obtained by adding the MSB of the binary number to the second MSB of the Gray code number and ignoring any carry.
3. The third MSB of the binary number is obtained by adding the second MSB of the binary number to the third MSB of the Gray code number. Again, carry, if applicable, should be ignored.
4. The process continues until the LSB of the binary number is obtained.

Example:

$(0101)_{Gray} =(?)_2$



~~1~~

Gray 0 1 0 1

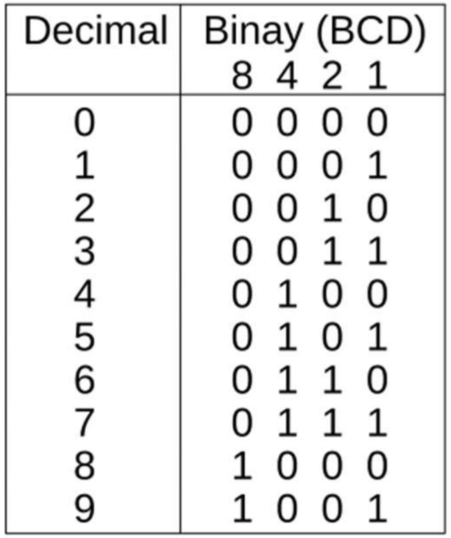
+ \+ \+

Binary 0 1 1 0

6. Binary codes

4.2. BCD code (Binary Coded Decimal)

- BCD code or Binary coded Decimal codes is a numeric **weighted binary codes**, where every digit of a decimal number is expressed by a separate group of 4-bits.
- There are various BCD codes like 8421, 2421, 5211, etc.
- The BCD code is also known as the 8421 code.



Decimal	Binay (BCD) 8 4 2 1
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	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

~~10~~	~~1 0 1 0~~
~~11~~	~~1 0 1 1~~
~~12~~	~~1 1 0 0~~
~~13~~	~~1 1 0 1~~
~~14~~	~~1 1 1 0~~
~~15~~	~~1 1 1 1~~

6. Binary codes

Examples

$(13)_{10}=(?)_{BCD}$

1				3			
0	0	0	1	0	0	1	1

⇨ $(13)_{10}=(0001\ 0011)_{BCD}$

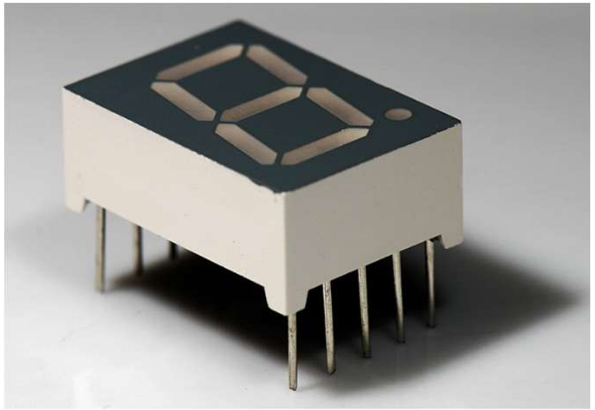
$(123)_{10}=(?)_{BCD}$

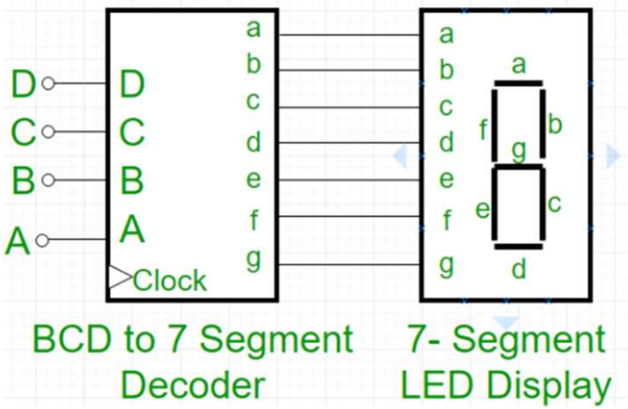
1				2				3			
0	0	0	1	0	0	1	0	0	0	1	1

Decimal	Binay (BCD)
	8 4 2 1
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	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

6. Binary codes

- The Binary Coded Decimal (BCD) format is a widely used format for storing data. It is useful for applications that require high accuracy in decimal calculations and frequent conversions between decimal and binary.
- Compared to the binary system, it is easy to code and decode binary-coded decimal numbers. Thus, binary-coded decimal offers a fast and efficient system to convert decimal numbers into binary numbers. One of the primary uses of BCD was for 7-segment displays. The integrated circuits that drive the 7-segment LEDs require 4 bits to output a value of 0-9





6. Binary codes

6.3. Excess 3 code

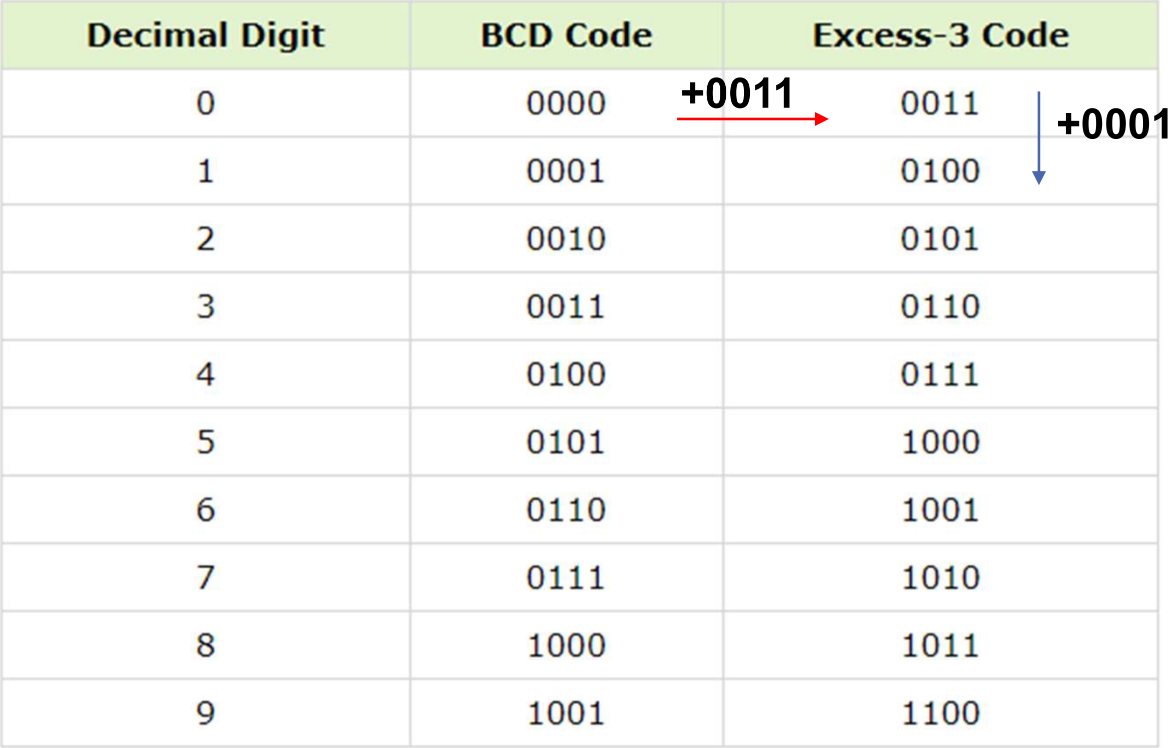
- Excess-3 codes are **unweighted** and can be obtained by adding 3 to each decimal digit then it can be represented by using 4 bit binary number for each digit.

- An Excess-3 equivalent of a given binary number is obtained using the following steps:
  Step 1 : Find the decimal equivalent of the given binary number.
  Step 2 : Add +3 to each digit of decimal number.
  Step 3 : Convert the newly obtained decimal number back to binary number to get required excess-3 equivalent.

You can add 0011 to each four-bit group in binary coded decimal number (BCD) to get desired excess-3 equivalent.

6. Binary codes

These are following excess-3 codes for decimal digits



Decimal Digit	BCD Code	Excess-3 Code
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

+0011

+0001

6. Binary codes

Decimal to Excess 3 code conversion

Decimal → 8421 BCD $\xrightarrow{+0011}$ Excess 3 code

Example :

$(23)_{10} = (?)_{\text{Excess 3}}$

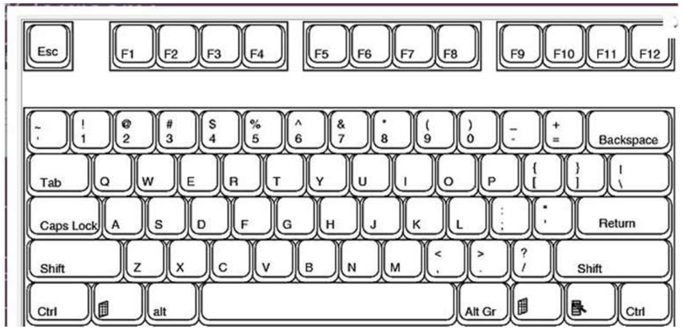
	2	3
	↓	↓
	0010	0011
	+0011	**+0011**
Excess 3 code	**0101**	**0110**

6. Binary codes

6.3. Alphanumeric code - ASCII

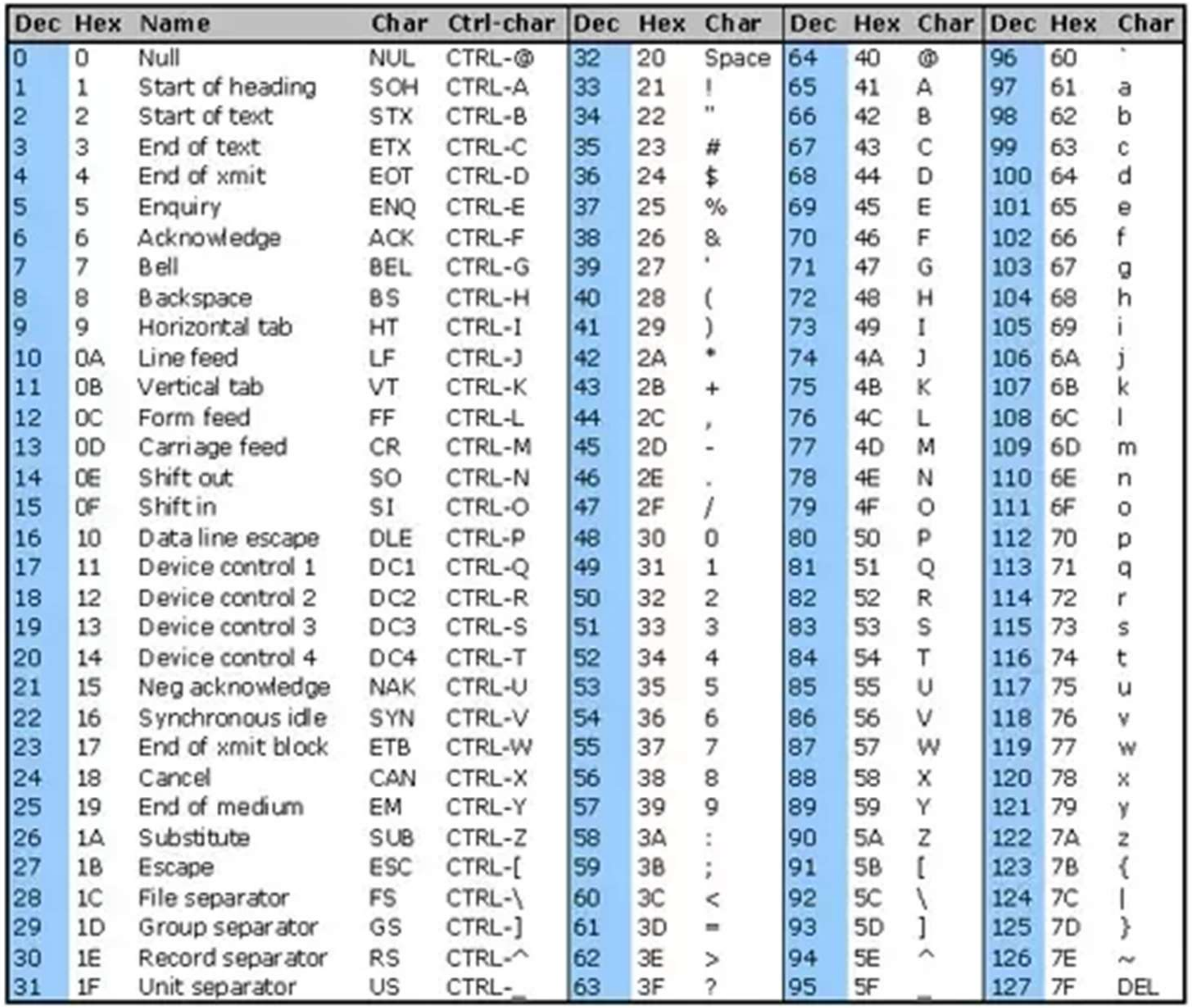
- ASCII stands for **American Standard Code for Information Interchange**.
- The American Standards Association (ASA) introduced it in 1963.
- Its form ranges from 0-127. Hence, it represents a total of 128 characters.
- **What is ASCII code used for?**

ASCII is used for encoding ordinary text into strings of bits, ***seven for each character***.
As computers operate on binary code, ***ASCII serves as an intermediary layer*** between human-readable text and machine-readable code. Each character is assigned exactly one value of 7 bits (for standard ASCII) or 8 bits (for extended ASCII).


Esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12
~ ` ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = Backspace
Tab Q W E R T Y U I O P { [}] | \
Caps Lock A S D F G H J K L : ; " ' Return
Shift Z X C V B N M < , > . ? / Shift
Ctrl alt Alt Gr Ctrl

- Computers can only understand numbers, so an ASCII code is the numerical representation of a character such as 'a' or '@' etc.

- The first 32 characters in the ASCII-table are unprintable control codes and are used to control peripherals such as printers.
- Codes 32-127 are common for all the different variations of the ASCII table, they are called printable characters, represent letters, digits, punctuation marks, and a few miscellaneous symbols.



Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

6. Binary codes

1) Convert the word ‘Blue’ to binary

Letter	Decimal	Binary						
		64	32	16	8	4	2	1
B	66	1	0	0	0	0	1	0
l	108	1	1	0	1	1	0	0
u	107	1	1	1	0	1	0	1
e	101	1	1	0	0	1	0	1

→ Blue: 1000010 1101100 1110101 1100101

6. Binary codes

2) Using ASCII table convert the following binary coded message into a word
1001100 1100101 1101101 1101111 1101110

First let’s get the decimal equivalent of each binary code

Binary							Decimal	Letter
64	32	16	8	4	2	1		
1	0	0	1	1	0	0	76	L
1	1	0	0	1	0	1	101	e
1	1	0	1	1	0	1	109	m
1	1	0	1	1	1	1	111	o
1	1	0	1	1	1	0	110	n

1001100 1100101 1101101 1101111 1101110 : Lemon

6. Binary codes

6.4. Error detecting Codes

In digital systems, the analog signals will change into digital sequence (in the form of bits). This sequence of bits is called as “Data stream”. The change in position of single bit also leads to major error in data output. Almost in all electronic devices, we find errors and we use error detection and correction techniques to get the exact or approximate output.

The data can be corrupted during transmission (from source to receiver). It may be affected by external noise or some other physical imperfections.
In this case, the input data is not same as the received output data. This mismatched data is called “Error”.

Tx Rx

0110 → 0111

Single bit error should be detect and correct by different schemes such as: Parity and Block parity.

6. Binary codes

Error detecting codes

Parity

- Parity is the simplest technique.
- There are two type of parity **Odd parity** and **Even parity**

  Even Parity

  - If the data has even number of 1’s, the parity bit is 0. Ex: data is 10000001 -> parity bit 0
  - Odd number of 1’s, the parity bit is 1. Ex: data is 10010001 -> parity bit 1

  Odd Parity

  - If the data has odd number of 1’s, the parity bit is 0. Ex: data is 10011101 -> parity bit 0
  - Even number of 1’s, the parity bit is 1. Ex: data is 10010101 -> parity bit 1

  NOTE: The counting of data bits will include the parity bit also.

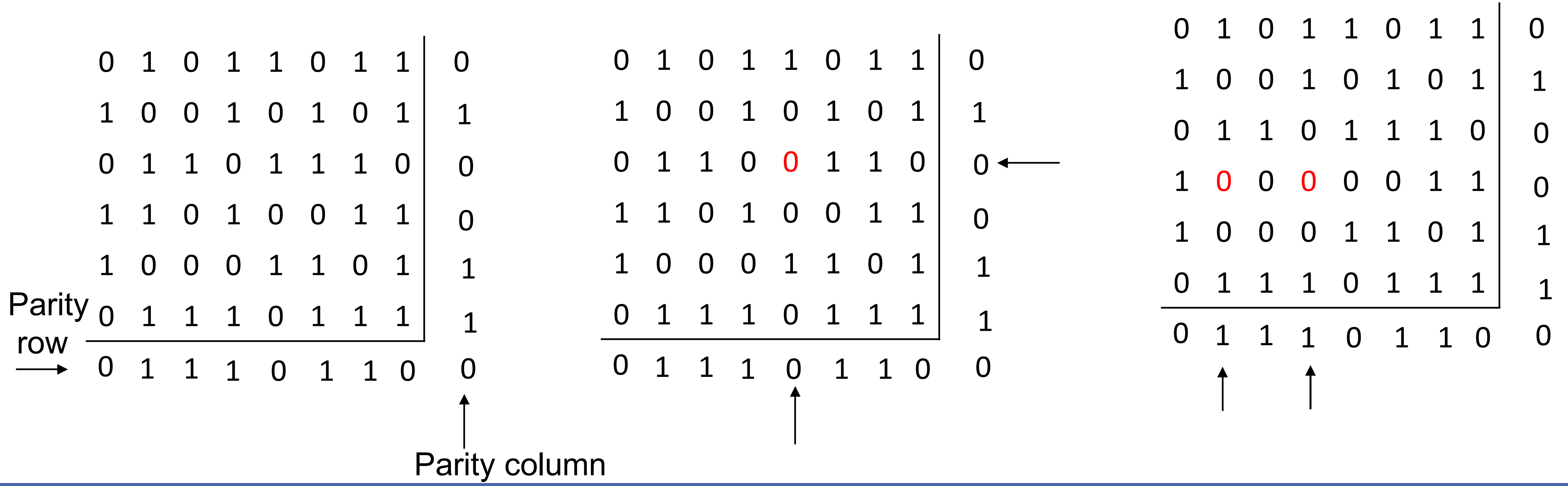
- Detect a single bit error but cannot detect two or more errors within the same word.
- Example: In an even parity sheme, code 10111001 is erroneous because number of 1s is odd(5), while code (11110110) is error free because number of 1s is even (6).

4. Binary codes

Error detecting codes

Block parity

We consider odd parity



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

Parity row

Parity column

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

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