

ENEE244-020x

Digital Logic Design

Lecture 3

Announcements

- Reminder: Check UTF Office Hours and contact information on Syllabus/Webpage
- Reminder: First homework assigned (see course webpage). Due date: Sept. 10 @ 11:59pm.
- Reminder: Readings for each lecture up on course webpage
- Reminder: First recitation is **tomorrow** (Wednesday, September 9).

Agenda

- Last time:
 - Signed numbers and Complements (1.5, 1.6)
 - Addition and Subtraction with Complements (1.5, 1.6)
- This time:
 - Overflow in 2's Complement
 - Binary Codes (1.7)
 - Parity and Arithmetic Modulo 2
 - Error detecting/correcting codes (ECC)
 - Not following presentation in textbook for ECC.

Example of Overflow in 2's complement

- Assume $\ell = 8$
- Compute:
 $01110000 + 01011100$

$[-128, +127]$

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

Overflow

- Compute
 $-01110000 - 01011100$

$$(10001111 + 1) + (10100011 + 1) +$$

$$10010000 + 10100100$$

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

Overflow

Overflow in 2's complement

- Overflow occurs in the following cases:

Operation	Operand A	Operand B	Result
A + B	≥ 0	≥ 0	< 0
A+B	< 0	< 0	≥ 0
A-B	≥ 0	< 0	< 0
A-B	< 0	≥ 0	≥ 0

- These conditions are the same as:
Carry-In to sign position $\neq$ Carry-Out from sign position

Binary Codes

Decimal Codes

- Different ways to represent individual decimal digits in binary
- Adding encoded digits requires a dedicated adder for each code
- As we will see, different encodings can provide different advantages.

BCD Code

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

$$\begin{array}{r}
 5 \quad 0101 \\
 + 5 \quad 0101 \\
 \hline
 10 \quad 1010 \\
 + 110 \\
 \hline
 10000
 \end{array}$$

0001
0000

$$\begin{array}{r}
 4 \quad 0100 \\
 + 5 \quad 0101 \\
 \hline
 9 \quad 1001
 \end{array}$$

Advantage: Decimal addition can be performed over the binary representation. Just need to add 110 to the sum if it is equal to 10 or above.

Other Codes

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, -2, -1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
	1010	0101	0000	0001
Unused bit combinations	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

1011 0101

Weighted codes: BCD (8421), 2421, 8,4,-2,-1

Each binary digit is multiplied by the corresponding weight to obtain the decimal value.

Other Codes

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, -2, -1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
	1010	0101	0000	0001
Unused bit combinations	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Excess 3 code: obtained from the binary value plus 3

Other Codes

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, -2, -1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
	1010	0101	0000	0001
Unused bit combinations	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Advantages: 2421, Excess-3, 8,4,-2,-1 codes are **self-complementing**
 The 9's complement is obtained by changing 1's to 0's and 0's to 1's.

Other Codes

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

Advantage: Consecutive values differ in exactly one bit

If multiple bits must change when transitioning from one number to the next, they may not switch simultaneously in a physical device, potentially producing incorrect intermediate values.

ASCII Character Code

- American Standard Code for Information Interchange (ASCII)
- Binary code for representing *characters*
- Includes decimal digits (0-9) as before but now also includes
 - Letters (uppercase and lowercase)
 - Special characters (such as \$#@!)
- Uses seven bits to encode 128 characters

ASCII Character Code

↓ ↓ ↓

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	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

1000001
" 65

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

Parity and Arithmetic Mod 2

Parity

- Parity 0: A 0/1 string has an even number of 1's.
 - Example: 001011100
- Parity 1: A 0/1 string has an odd number of 1's.
 - Example: 101010000
- Given a string, can also ask about the parity of a subset of positions
 - Example: Parity of positions 1, 3, 5, 6 in the string 001011100 is 1.

Mod 2 Arithmetic

- $(N \bmod 2)$ is the remainder when dividing N by 2
 - 0 when N is even
 - 1 when N is odd
- Parity of a string is the sum of the bits modulo 2
 - Example: $001011100 = 0 + 0 + 1 + 0 + 1 + 1 + 1 + 0 + 0 = 4 = 0 \bmod 2 = 0$.
- Parity of a subset of a string is exactly the dot-product mod 2.
 - Example: Parity of positions 1, 3, 5, 6 in the string 001011100 is the dot product 101011000 .
 $001011100 = 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 + 0 \cdot 0 = 3 \bmod 2 = 1$.

Codes for Error Detection and Correction

Codes

- Encode algorithm $Enc(\vec{m}) = \vec{c}$. $\vec{m}$ is the **message**, $\vec{c}$ is the **codeword**.
- Decode algorithm $Dec(\vec{c}) = \vec{m}$
- Typically, $\vec{c}$ will be longer than $\vec{m}$ and will include **redundant** information.
- Redundancy is useful for detecting and/or correcting errors introduced during transmission.
- Assume $\vec{m}, \vec{c}$ are in binary.
- Would like to detect and/or correct the flipping of one or multiple bits.

Error Detection/Correction

- Basic properties:
 - Distance of a code: minimum distance between any two codewords (number of bits that need to be flipped to get from one codeword to another)
 - Rate of a code: $\frac{|\vec{m}|}{|\vec{c}|}$ (length of $\vec{m}$ / length of $\vec{c}$)
- Distance determines the maximum number of errors that can be detected/corrected.
- Goal of coding theory is to construct codes with optimal tradeoff between distance and rate.
- Must also have efficient encoding, decoding and error correcting procedures.

Error Detection/Correction

- Error detection: can detect at most $dist-1$ errors, where $dist$ is the minimum distance of the code.
- Error correction: can correct at most $(dist - 1)/2$ errors

Error Detection:

Parity Check

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 0$$

- Encode: On input $\vec{m} = 11001010$
 - Output $\vec{c} = 11001010|b$, where b is the parity of $\vec{m}$.
 $b = 1 + 1 + 0 + 0 + 1 + 0 + 1 + 0 = 4 \bmod 2 = 0$
- Decode: On input $\vec{c} = 11001010|b$, output 11001010
- Error detection:
 - If a non-parity bit is flipped
 - If the parity bit is flipped
- Can detect only one error. Why?

Is every 9-bit
string w/ even
parity a valid
codeword?

Error Correction for 1 Error:

$[7, 4]$

The Hamming Code

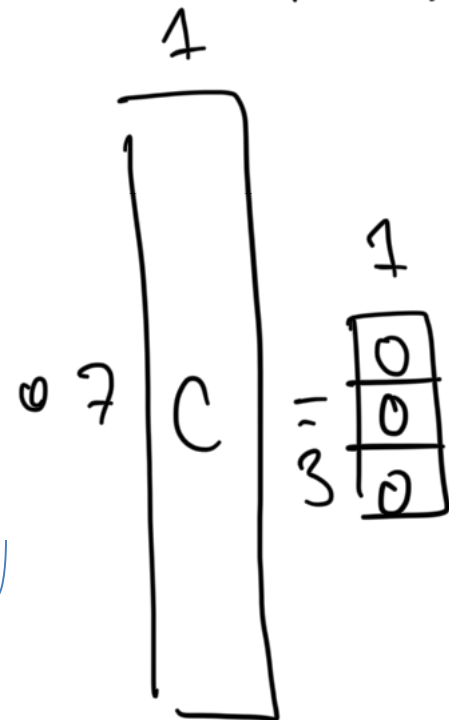
codewords have
length 7 bits

- View codeword as a vector $(c_1, c_2, \dots, c_7)$ messages have length 4 bits.
- Some bits will be information bits, some bits will be parity-check bits.
- Parity-check matrix H:

k parity bits

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$2^k - 1$ codeword length



Property of the Hamming Code

- For any codeword $\vec{c}$, $H \cdot \vec{c} = \vec{0}$.
- Parity-check matrix H :

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix} = (0 \quad 0 \quad 0)^T$$

Property of the Hamming Code

- To encode a message $\vec{m} = m_1, m_2, m_3, m_4$

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix} = (0 \quad 0 \quad 0)^T$$



 Message bit positions

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Message bit positions

C_1 must be equal to parity of C_3, C_5, C_7

$$\begin{pmatrix} 1 & C_1 \\ 1 & C_2 \\ & C_4 \\ & 1 \\ & 1 \\ & 0 \end{pmatrix}$$

C_4 must be equal to parity of C_5, C_6, C_7

C_2 must be equal to parity of C_3, C_6, C_7

1100110

Property of the Hamming Code

- To encode a message $\vec{m} = m_1, m_2, m_3, m_4$

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}}_{\text{Parity bit positions}} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix} = (0 \quad 0 \quad 0)$$

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

$$\begin{pmatrix}
 \boxed{0} & \boxed{0} & 0 & \boxed{1} & 1 & 1 & 1 \\
 \boxed{0} & \boxed{1} & 1 & \boxed{0} & 0 & 1 & 1 \\
 \boxed{1} & \boxed{0} & 1 & \boxed{0} & 1 & 0 & 1
 \end{pmatrix} \cdot \begin{pmatrix}
 \boxed{c_1} \\
 \boxed{c_2} \\
 0 \\
 \boxed{c_4} \\
 1 \\
 1 \\
 0
 \end{pmatrix} = (0 \quad 0 \quad 0)$$

Parity bit positions

Put in a value in c_4 so that

$$\begin{aligned}
 &0 \cdot c_1 + 0 \cdot c_2 + 0 \cdot 0 + 1 \cdot c_4 + \\
 &\quad 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 \\
 &= c_4 + 0 = 0 \text{ mod } 2
 \end{aligned}$$

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

$$\begin{pmatrix}
 \boxed{0} & \boxed{0} & 0 & \boxed{1} & 1 & 1 & 1 \\
 \boxed{0} & \boxed{1} & 1 & \boxed{0} & 0 & 1 & 1 \\
 \boxed{1} & \boxed{0} & 1 & \boxed{0} & 1 & 0 & 1
 \end{pmatrix} \cdot \begin{pmatrix}
 \boxed{c_1} \\
 \boxed{c_2} \\
 \boxed{0} \\
 1 \\
 1 \\
 0
 \end{pmatrix} = (0 \quad 0 \quad 0)$$

Parity bit positions

Put in a value in c_4 so that

$$\begin{aligned}
 &0 \cdot c_1 + 0 \cdot c_2 + 0 \cdot 0 + 1 \cdot c_4 + \\
 &\quad 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 \\
 &= c_4 + 0 = 0 \text{ mod } 2
 \end{aligned}$$

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}}_{\text{Parity bit positions}} \cdot \begin{pmatrix} c_1 \\ c_2 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = (0 \quad 0 \quad 0)$$

Put in a value in c_2 so that

$$\begin{aligned}
 &0 \cdot c_1 + 1 \cdot c_2 + 1 \cdot 0 + 0 \cdot c_4 + \\
 &\quad 0 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 \\
 &= c_2 + 1 = 0 \text{ mod } 2
 \end{aligned}$$

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}}_{\text{Parity bit positions}} \cdot \begin{pmatrix} c_1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = (0 \quad 0 \quad 0)$$

Put in a value in c_2 so that

$$\begin{aligned}
 &0 \cdot c_1 + 1 \cdot c_2 + 1 \cdot 0 + 0 \cdot c_4 + \\
 &\quad 0 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 \\
 &= c_2 + 1 = 0 \text{ mod } 2
 \end{aligned}$$

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

$$\begin{pmatrix}
 \boxed{0} & \boxed{0} & 0 & \boxed{1} & 1 & 1 & 1 \\
 \boxed{0} & \boxed{1} & 1 & \boxed{0} & 0 & 1 & 1 \\
 \boxed{1} & \boxed{0} & 1 & \boxed{0} & 1 & 0 & 1
 \end{pmatrix} \cdot \begin{pmatrix}
 \boxed{c_1} \\
 \boxed{1} \\
 \boxed{0} \\
 \boxed{0} \\
 1 \\
 1 \\
 0
 \end{pmatrix} = (0 \quad 0 \quad 0)$$

Parity bit positions

Put in a value in c_1 so that

$$\begin{aligned}
 &1 \cdot c_1 + 0 \cdot c_2 + 1 \cdot 0 + 0 \cdot c_4 + \\
 &\quad 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 0 \\
 &= c_1 + 1 = 0 \text{ mod } 2
 \end{aligned}$$

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

Put in a value in c_1 so that

$$1 \cdot c_1 + 0 \cdot c_2 + 1 \cdot 0 + 0 \cdot c_4 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 0 = c_1 + 1 = 0 \text{ mod } 2$$

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = (0 \quad 0 \quad 0)^T$$

Parity bit positions

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}}_{\text{Parity bit positions}} \cdot \underbrace{\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}}_{\text{Codeword}} = (0 \quad 0 \quad 0)$$

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}}_{\text{Parity bit positions}} \cdot \underbrace{\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}}_{\text{Codeword}} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

The diagram illustrates the Hamming code parity check. The matrix on the left represents the parity bit positions for a 7-bit codeword. The vector on the right represents the received codeword. The product of the matrix and the codeword is shown as a 2x3 matrix of results. A blue box highlights the third row of the codeword (1), and a blue box highlights the second row of the result matrix (1 0 0). A speech bubble points to the third row of the codeword with the text "Error!".

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}}_{\text{Parity bit positions}} \cdot \underbrace{\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}}_{\text{Corrupted Codeword}} = (1 \quad 0 \quad 0)$$

Property of the Hamming Code

- To encode a message $\vec{m} = 0, 1, 1, 0$

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}}_{\text{Parity bit positions}} \cdot \underbrace{\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}}_{\text{Corrupted Codeword}} = (1 \quad 0 \quad 0)$$