DEI
DIPARTIMENTO DI
INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sistemi Digitali

Sistemi di numerazione, conversioni, aritmetica, codifiche

Marta Bagatin, marta.bagatin@unipd.it

Corso di Laurea in Ingegneria dell'Informazione

Anno accademico 2022-2023

Scopo della lezione

- I sistemi digitali operano su elementi discreti e rappresentati in forma binaria
  - Gli operandi usati nei calcoli all'interno di un computer, le lettere dell'alfabeto, etc. sono tutti rappresentati attraverso codici binari
- Lo scopo di questa lezione è studiare diversi **sistemi di numerazione** (**binario**, ottale, esadecimale) e l'aritmetica ad essi associata
- Studiare i **codici binari** più usati
  - Cifre decimali: BCD (Binary-Coded Decimal)
  - Caratteri alfanumerici (ASCII)
  - Codici Gray
- Bit di parità

Sistema decimale, binario, ottale ed esadecimale

Il sistema decimale

- Storicamente l'uomo fa uso del sistema numerico decimale (10 dita!)
- E' un sistema di **notazione posizionale** e fa uso di 10 cifre da 0 a 9
  - A seconda della sua posizione all'interno del numero, ciascuna cifra viene moltiplicata per la corrispondente potenza di 10
  - **Es**: 384 => 3 centinaia, 8 decine e 4 unità

384_{10}

2719.5_{10}

Base generica r

- Un numero $a_{n-1}\ a_{n-2} \ldots a_1\ a_0$ in base *r* rappresenta il valore

$$\sum_{i=0}^{n-1} a_i\, r^i$$

a_{n-1} : Cifra più significativa
a_0 : Cifra meno significativa

*<<Nella numerazione posizionale, il valore di una cifra dipende dalla posizione che occupa nella trascrizione del numero. Per dirla in una parola, la posizione 'conta'! Il numero uno vale uno, dieci o cento, a seconda che occupi l'ultimo, il penultimo o il terzultimo posto>>… Assumendo il tono di un vecchio saggio orientale, declamò: <<**Un nano seduto sul gradino più alto è più alto di un gigante che sta in piedi sul più basso**. Antico proverbio arabo>>... Il signor Ruche prese la palla al balzo. << E il numero uno di 1000 vale di più dei tre nove di 999>>*

(cit. Denis Guedj, Il Teorema del Pappagallo)

Il sistema binario

- Il sistema binario usa **2 cifre**, **0** e **1**
  - Sistema di **notazione posizionale**: a seconda della sua posizione all'interno del numero, ciascuna cifra viene moltiplicata per la corrispondente potenza di 2
  - **MSB** (Most Significant Bit) è il bit più a sinistra
  - **LSB** (Most Significant Bit) è il bit più a destra
  - Es: 10011_2

    110101.11_2

Potenze di 2

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

2^{10}, ~10^3 → K (kilo)

2^{20} , ~10^6 → M (mega)

2^{30}, ~10^9 → G (giga)

2^{40}, ~10^{12} → T (tera)

Es: una memoria da 4 Gbyte contiene $4 \cdot 2^{30} \cdot 8$ bit = $2^2 \cdot 2^{30} \cdot 2^3$ bit = 2^{35} bit

Conversione decimale-binario

- La procedura consiste nel sottrarre successivamente potenze di 2 dal numero decimale

1) Trovare la più grande potenza di 2 che, sottratta da N, produce una differenza positiva, che chiamiamo N1
2) Trovare la più grande potenza di 2 che, sottratta da N1, produce una differenza positiva, che chiamiamo N2
3) Continuare la procedura finchè si ottene 0

Conversione decimale-binario

– Esempio: convertire 625_{10} in numero binario

Sistema ottale e sistema esadecimale

- I sistemi in ottale (**base 8, OCT**) ed esadecimale (**base 16, HEX**) sono molto usati (8 e 16 sono potenze di 2)
  - $2^3 = 8$: ogni cifra ottale corrisponde a 3 cifre binarie
  - $2^4 = 16$: ogni cifra esadecimale corrisponde a 4 cifre binarie
- Questi sistemi forniscono rappresentazioni più compatte, che risultano più convenienti rispetto ai numeri binari, usando stringhe 3 o 4 volte più corte!

Sistema esadecimale

- Usa **16 cifre: da 0 a 9** più le lettere **A, B, C, D, E** e **F**, per i valori 10, 11, 12, 13, 14 e 15

  - **Es**: $(B65F)_{16}$

Conversione da binario a esadecimale

- Si divide il numero binario in gruppi di 4 bit, iniziando dalla virgola e procedendo a sinistra e poi a destra. Infine si assegna ad ogni gruppo di bit la corrispondente cifra esadecimale
  - **Es**: $(0010\ 1100\ 0110\ 1011.\ 1111\ 0000\ 0110)_2$

Conversione da esadecimale a binario

- Consiste nella procedura inversa
  - **Es**: $(3A6.C)_{16}$

Sistema ottale

- E’ meno usato rispetto al sistema esadecimale
- Il sistema ottale usa **8 cifre, da 0 a 7**
- Ogni cifra nel numero ottale è pesata con la corrispondente potenza di 8
  - Es: $(127.4)_8 = 1 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0 + 4 \cdot 8^{-1} = (87.5)_{10}$
- **Conversione da binario a ottale**: analogo a esadecimale. Si divide il numero binario in gruppi di 3 bit, iniziando dalla virgola e procedendo a sx e poi a dx. Infine si assegna ad ogni gruppo di bit la corrispondente cifra ottale
- **Conversione da ottale a binario**: procedura inversa

Numeri in base 10, 2, 8 e 16

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Aritmetica con il sistema binario

Operazioni aritmetiche

- Le operazioni aritmetiche in sistemi diversi dal decimale seguono le **stesse regole usate in base 10**
  - Le cifre disponibili e il peso in ciascuna posizione cambiano a seconda del sistema usato!

Somma binaria

- La somma di due cifre può valere 0 o 1
  - 0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1; 1 + 1 = 0 con riporto di 1
- C’è un **riporto** se entrambe le cifre sono 1

Somma binaria

- La somma di due cifre può valere 0 o 1
  - 0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1; 1 + 1 = 0 con riporto di 1
- C'è un **riporto** se entrambe le cifre sono 1
- Esempio 1

		Binario					Decimale
Posizione		16	8	4	2	1	
Riporto							
Addendo1		0	1	1	0	0	
Addendo2	+	1	0	0	0	1	
Somma							

Somma binaria

- La somma di due cifre può valere 0 o 1
  - 0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1; 1 + 1 = 0 con riporto di 1
- C'è un **riporto** se entrambe le cifre sono 1
- Esempio 2

		Binario					Decimale
Posizione		32	16	8	4	2	1	
Riporto							
Addendo1			1	0	1	1	0	
Addendo2	+		1	0	1	1	1	
Somma								

Differenza binaria

- La differenza di due cifre può valere 0 o 1
- Un **prestito** aggiunge 2 nella cifra adiacente a destra (così come nel sistema decimale aggiunge 10)
- Esempio 1

		Binario					Decimale
Posizione		32	16	8	4	2	1	
Prestito							
Minuendo			1	0	1	1	0	
Sottraendo	-		1	0	0	1	0	
Differenza								

Differenza binaria

- La differenza di due cifre può valere 0 o 1
- Un **prestito** aggiunge 2 nella cifra adiacente a destra (così come nel sistema decimale aggiunge 10)
- Esempio 2

	Binario					Decimale
Posizione		16	8	4	2	1	
Prestito							
Minuendo		1	0	1	1	0	
Sottraendo	-	1	0	0	1	1	
Differenza							

Differenza binaria (minuendo < sottraendo)

- Se il minuendo è più piccolo del sottraendo, si scambiano gli operandi e alla fine si inverte il segno del risultato
- Esempio: 10011-11110

		Binario					Decimale
Posizione		16	8	4	2	1	
Prestito							
Minuendo		1	1	1	1	0	
Sottraendo	-	1	0	0	1	1	
Differenza							

Moltiplicazione binaria

- Le cifre da moltiplicare sono sempre 0 o 1
- I **prodotti parziali** sono uguali al moltiplicando o uguali a 0
- Esempio

	Binario						Decimale
Posizione	32	16	8	4	2	1	
Moltiplicando			1	0	1	1	
Moltiplicatore				1	0	1	
Prodotto parz. 1							
Prodotto parz. 2							
Prodotto parz. 3							
Prodotto							

Operazioni aritmetiche

- Operazioni tra numeri in basi diverse dalla base 2 seguono regole analoghe
- In generale, **operazioni tra numeri espressi in base r possono essere eseguite facendo le operazioni tra i rispettivi numeri in base 10 e convertendo i risultati nella base desiderata**

Conversione da decimale ad altre basi

- Per convertire un numero decimale intero in un numero in **base r**: si **divide il numero e i successivi quozienti per r** e si tiene traccia dei **resti**
- Si considerano i **resti in ordine inverso** e si ottiene la conversione nella base desiderata

Conversione da decimale ad altre basi

- Per convertire un numero decimale intero in un numero in **base r**: si **divide il numero e i successivi quozienti per r** e si tiene traccia dei **resti**
- Si considerano i **resti in ordine inverso** e si ottiene la conversione nella base desiderata
  - **Es**: Convertire 2471_{10} in numero ottale

Conversione da decimale ad altre basi

- Questo metodo funziona ovviamente anche per la conversione in base 2
  - **Es**: Conversione di 41_{10} in numero binario

Codifiche

Codifica

- E’ un codice che **associa un insieme di simboli ad un insieme discreto di elementi** (per esempio numeri: ASCII, impulsi elettrici: codice Morse, etc.)
- Lo scopo di una codifica è facilitare la **memorizzazione di dati** o la loro **trasmissione** attraverso apposite reti
- Un **codice binario a n bit** è un gruppo di n bit che può assumere 2^n combinazioni distinte di 1 e 0
  - Un insieme di 4 elementi può essere codificato con un codice binario a 2 bit: 00, 01, 10, 11
  - Un insieme di 8 elementi può essere codificato con un codice binario a 3 bit: 000, 001, 010, 011, 100, 101, 110, 111
  - 16 elementi con 4 bit, e così via …

Codici decimali

- I calcolatori lavorano con numeri binari, ma l'uomo è abituato al sistema decimale: è comodo **convertire i numeri decimali in binario**, eseguire le operazioni in base 2 e infine riconvertire i risultati in base 10
- La **codifica BCD (Binary Coded Decimal)** è un **modo di rappresentare le cifre decimali con un codice binario**
  - Per rappresentare in base 2 le 10 cifre decimali servono **4 bit**

Binary-Coded Decimal (BCD)

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

Ogni cifra decimale viene rappresentata con la sua codifica binaria a 4 bit

Binary-Coded Decimal (BCD)

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

Es: $(396)_{10} = (0011\ 1001\ 0110)_{BCD}$

-> gruppi di 4 bit, ognuno rappresenta la **codifica binaria di ciascuna cifra decimale**

- La rappresentazione BCD di un numero maggiore di 10_{10} è **diversa** dalla sua rappresentazione binaria!
  - Es: $(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2$

Binary-Coded Decimal (BCD)

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

Es: $(396)_{10} = (0011\ 1001\ 0110)_{BCD}$

-> gruppi di 4 bit, ognuno rappresenta la **codifica binaria di ciascuna cifra decimale**

- Ci sono 6 combinazioni binarie che **non hanno significato nel codice BCD**: 1010, 1011, 1100, 1101, 1110, 1111
- Sebbene comporti uno **spreco di bit**, il BCD è comodo nelle applicazioni con ingressi e uscite costituite da numeri decimali

Codici alfanumerici

- Oltre che i numeri, anche le **lettere**, i **simboli** e i **caratteri speciali** hanno bisogno di una **rappresentazione binaria**
- Il codice binario standard per rappresentare i caratteri alfanumerici è chiamato codifica **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange), storicamente sviluppato per il linguaggio anglosassone (pubblicato dall'American National Standards Institute nel 1968)

Codice ASCII

	$B_7B_6B_5$							
$B_4B_3B_2B_1$	000	001	010	011	100	101	110	111
0000	NULL	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

- 7 bit per codificare 128 caratteri ($B_7B_6B_5$: colonna; $B_4B_3B_2B_1$: riga)
- Spesso è immagazzinato in 1 byte, con il primo bit a ‘0’ o usato per caratteri aggiuntivi

Codice ASCII

	$B_7B_6B_5$							
$B_4B_3B_2B_1$	000	001	010	011	100	101	110	111
0000	NULL	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

- Il codice BCD di una cifra è **direttamente legato con il codice ASCII**: è sufficiente aggiungere 011 a sinistra della codifica BCD per ottenere il corrispondente ASCII

Unicode

- E’ un sistema di codifica che è stato sviluppato nel 1991, successivamente al codice ASCII, con lo scopo di colmarne i limiti (e.g. i caratteri internazionali non rappresentati da ASCII)
- E’ uno standard industriale per fornire una **codifica comune a tutte le lingue** (include lettere accentate, ideogrammi, etc.)
- Assegna una codifica binaria univoca ad ogni carattere in modo indipendente dalla lingua, dalla piattaforma informatica e dal programma usato

Unicode

- **Notazione Unicode**: “U+” seguito dalle cifre esadecimali che corrispondono al carattere, chiamate **code point**
- Ci sono diverse codifiche standard dei code point Unicode (UTF-8, UTF-16, UTF-32)
- Per esempio, la codifica UTF-8 usa un numero di byte che varia da 1 a 4 per identificare ciascun code point

Unicode: Codifica UTF-8

UTF-8 Encoding for Unicode Code Points

Code point range (hexadecimal)	UTF-8 encoding (binary, where bit positions with x are the bits of the code point value)
U+0000 0000 to U+0000 007F	0xxxxxxx
U+0000 0080 to U+0000 07FF	110xxxxx 10xxxxxx
U+0000 0800 to U+0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
U+0001 0000 to U+0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

- Esempi:
  - **T** : code point **U+0054**, nel range U+0000 0000 - U+0000 007F -> codificato con 1 byte: $(01010100)_2$
  - **±** : code point **U+00B1**, nel range U+0000 0080 - U+0000 07FF -> codificato con 2 bytes: $(11000010\ 10110001)_2$

Bit di parità

- L'**aggiunta di un bit di parità** ad una codifica binaria fornisce un **metodo di controllo** per rilevare possibili errori nella elaborazione, trasferimento e immagazzinamento dei dati
- E' un bit addizionale scelto in maniera tale che il **numero totale di '1' nella codifica sia pari** (parità pari, più usata) o dispari (parità dispari)
  - Esempi
    - **1000001**
      - con parità pari: **0**1000001; con parità dispari **1**1000001
    - **1010100**
      - con parità pari: **1**1010100; con parità dispari **0**1010100

Bit di parità

- Questo codice di controllo (ma non di correzione degli errori!) viene usato con i numeri binari, ma anche con codici ASCII (il bit di controllo non deve essere obbligatoriamente il MSB)
- **Es.**: trasmissione di codice ASCII con bit di parità
  - Vengono trasmessi 8 bit: 7 bit del codice ASCII del carattere + 1 bit di parità
  - Dal lato del ricevitore viene controllata la parità del carattere trasmesso
    - Se non c'è parità (e.g. un numero pari di 1), almeno un bit è stato corrotto nella trasmissione – il ricevitore può mandare un segnale di NAK (negative acknowledge)
    - Se c'è parità significa che non ci sono stati errori su un singolo bit– il ricevitore può mandare un segnale di ACK (acknowledge)
  - In caso di errore, a seconda dell'applicazione
    - Può essere chiesta una nuova trasmissione del carattere
    - Se l'errore continua a ripetersi viene segnalato un malfunzionamento della trasmissione

Codice Gray

- Per alcune applicazioni è vantaggioso che le **codifiche di numeri consecutivi differiscano di un solo bit**
- Il codice Gray è un codice binario che prevede che passando da un numero al successivo cambi un solo bit

Binary Code	Bit Changes	Gray Code	Bit Changes
000		000	
001	1	001	1
010	2	011	1
011	1	010	1
100	3	110	1
101	1	111	1
110	2	101	1
111	1	100	1
000	3	000	1

Codice Gray: perché usarlo?

- Semplifica e rende **meno soggette ad errori le operazioni di dispositivi elettronici che devono scorrere informazioni organizzate in sequenze**
- Supponiamo che un dispositivo indichi la propria posizione chiudendo e aprendo degli interruttori. Se la posizione viene rappresentata con una codifica binaria ordinata in modo naturale, le posizioni 3 e 4 sono consecutive, ma hanno tutti i bit di valore diverso:

Decimale	Binario
1	000
2	001
3	**011**
4	**100**

- Ciò può essere fonte di problemi con interruttori reali, essendo improbabile che questi cambino il proprio stato (aperto/chiuso) tutti nello stesso istante
- Oscillazioni/ambiguità nella posizione

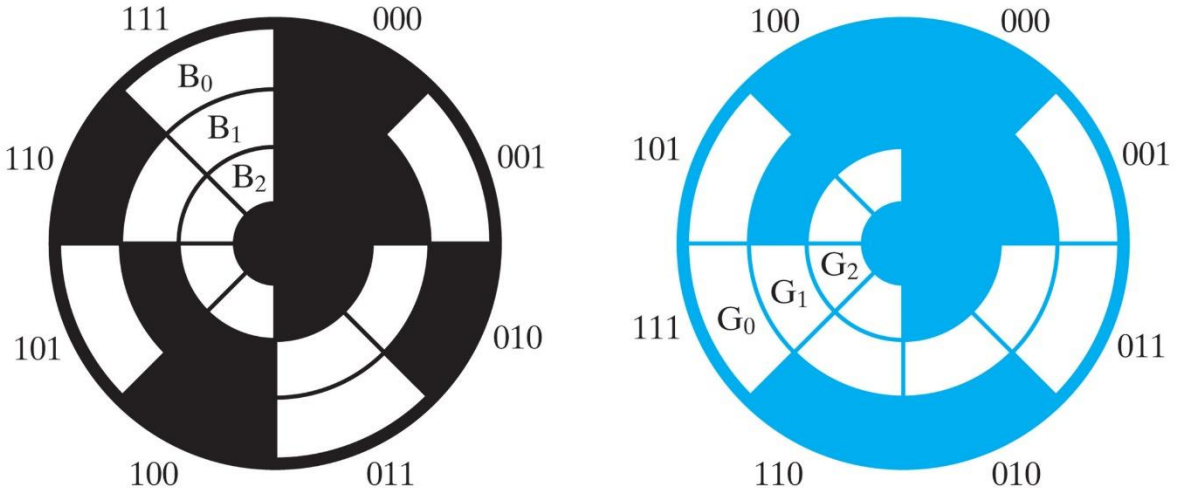
Codice Gray: perché usarlo?

- Semplifica e rende **meno soggette ad errori le operazioni di dispositivi elettronici che devono scorrere informazioni organizzate in sequenze**
- Supponiamo che un dispositivo indichi la propria posizione chiudendo e aprendo degli interruttori. Se la posizione viene rappresentata con una codifica binaria ordinata in modo naturale, le posizioni 3 e 4 sono consecutive, ma hanno tutti i bit di valore diverso:

Gray Code	Bit Changes
000	
001	1
011	1
010	1
110	1
111	1
101	1
100	1
000	1

- Una numerazione che utilizza una codifica in cui si commuta un solo interruttore alla volta (codifica Gray, cambia un solo bit alla volta) risolve il problema delle oscillazioni

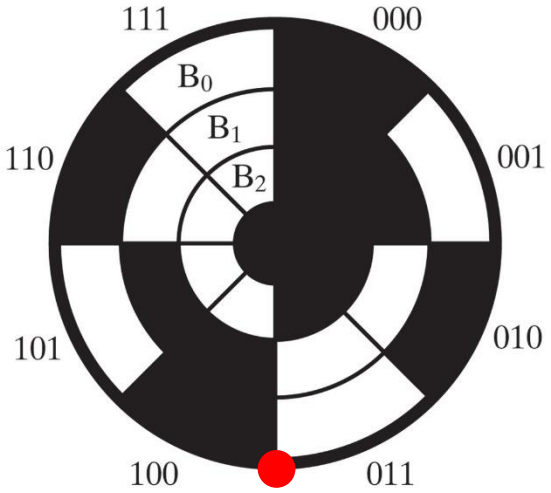
Codice Gray: esempio di applicazione



(a) Binary code for positions 0 through 7 (b) Gray code for positions 0 through 7

- Il codice Gray fu brevettato da Frank Gray (1953) per l'**encoder** che converte una posizione angolare in un codice digitale
  - Disco (solidale con albero rotante) contiene aree trasparenti e opache
  - Per ogni bit da decodificare c'è un LED (la cui uscita è digitale, vale 0 o 1) e un sensore ottico da parti opposte del disco: se la regione è trasparente il sensore legge '1', se opaca legge '0'

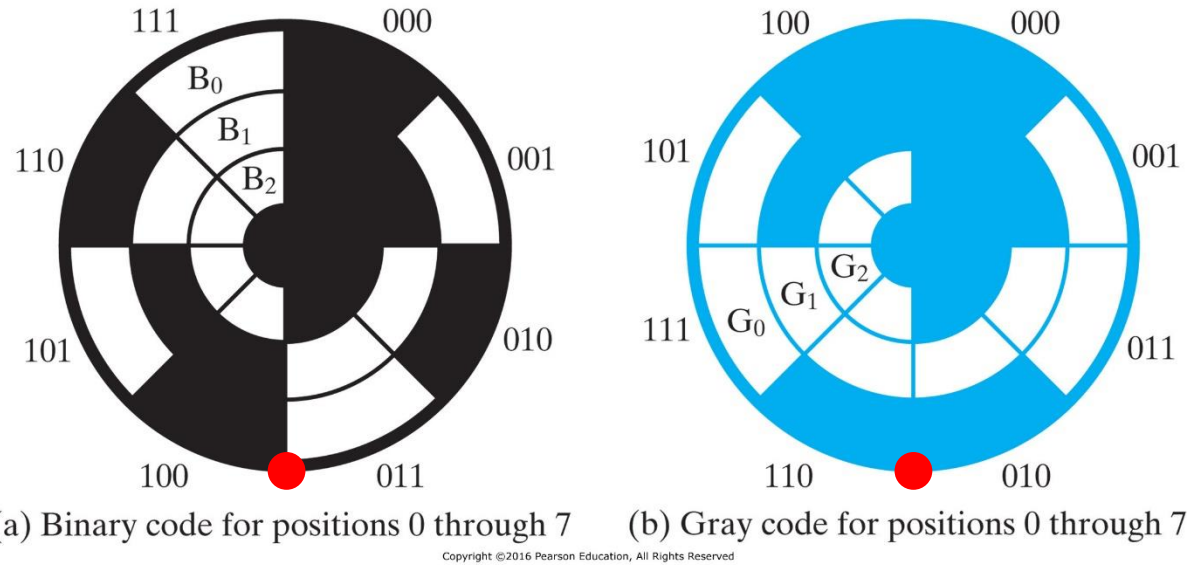
Codice Gray: esempio di applicazione



(a) Binary code for positions 0 through 7

- Caso in cui il sensore si trovi sul confine tra 2 posizioni adiacenti
  - Sensore tra 011 e 100: con codifica binaria, B_0 B_1 B_2 possono essere letti come ‘0’ o ‘1’ (8 possibilità, di cui 6 errate!)

Codice Gray: esempio di applicazione



(a) Binary code for positions 0 through 7 (b) Gray code for positions 0 through 7

- Caso in cui il sensore si trovi sul confine tra 2 posizioni adiacenti
  - Sensore tra 011 e 100: con codifica binaria, B_0 B_1 B_2 possono essere letti come ‘0’ o ‘1’ (8 possibilità, di cui 6 errate!)
  - Sensore tra 010 e 011: con codifica Gray G_2 può essere letto come ‘0’ o ‘1’, G_1 solo come ‘1’, G_0 solo come ‘0’

→ **Codifica Gray permette di ottenere due posizioni, entrambe corrette**!

Codice Gray e distanza di Hamming

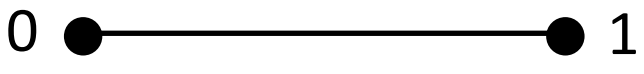
- Definiamo distanza di Hamming il **numero di bit per cui le rappresentazioni di due numeri differiscono tra loro**

- Il **codice Gray** è un codice a **distanza di Hamming unitaria**, cioè due numeri consecutivi differiscono di un solo (a differenza del codice binario usuale)
- Il codice Gray è un **codice n-chiuso** (il primo e l'ultimo numero del codice hanno distanza unitaria)

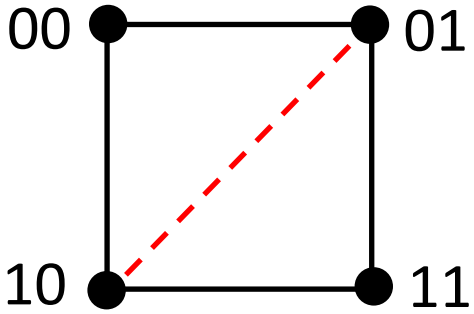
Gray Code	Bit Changes
000	
001	1
011	1
010	1
110	1
111	1
101	1
100	1
000	1

Distanza di Hamming

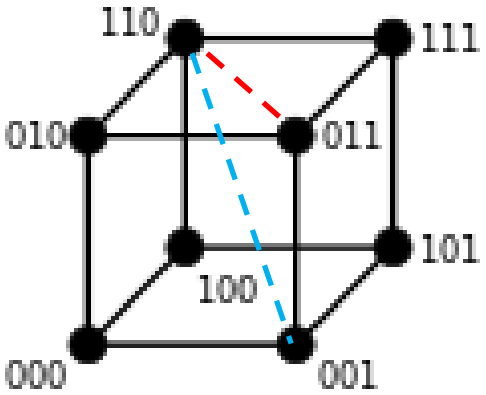
- **Rappresentazione geometrica**: numeri adiacenti sono congiunti da una linea continua



1 bit (unidimensionale)



2 bit (bidimensionale)



distanza 1
distanza 2
distanza 3

3 bit (tridimensionale)

Costruzione di un codice Gray a n bit

- Un codice Gray si può costruire in modo **ricorsivo**
- Per costruire un codice Gray da n parole, si parte dalla sequenza binaria di n codici (con **n pari**)
  - I primi n/2 codici sono costituiti da ‘0’ seguito dalla parità tra ogni bit del codice binario e il bit alla sua sinistra
  - Gli n/2 codici rimanenti sono costituiti dalla sequenza di numeri trovati al punto precedente, disposti in ordine inverso (specchiati) e con il bit più significativo posto a ‘1’
- Per sequenze di codici binari di 2n – 1 parole (numero **dispari** di parole), il corrispondente in codice Gray si ottiene copiando il bit più significativo e sostituendo ogni bit rimanente con la parità tra ciascun bit e il bit alla sua sinistra

Esempio: codice Gray (con n pari)

- Costruire il codice Gray da 10 parole

Binario	Gray
0000	
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	

I primi n/2 codici sono costituiti da '0' seguito dalla parità tra ogni bit del codice binario e il bit alla sua sinistra

I codici rimanenti si trovano dalla sequenza trovata sopra, specchiati e con il bit più significativo posto a '1'

Esempio: codice Gray (con n pari)

- Costruire il codice Gray da 10 parole

Binario	Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	1110
0110	1010
0111	1011
1000	1001
1001	1000

I primi n/2 codici sono costituiti da '0' seguito dalla parità tra ogni bit del codice binario e il bit alla sua sinistra

I codici rimanenti si trovano dalla sequenza trovata sopra, specchiati e con il bit più significativo posto a '1'

Esempio: codice Gray (con n dispari)

- Costruire il codice Gray da 7 parole (3 bit)

Binario	Gray
000	000
001	
010	
011	
100	
101	
110	

Si copia il bit più significativo e si sostituisce ogni bit rimanente con la parità tra ciascun bit e il bit alla sua sinistra

Esempio: codice Gray (con n dispari)

- Costruire il codice Gray da 7 parole (3 bit)

Binario	Gray
000	000
001	001
010	011
011	010
100	110
101	111
110	101

Si copia il bit più significativo e si sostituisce ogni bit rimanente con la parità tra ciascun bit e il bit alla sua sinistra

Esercizi

a) Convertire da binario a decimale: 1011001
b) Convertire da decimale a binario: 3871
c) Convertire da esadecimale a binario: E429
d) Convertire da decimale a ottale: 347
e) Eseguire la somma tra numeri binari: 10001 + 1111
f) Eseguire la differenza tra numeri binari: 10001 – 1111
g) Eseguire la moltiplicazione tra numeri binari: 10001 x 101
h) Rappresentare i numeri decimali 715 e 354 in BCD
i) Trovare il codice di parità pari per il seguente numero: 10001
j) Trovare il codice di parità dispari per il seguente numero: 10001
k) Costruire il codice Gray da 16 parole

Disclaimer

Figures from *Logic and Computer Design Fundamentals*, Fifth Edition, GE Mano | Kime | Martin