Group Assignment-01

EE1201: Digital Systems

Submitted by

Group-3

Submitted to

Siva Vanjari Sir
Indian Institute of Technology Hyderabad

Date of Submission: February 2, 2025

1 Binary Codes

Digital systems operate using binary signals (0 and 1) and circuit elements that have two stable states. Binary numbers and other discrete information are represented using binary codes, which are patterns of 0s and 1s. These codes do not change the meaning of the information but provide a way for digital circuits to process it efficiently.

An **n-bit binary code** consists of n bits, allowing for 2^n distinct combinations, with each combination representing a unique element. For example, a two-bit code can represent four elements:

$$\{00, 01, 10, 11\}$$

while a three-bit code can represent eight elements. To avoid ambiguity, each element must have a unique binary combination.

While the minimum number of bits required to represent 2^n elements is n, there is no maximum limit. For instance, decimal digits (0-9) can be represented using a 10-bit code, where each digit is assigned a unique combination with a single 1 among nine 0s. For example, the digit 6 can be represented as:

$$0001000000$$

This illustrates how binary coding is essential for digital systems to function effectively.

1.1 Binary Coded Decimal Code

1.1.1 Binary vs. Decimal in Computers

- Computers use the **binary number system** because it is compatible with electronic technology.
- Humans are more familiar with the **decimal system** (base 10).
- Conversion between **decimal and binary** is often required for calculations.

1.1.2 Storing Decimal Numbers in Binary Form

- Since computers accept only binary values, decimal numbers must be represented using **binary-coded forms**.
- **Binary-Coded Decimal (BCD)** is one method of storing decimal numbers in binary.

1.1.3 Structure of BCD

- Each decimal digit (0–9) is represented using **4 bits**.
- A decimal number with k digits requires **4k bits** in BCD.
- Example: Decimal 396 in BCD:

$$396_{10} = 0011\ 1001\ 0110_{BCD}$$

1.1.4 Comparison: BCD vs. Binary Representation

- BCD is different from standard binary representation.
- Example:

$$\begin{aligned} 185_{10} &= 0001\ 1000\ 0101_{BCD} \quad \text{(12 bits)} \\ 185_{10} &= 10111001_2 \quad \text{(8 bits)} \end{aligned}$$

- BCD requires **more bits** than standard binary encoding.

1.1.5 Unused Bit Combinations in BCD

- A **4-bit binary** system provides **16 combinations** (0000 to 1111).
- Only **10 combinations** (0000 to 1001) are used for decimal digits.
- Combinations **1010 to 1111 are not used** in BCD.

1.1.6 Advantages & Disadvantages of BCD

- Advantage:
  - Easier for humans since input/output data remains in **decimal format**.
- Disadvantage:
  - BCD requires **more storage space** compared to binary representation.

1.1.7 BCD vs. Decimal Representation

- Decimal numbers use symbols **0–9**.
- BCD represents each decimal digit using a **4-bit binary code**.
- Example:

$$
\begin{aligned}
10_{10} &= 0001\ 0000_{BCD} \quad \text{(8 bits)}\\
10_{10} &= 1010_2 \quad \text{(4 bits)}
\end{aligned}
$$

1.1.8 BCD Addition

- When adding two BCD digits, the sum can range from 0 to 19 (including a carry from a previous operation).
- If the binary sum is greater than or equal to 1010, the result is invalid in BCD and requires correction.
- Adding 6 (0110) to the binary sum corrects the digit and adjusts the carry.
- Example calculations:

$$
\begin{aligned}
&4 + 5 = 9 \quad (0100 + 0101 = 1001)\\
&4 + 8 = 12 \quad (0100 + 1000 = 1100) \quad \text{(Invalid BCD, add 0110)}\\
&\qquad = 0010 \quad \text{(Correct BCD sum) with a carry}\\
&8 + 9 = 17 \quad (1000 + 1001 = 10001)\\
&\qquad\quad \text{(Requires correction, add 0110)}\\
&\qquad = 0111 \quad \text{(Correct BCD sum) with a carry}
\end{aligned}
$$

- The same method applies to multi-digit BCD addition, ensuring correct decimal results.
- Example: Adding 184 + 576 = 760 in BCD follows the same process.

1.1.9 Decimal Arithmetic in BCD

The representation of signed decimal numbers in BCD (Binary-Coded Decimal) follows a similar approach to signed binary numbers. Two main systems are used:

- Signed-magnitude system (rarely used in computers).

- Signed-complement system, which includes 9's complement and 10's complement (the latter being more common).

To compute the 10's complement of a BCD number:

1. Compute the 9's complement by subtracting each digit from 9.
2. Add 1 to the least significant digit.

For addition in the 10's complement system:

- Sum all digits, including the sign digit.
- Discard the end carry.

For subtraction, take the 10's complement of the subtrahend and add it to the minuend, similar to binary arithmetic. Many computers incorporate hardware to directly perform BCD arithmetic, allowing programmed instructions to handle decimal calculations without conversion to binary.

1.2 Weighted Codes

In weighted codes, each digit position is assigned a specific weight.

Code Type	**Weights**	**Example (Decimal 7)**
8421 Code (BCD)	8, 4, 2, 1	0111
2421 Code	2, 4, 2, 1	1100
5211 Code	5, 2, 1, 1	0111
8, 4, -2, -1 Code	8, 4, -2, -1	0110

Table 1: Comparison of Weighted Codes

1.2.1 Excess-3 (XS-3) Code

A non-weighted binary code where each decimal digit is represented by adding 3 before converting to binary.

Decimal	Binary	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

Table 2: Excess-3 Code Table

1.2.2 Gray Code

A special binary code where only one bit changes between successive values.

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

Table 3: Gray Code vs Binary Code

1.3 ASCII code

1.3.1 Introduction

Many digital computer applications require handling not only numerical data but also alphanumeric characters and symbols. To represent these characters, a binary encoding system is needed. One such system is the American Standard Code for Information Interchange (ASCII), which is widely used for encoding text.

1.3.2 ASCII Encoding

ASCII is a 7-bit character encoding system that can represent 128 characters. The encoding includes:

- 26 uppercase letters (A-Z)
- 26 lowercase letters (a-z)
- 10 decimal digits (0-9)
- 32 special printable characters (e.g., %, *, $)
- 34 control characters for formatting and communication

Each character in ASCII is represented by a unique 7-bit binary number. For example, the letter 'A' is represented as `1000001`.

1.3.3 Control Characters

The ASCII table includes 34 non-printable control characters, which are categorized as:

- **Format Effectors**: Control text layout (e.g., Backspace (BS), Carriage Return (CR), Horizontal Tab (HT)).
- **Information Separators**: Divide text into sections (e.g., Record Separator (RS), File Separator (FS)).
- **Communication-Control Characters**: Used in text transmission (e.g., Start of Text (STX), End of Text (ETX)).

1.3.4 ASCII and Byte Representation

Although ASCII is a 7-bit code, most computers use 8-bit bytes. The extra bit is often used for extended characters, such as Greek letters or italic fonts. Some systems set the most significant bit (MSB) to 0 for standard ASCII characters and to 1 for extended character sets.

1.3.5 Conclusion

ASCII provides a standardized method for encoding text in computers and digital communication. Its widespread adoption has made it a fundamental component of text processing and data exchange.

Table 1.7
American Standard Code for Information Interchange (ASCII)

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L		l	|
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	∧	n	~
1111	SI	US	/	?	O	–	o	DEL

Control Characters

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

Figure 1: ASCII CODE

1.4 Error Handling Code

To detect errors in data communication, an additional parity bit is added to ASCII characters, ensuring an even or odd number of 1's. Even parity is commonly used.

At the sender's end, parity bits are generated, and at the receiver's end, they are checked. If a parity error is detected, the receiver sends an NAK (Negative Acknowledge) signal:

$$\text{NAK} = 10010101$$

This prompts retransmission. If no error is found, an ACK (Acknowledge) signal is sent:

$$\text{ACK} = 00000110$$

If repeated errors occur, manual intervention is required.