

Sławomir Kulesza

# Technika cyfrowa

## Wprowadzenie

Wykład dla studentów III roku Informatyki

Wer. 6.0, 01/10/2016

# Organizacja zajęć

Wykład:

2h × 15 tyg.

**Zaliczenie**

Pracownia:

2h × 10 tyg.

Ocena

Materiały:

[wmii.uwm.edu.pl/~kulesza](http://wmii.uwm.edu.pl/~kulesza) →  
→ Dydaktyka → Technika cyfrowa

Prowadzący: dr Sławomir Kulesza

[kulesza@matman.uwm.edu.pl](mailto:kulesza@matman.uwm.edu.pl)

p. D1/4

# Zaliczenie zajęć

- Kolokwia wejściowe – max 40 pkt
- Ocena końcowa:
  - 0-50 % → ndst
  - 51-60 % → dst
  - 61-70 % → dst+
  - 71-80 % → db
  - 81-90 % → db+
  - 91-100 % → bdb
- Jeśli nie – całościowe kolokwium

# Ramowy program wykładu

- Cyfrowy zapis informacji
- Technologie układów cyfrowych
- Algebra Boole'a
- Synteza układów kombinacyjnych
- Minimalizacja układów kombinacyjnych
- Arytmetyka binarna
- Kombinacyjne bloki funkcjonalne
- Synteza synchronicznych układów sekwencyjnych
- Synchroniczne automaty skończone

# Ramowy program ćwiczeń

Podstawowe funkcje logiczne

Algebra Boole'a

Prawa de Morgana

Bramka XOR

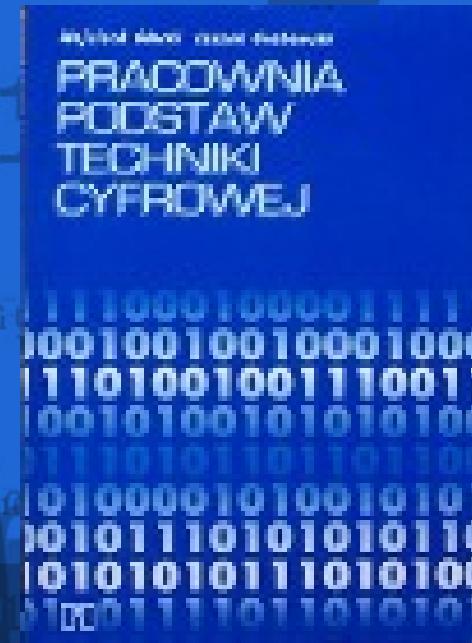
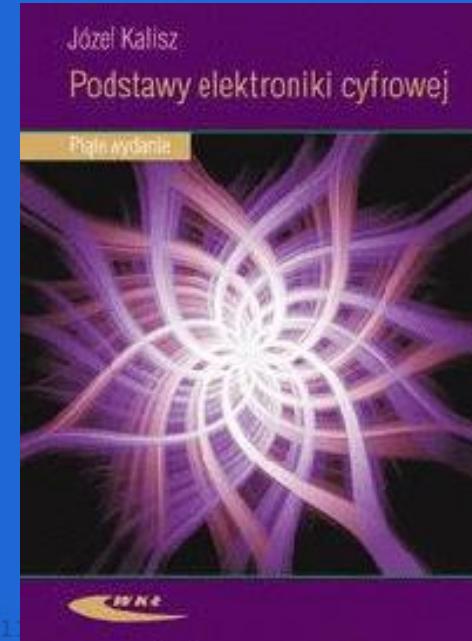
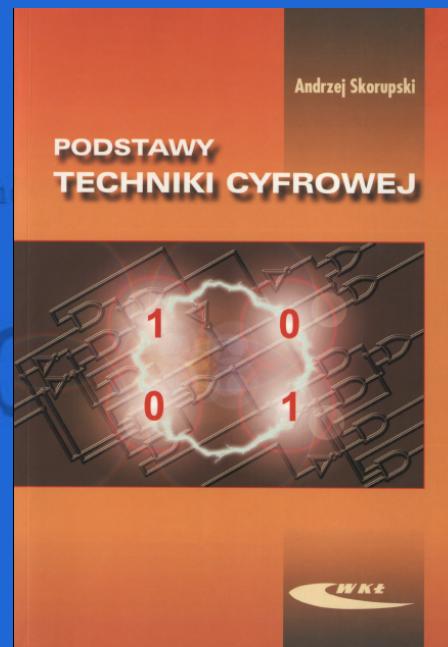
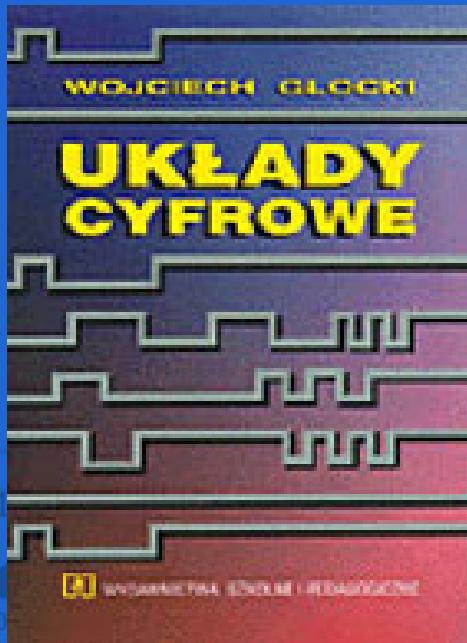
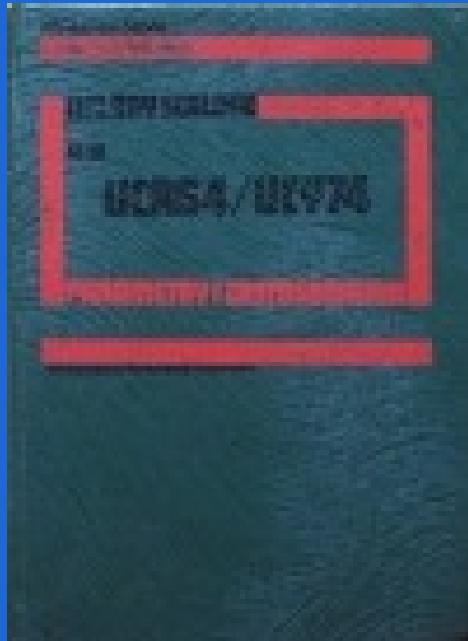
Konwertery kodów

Synteza układów kombinacyjnych

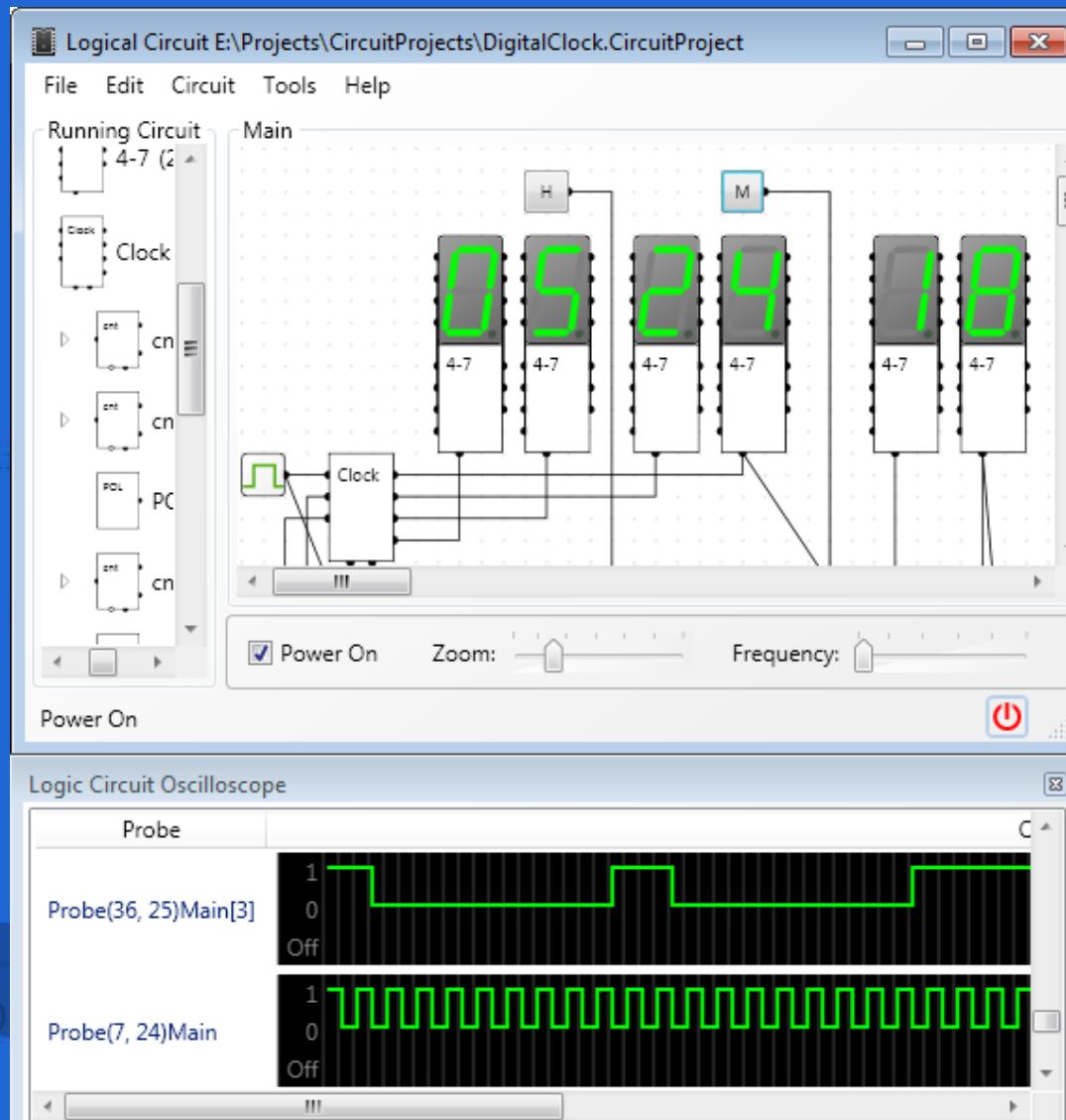
Przerzutniki  
Liczniki  
Rejestry

Synteza automatów skończonych

# Literatura



# Programy - LogicCircuit



[www.logiccircuit.org](http://www.logiccircuit.org)

# Programy – Logic Friday

Logic Friday

File Operation Truthtable Equation Gates View Help

Function Inputs Outputs True False DC PI Gates

F0	4	1	3	13	0	2	5
----	---	---	---	----	---	---	---

A B C D => F0  
Entered by truthtable:  
 $F0 = A' B' C' D' + A' B' C' D + A B' C D'$ ;  
Minimized:  
 $F0 = A' B' D' + A' B' C'$ ;

Inputs dialog box:

A = 0
B = 1
C = 0
D = 1
Trace
Cancel
Help

Ready

Logic Friday

File Operation Truthtable Equation Gates View Help

Function Inputs Outputs True False DC PI Gates

<none>

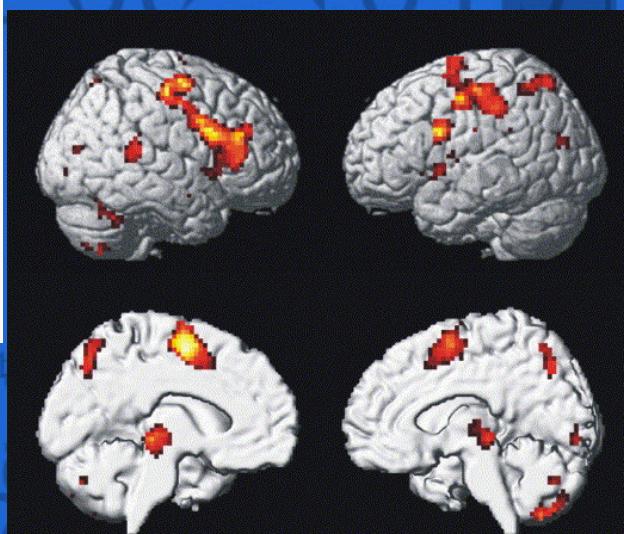
Term	A	B	C	D	=>	F0
0	0	0	0	0		0
1	0	0	0	1		0
2	0	0	1	0		0
3	0	0	1	1		0
4	0	1	0	0		0
5	0	1	0	1		1
6	0	1	1	0		1
7	0	1	1	1		0
8	1	0	0	0		0
9	1	0	0	1		X
10	1	0	1	0		0
11	1	0	1	1		0
12	1	1	0	0		0
13	1	1	0	1		0
14	1	1	1	0		0
15	1	1	1	1		0

Double-click an output cell to change state. Press <Enter> when done.

# Sygnały

Sygnały przenoszą информацию o stanie otaczającego nas świata:

- sygnał = nośnik + cechy + sens,
- sygnały wielkości fizycznych (mierzalnych),
- sygnały wielkości niefizycznych (niemierzalnych)



Sygnały  
świetlne

Sygnały  
magnetyczne

Sygnały  
elektryczne

# Nośniki sygnałów

Sygnały  
cieplne

Sygnały  
akustyczne

Sygnały  
mechaniczne

# Sygnały elektryczne

Skalowalność  
zastosowań

Szybkość  
przetwarzania

Uniwersalny  
format

Powszechnie  
występowanie

Wygoda  
przechowywania

Łatwość  
przesyłania i konwersji

Dostępność  
nośnika

# Parametry sygnałów elektrycznych

Częstotliwość

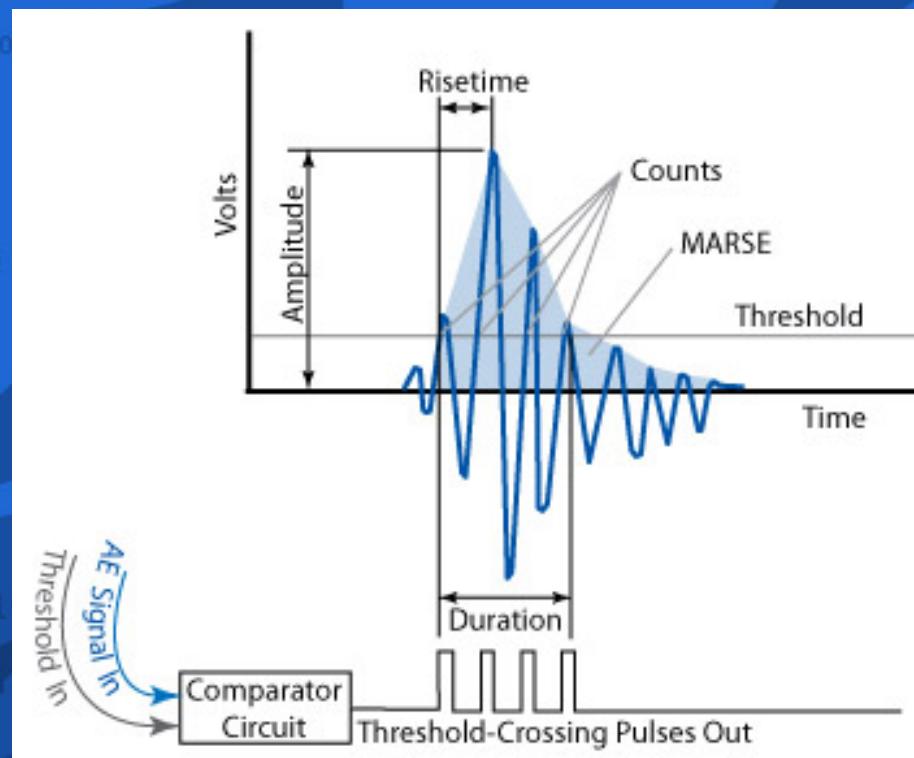
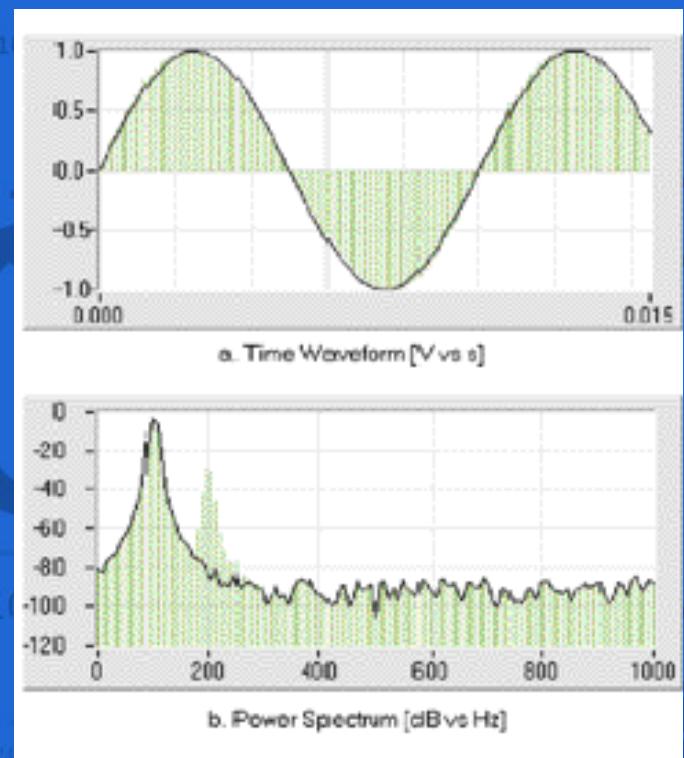
Amplituda

Energia/moc

Czas trwania

Napięcie/prąd  
Kształt obwiedni

Okres



# Analogowe sygnały elektryczne

Podatność na zakłócenia,  
trudne przechowywanie

Trudne projektowanie  
i przetwarzanie

Ograniczenia fizyczne

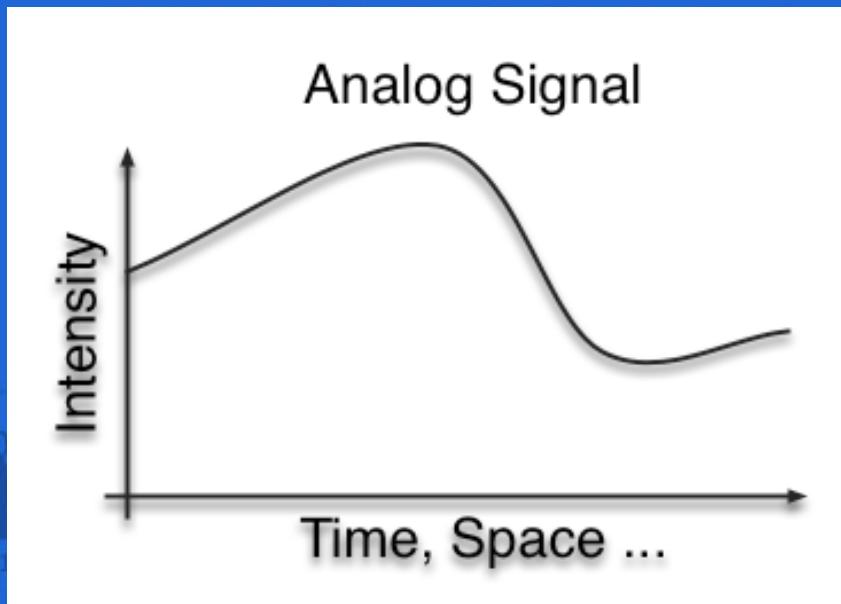
Redundancja

Sygnały ciągłe w czasie,  
przestrzeni i amplitudzie

Elektryczne postaci  
większości sygnałów  
pochodzących z otoczenia

Duża szybkość  
przetwarzania w czasie  
rzeczywistym

Łatwość przenoszenia,  
dostępność nośnika



# Cyfrowe sygnały elektryczne

Wydłużenie toru  
przetwarzania

Konwersja A/C, C/A

Konwersja stratna

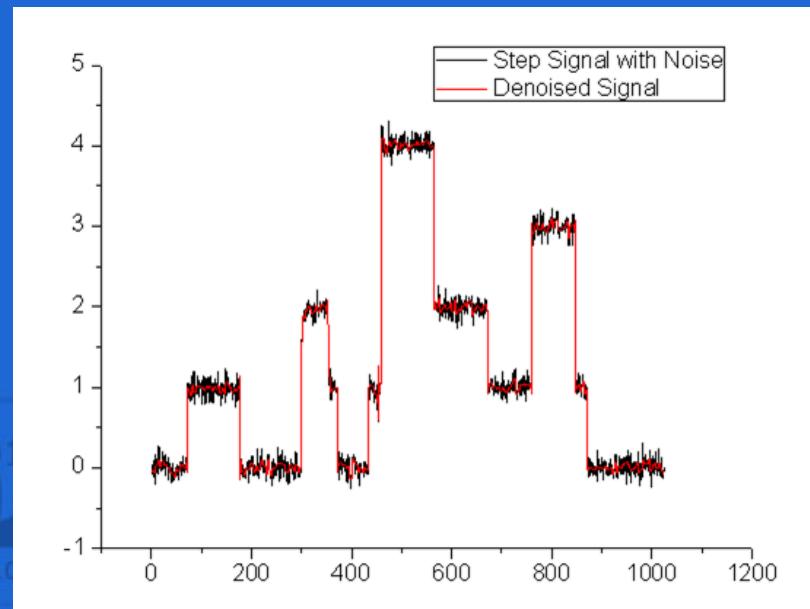
Niewielkie upakowanie

Sygnały nieciągłe  
w czasie, przestrzeni  
i amplitudzie

Sygnały łatwo  
obrabialne przez  
maszyny

Łatwość i elastyczność  
przetwarzania  
i przechowywania

Odporność na  
zakłócenia



# Sygnały binarne

Niewielka zwartość zapisu

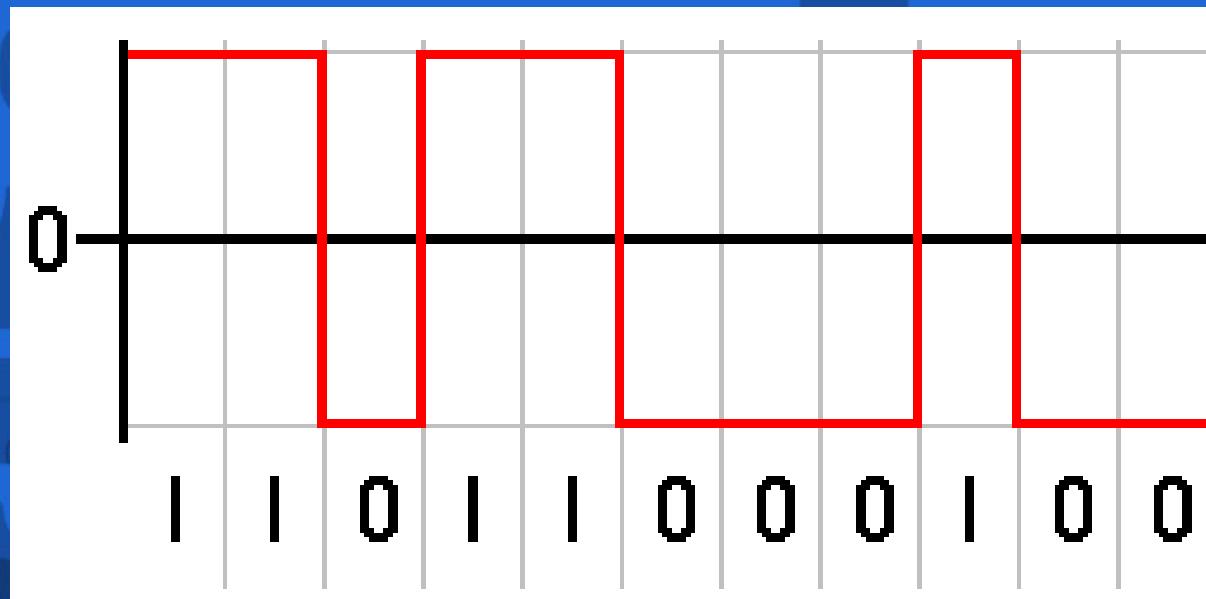
Jednoznaczność przekazu

Niejednoznaczność zapisu

Łatwość korekcji i odzyskiwania utraconej informacji

Prostota opisu i implementacji

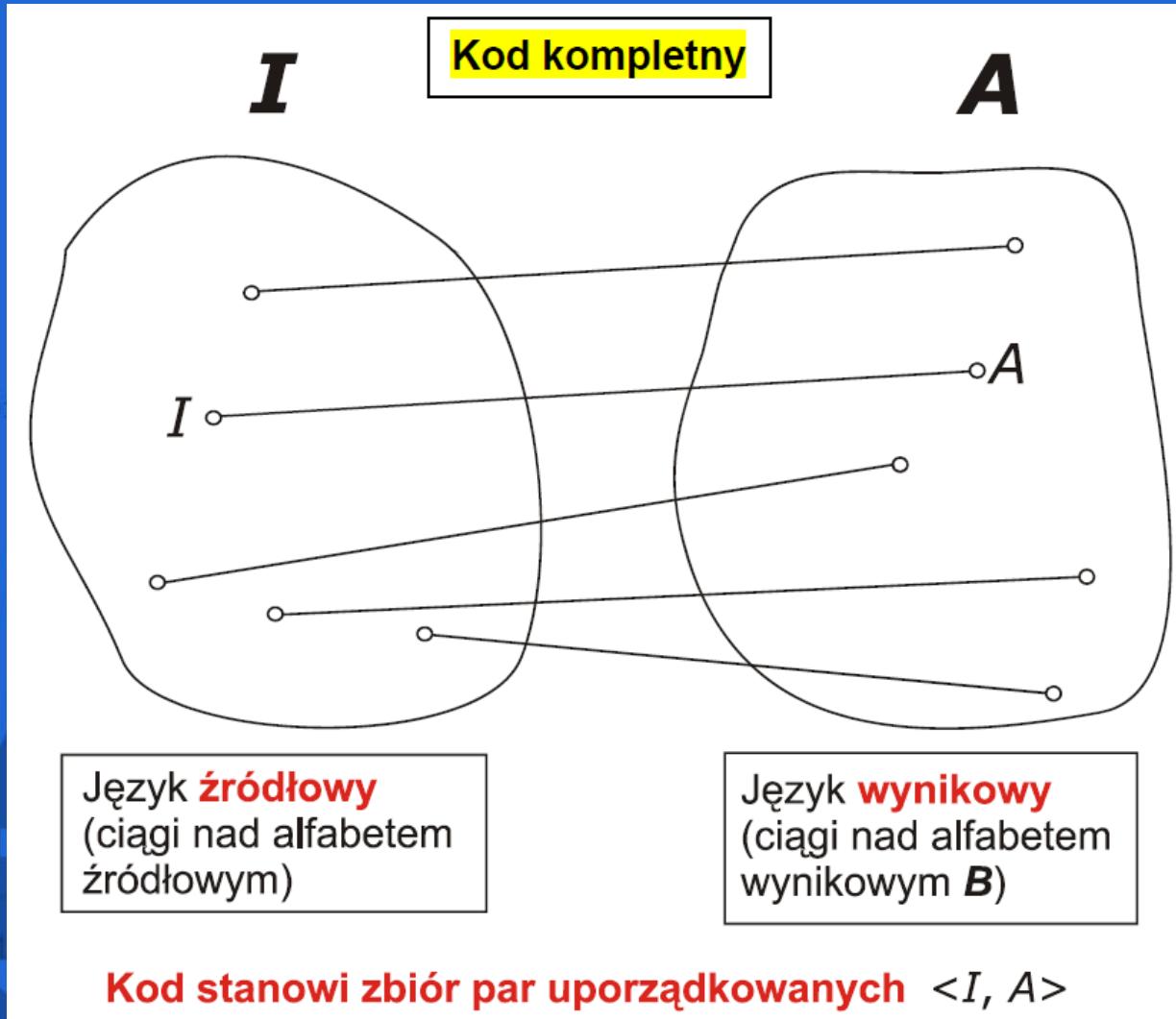
Kompresowalność



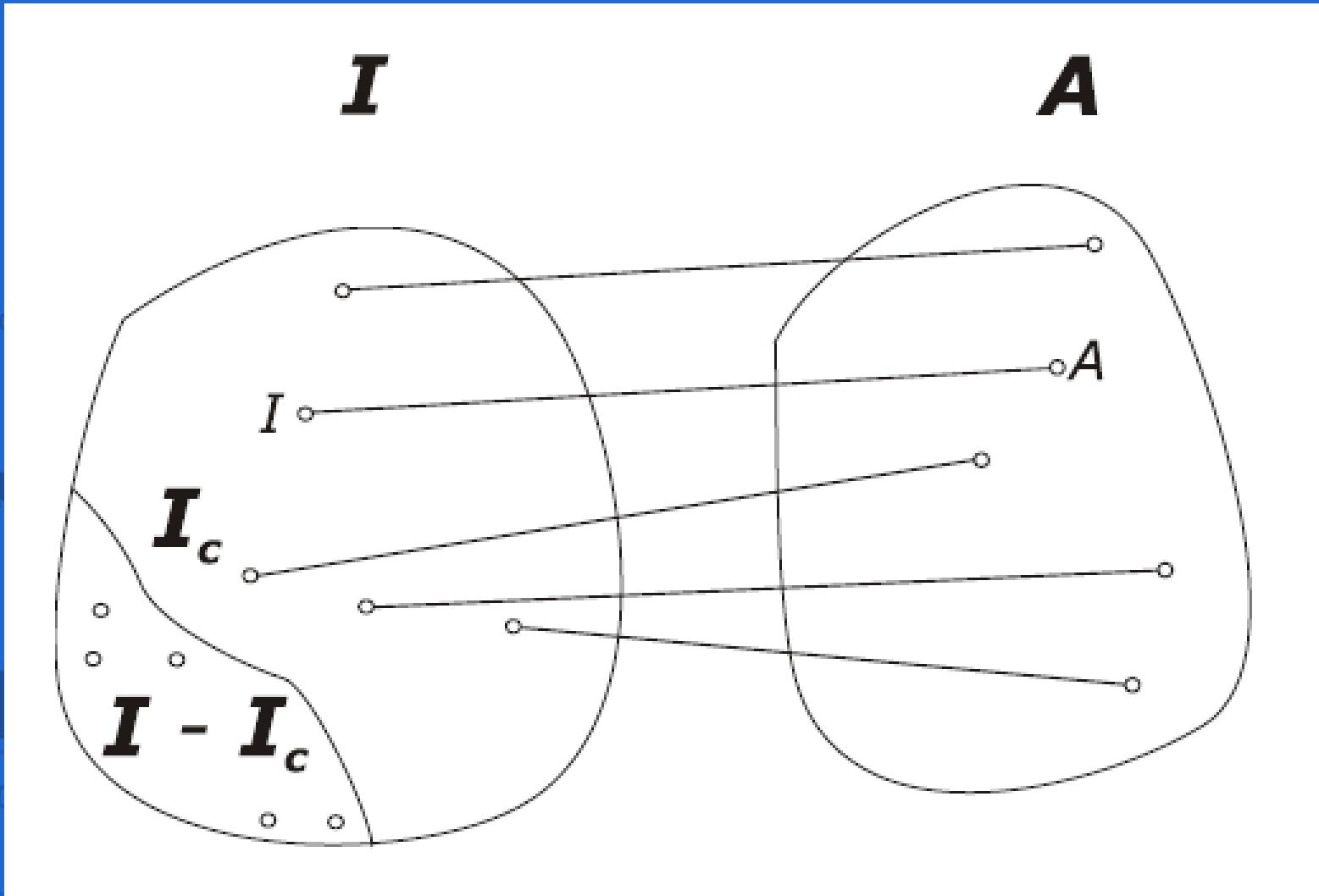
# Binarny zapis informacji

- Alfabet binarny: zbiór dwuznakowy, np.  $B = \{L,H\}$ ,  $\{0,1\}$
- Słowa: łańcuchy znakowe, np.  $B_i = 1000100101$
- Język: niepusty zbiór słów nad alfabetem,  
posiadających określone znaczenie (powiązanych  
z sygnałami)
- Długość słowa n: liczba znaków słowa, np.  $n(1010) = 4$
- Złożenie (konkatenacja):  $B_i \circ B_j$ , np.  $101 \circ 10 = 10110$
- Bit (binary unit): słowo jednoznakowe
- Tetrada (nibble): słowo czteroznakowe
- Bajt (byte): słowo ośmioznakowe

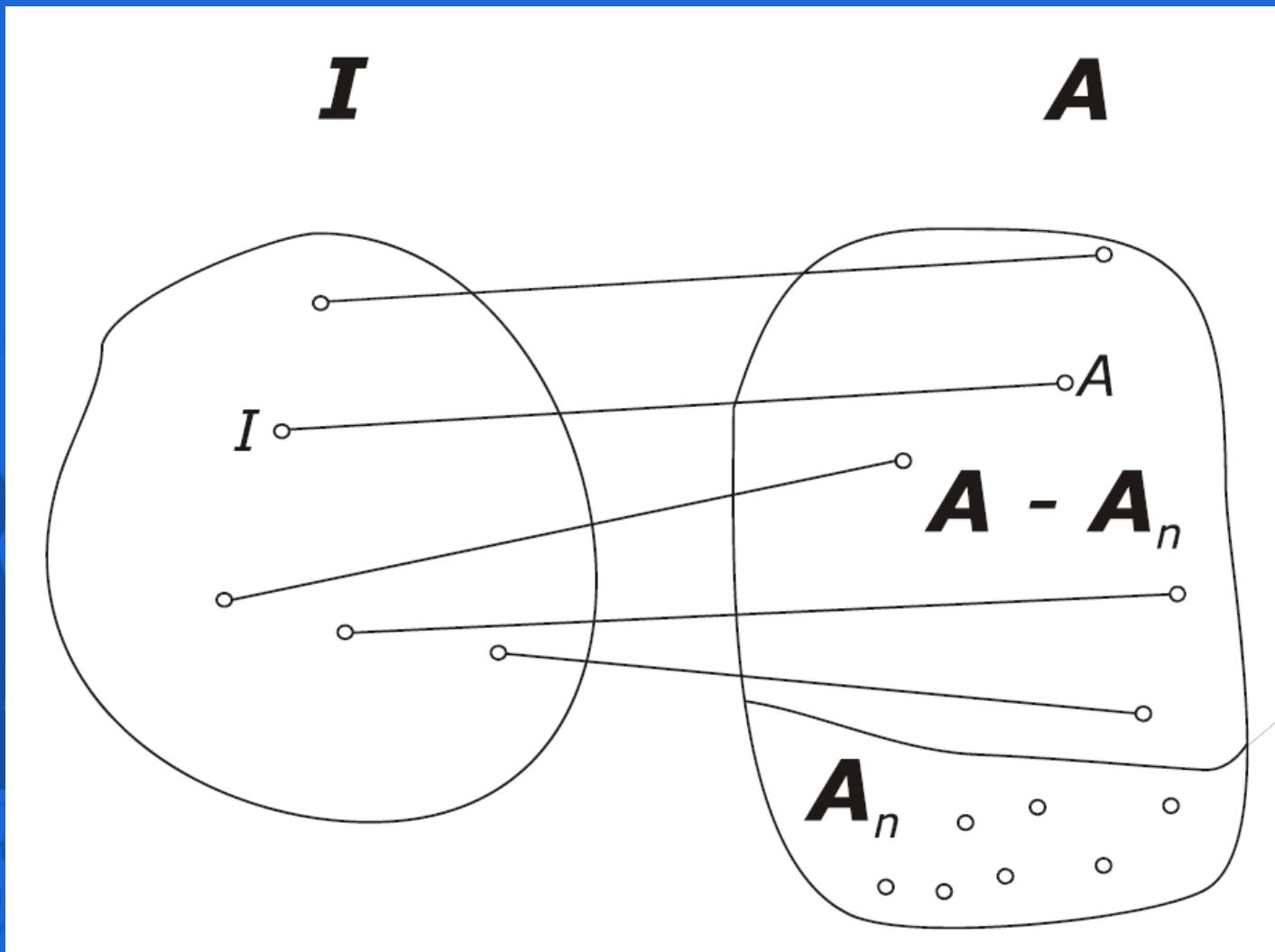
# Kodowanie informacji



# Kod niekompletny



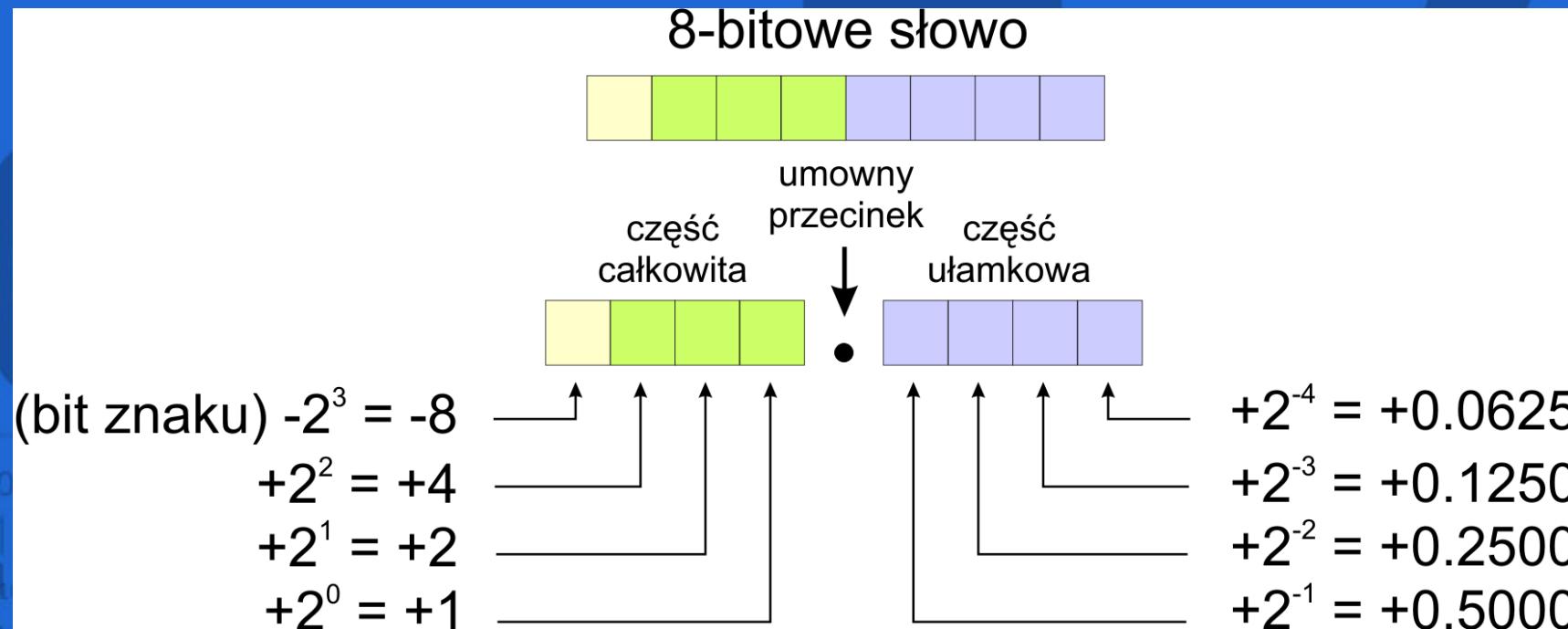
# Kod nadmiarowy



# Zapis stałoprzecinkowy

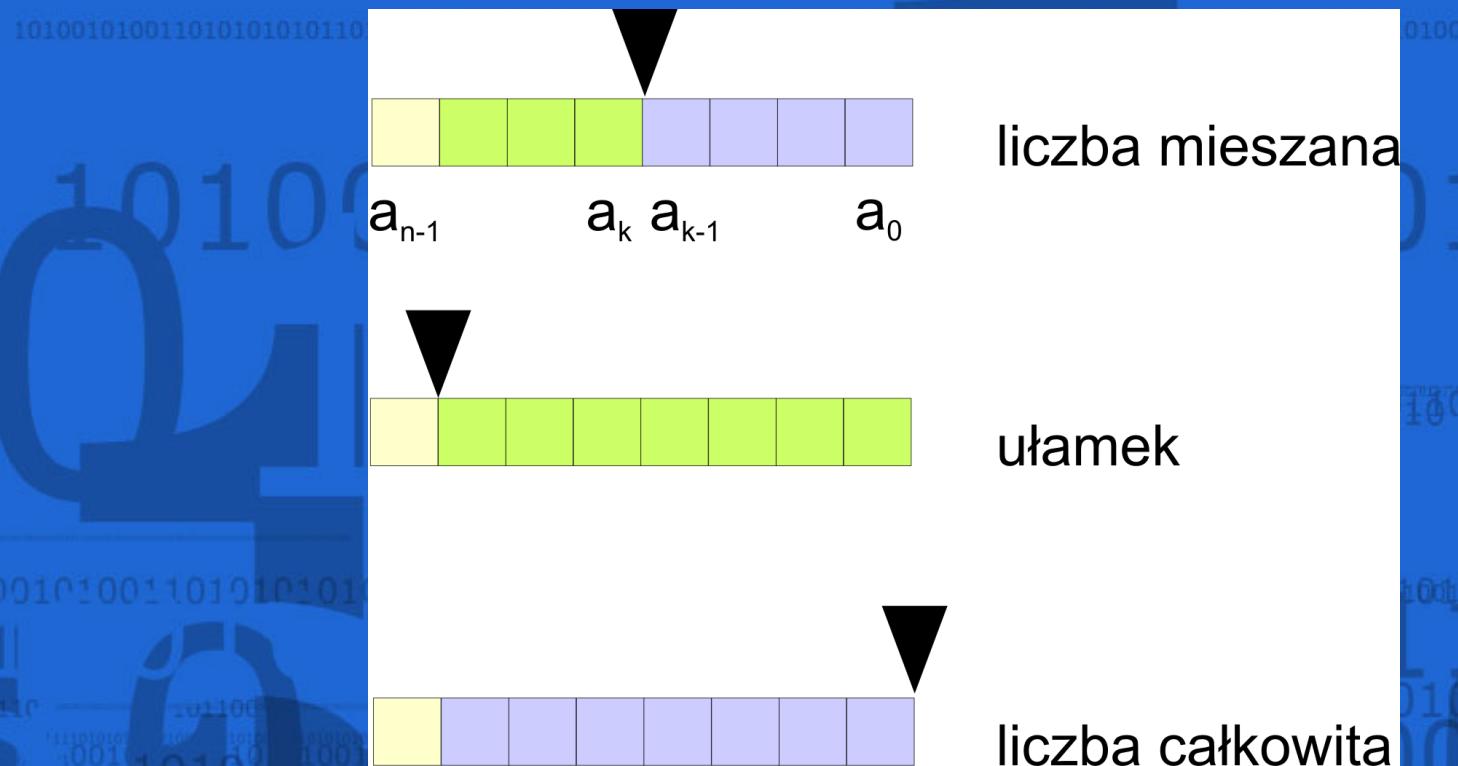
Liczby stałoprzecinkowe (stałopozycyjne, fixed-point numbers): z góry wiadomo, na której pozycji stoi umowny przecinek oddzielający część całkowitą i ułamkową danej liczby.

1010001010011010101011011000101101011110101001010010101001010010101001010100110010101001



# Liczby stałoprzecinkowe

Naczelną cechą zapisu stałopozyjnego jest stały skok liczb w całym zakresie ich zmienności. Wynosi on  $2^{-k}$ , gdzie k jest umownym położeniem przecinka.



# Kody naturalne

Nieujemna liczba rzeczywista  $A_i$  w kodzie naturalnym ma postać:

$$A_i = (a_{n-1} \dots a_1 a_0 \cdot a_{-1} \dots a_{-m})_p = \dots$$

$$\dots = \sum_{k=-m}^{n-1} a_k \cdot w_k = \sum_{k=-m}^{n-1} a_k \cdot p^k$$

Kody naturalne to kody pozycyjne, wagowe o stałej podstawie  $p$ .

Długość słowa:  $(m + n)$

$a_{n-1}$  – Most Significant Digit (MSD)

$a_{-m}$  – Least Significant Digit (LSD)

# Kody binarne

Słowo n-cyfrowe (bez znaku) może reprezentować maksymalnie  $2^n$  różnych liczb nieujemnych:

$$\|\{b_2 b_1 b_0\}\| = \|\{000, 001, \dots, 110, 111\}\| = 2^3 = 8$$

1010010100110101010101101100010110101111010101001010010101010010010101001010100110010101001

- najmniejsza liczba: 0
- największa liczba całkowita:  $B_{i \max} = 2^n - 1$
- największa liczba ułamkowa:  $B_{i \max} = 1 - 2^{-n}$

np.:

$$n = 5 \rightarrow B_{i \max} = 31, B_{i \max} = 0.96875$$

# Kody binarne

W słowie binarnym:

$$(b_{n-1} \dots b_0.b_{-1} \dots b_{-m})_2$$

- $b_{n-1}$  – Most Significant Bit (MSB),
- $b_{-m}$  – Least Significant Bit (LSB),
- minimalna ilość znaków do zakodowania k-słów pewnego alfabetu wynosi:

$$n_{\min} = \text{CEIL}(\log_2 k)$$

# Naturalny kod binarny NB

$$(b_{n-1} \dots b_0.b_{-1} \dots b_{-m})_2 = \dots$$

$$\dots = \sum_{k=-m}^{n-1} b_k \cdot 2^k$$

11

$$10001_2 = 1 \cdot 2^4 + 1 \cdot 2^0 = \dots$$

... = 17 10

1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	(1K)	10 0.000 976 562 5
2 048	(2K)	11 0.000 488 281 25
4 096	(4K)	12 0.000 244 140 625
8 192	(8K)	13 0.000 122 070 312 5
16 384	(16K)	14 0.000 061 035 156 25
32 768	(32K)	15 0.000 030 517 578 125
65 536	(64K)	16 0.000 015 258 789 062 5
131 072	(128K)	17 0.000 007 629 394 531 25
262 144	(256K)	18 0.000 003 814 697 265 625
524 288	(512K)	19 0.000 001 907 348 632 812 5
1 048 576	(1M)	20 0.000 000 953 674 316 406 25
2 097 152	(2M)	21 0.000 000 476 837 158 203 125
4 194 304	(4M)	22 0.000 000 238 418 579 101 562 5
8 388 608	(8M)	23 0.000 000 119 209 289 550 781 25
16 777 216	(16M)	24 0.000 000 059 604 644 775 390 625
33 554 432	(32M)	25 0.000 000 029 802 322 387 695 312 5
67 108 864	(64M)	26 0.000 000 014 901 161 193 847 656 25
134 217 728	(128M)	27 0.000 000 007 450 580 596 923 828 125
268 435 456	(256M)	28 0.000 000 003 725 290 298 461 914 062 5
536 870 912	(512M)	29 0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	(1G)	30 0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	(2G)	31 0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	(4G)	32 0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	(8G)	33 0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	(16G)	34 0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	(32G)	35 0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	(64G)	35 0.000 000 000 014 551 915 228 366 851 806 640 625

# Konwersja ND → NB

N-krokowy algorytm konwersji liczb całkowitych  
(N jest liczbą bitów w końcowym słowie binarnym)  
dzielenia mod 2:

np.:  $20_{10}$  dzieli się następująco:

$$20 \bmod 2 = 0 \quad \leftarrow \text{LSB}$$

$$10 \bmod 2 = 0$$

$$5 \bmod 2 = 1$$

$$2 \bmod 2 = 0$$

$$1 \bmod 2 = 1 \quad \leftarrow \text{MSB}$$

Stąd:  $20_{10} = 10100_2$

# Konwersja ND → NB

Konwersja ułamków dziesiętnych na postać binarną polega na cyklicznym mnożeniu przez 2.

np.:  $0.40625_{10}$

$$\begin{array}{r} 0, \quad 40625 \times 2 \quad \leftarrow \text{MSB} \\ 0 \quad 81250 \times 2 \\ 1 \quad 62500 \times 2 \\ 1 \quad 25000 \times 2 \\ 0 \quad 50000 \times 2 \\ 1 \quad 00000 \quad \leftarrow \text{LSB} \end{array}$$

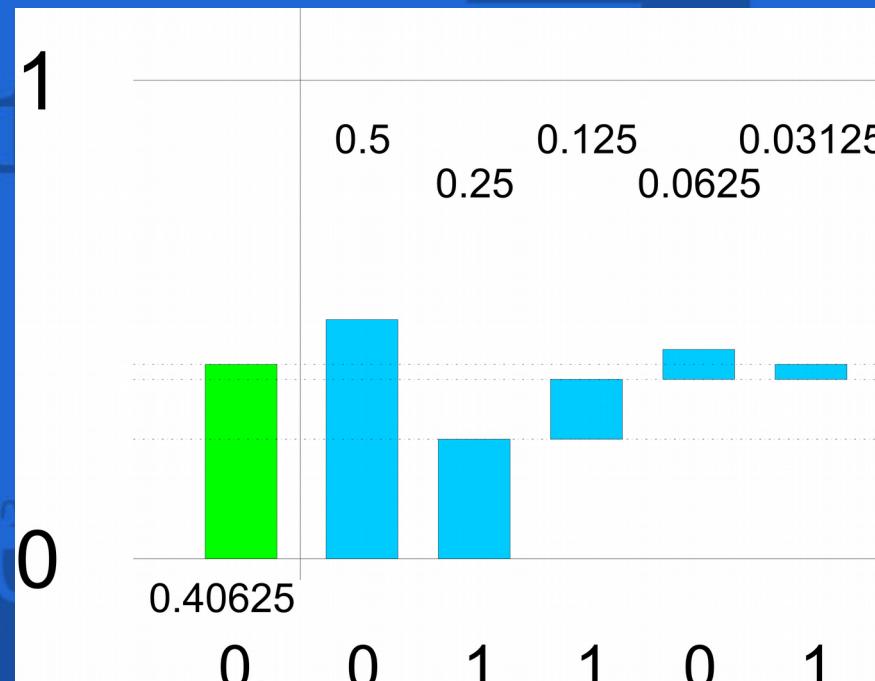
$$0.40625_{10} = 0.01101_2$$

# Konwersja ND → NB

Konwersja ułamków właściwych na postać binarną polega na dodawaniu składników o coraz mniejszych wagach  $2^{-n}$ , np.:

$$(1/3)_{10} = (1/4 + 1/12)_{10} = (1/4 + 1/16 + 1/48)_{10} = 0.0101(01)_2$$

$$(13/32)_{10} = (0.40625)_{10} = (1/4 + 1/8 + 1/32)_{10} = 0.01101_2$$



# Uwaga na ułamki!

Istnieją 'porządne' ułamki dziesiętne, które mają nieskończone rozwinięcie binarne:

$$\text{np.: } 0.1_{10} = 0.00011(00011)_2$$

1010010100110101010101101100010110101111010101001010010101010010010101001010100110010101001

Z drugiej strony istnieją także ułamki, których rozwinięcie dziesiętne jest nieskończone, a mimo to są one skończone w systemach niedziesiętnych:

$$\text{np.: } (1/3)_{10} = 0.3(3)_{10} = (3^{-1})_{10} = (0.1)_3$$

# Naturalny kod szesnastkowy HEX

$$\$(h_{n-1} \dots h_0.h_{-1} \dots h_{-m}) = \dots$$

$$\dots = \sum_{k=-m}^{n-1} h_k \cdot 16^k$$

np.:

$$\$10F = 10FH = 0x10F = \dots$$

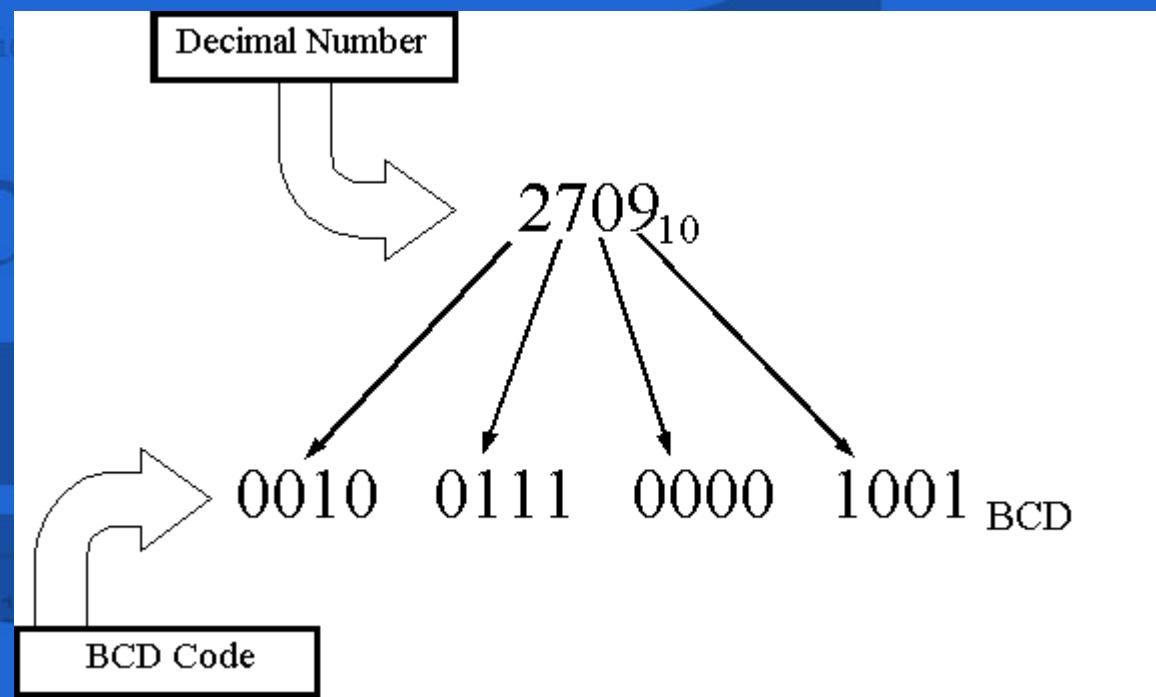
$$\dots = 1 \cdot 16^2 + 15 \cdot 16^0 = \dots$$

$$\dots = 0001\ 0000\ 1111_2 = 271_{10}$$

	BINARY	HEXADECIMAL	
$2^0$	0	0	$16^0$
	1	1	
$2^1$	10	2	
	11	3	
$2^2$	100	4	
	101	5	
	110	6	
	111	7	
$2^3$	1000	8	
	1001	9	
	1010	A	
	1011	B	
	1100	C	
	1101	D	
	1110	E	
	1111	F	
$2^4$	10000	10	$16^1$
	10001	11	
	10010	12	
	10011	13	
	10100	14	
	10101	15	
	10110	16	
	10111	17	
	11000	18	
	11001	19	
	11010	1A	
	11011	1B	
	11100	1C	

# Kody binarno-dziesiętne BCD

W kodach BCD (Binary-Coded Decimal) każda cyfra słowa dziesiętnego kodowana jest za pomocą liczby dwójkowej: łatwe w implementacji, ale rozrzutne (kody nadmiarowe)



# Kody binarno-dziesiętne BCD

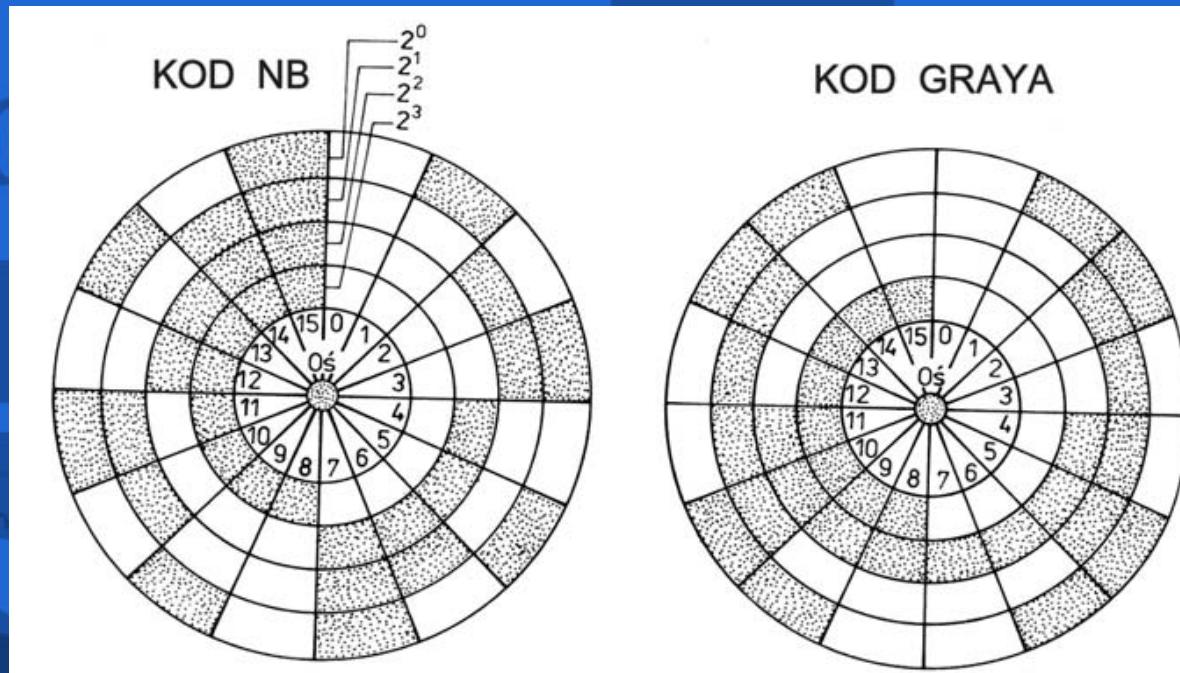
- **8421** (podzbiór 4-bitowego kodu NB), 4-bitowy
- **1-z-10**, 10-bitowy
- **wskaźnika 7-segmentowego**, 7-bitowy

<b>ND</b>	<b>8421</b>	<b>1-z-10</b>
0	0000	0000000001
1	0001	0000000010
2	0010	0000000100
3	0011	0000001000
4	0100	0000010000
5	0101	0000100000
6	0110	0001000000
7	0111	0010000000
8	1000	0100000000
9	1001	1000000000

# Kody refleksyjne (Graya)

W kodach refleksyjnych dwie kolejne liczby różnią się tylko zawartością jednej pozycji: gdy następuje zmiana bitu na najwyższej pozycji słowa kodowego, to tworzy się odbicie zwierciadlane (refleksja) słów z bitami o niższych pozycjach.

Praktyczne zastosowania kodu Graya: przetworniki przesuwu liniowego i kąta obrotu na dane cyfrowe, aby uniknąć dużych przejściowych błędów, możliwych przy kodzie NB.



# Kody refleksyjne (Graya)

Full-length Gray code (N=4, L <sub>g</sub> =16)	Symmetric Gray code (L <sub>g</sub> =6)	Non-continuous intermediate binary code	Full-length Binary Code	Binary Code (L = 6)	Decimal
---	--	---	-------------------------------	---------------------------	---------

0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0
0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1	1
0 0 1 1	0 0 1 1	0 0 1 0	0 0 1 0	0 0 1 0	2
0 0 1 0	1 0 1 1	1 1 0 1	0 0 1 1	0 0 1 1	3
0 1 1 0	1 0 0 1	1 1 1 0	0 1 0 0	0 1 0 0	4
0 1 1 1	1 0 0 0	1 1 1 1	0 1 0 1	0 1 0 1	5
0 1 0 1			0 1 1 0		6
0 1 0 0			0 1 1 1		7
1 1 0 0			1 0 0 0		8
1 1 0 1			1 0 0 1		9
1 1 1 1			1 0 1 0		10
1 1 1 0			1 0 1 1		11
1 0 1 0			1 1 0 0		12
1 0 1 1			1 1 0 1		13
1 0 0 1			1 1 1 0		14
1 0 0 0			1 1 1 1		15

(a)

(b)

(c)

(d)

(e)

# Porównanie kodów

Binary Base-2	Decimal Base-10	Hexa- decimal Base-16	Octal Base-8	BCD Code	Gray Code
0000	0	0	0	0	0000
0001	1	1	1	1	0001
0010	2	2	2	2	0011
0011	3	3	3	3	0010
0100	4	4	4	4	0110
0101	5	5	5	5	0111
0110	6	6	6	6	0101
0111	7	7	7	7	0100
1000	8	8	10	8	1100
1001	9	9	11	9	1101
1010	10	A	12	---	1111
1011	11	B	13	---	1110
1100	12	C	14	---	1010
1101	13	D	15	---	1011
1110	14	E	16	---	1001
1111	15	F	17	---	1000

# Kodowanie liczb binarnych ze znakiem

Kod znak-moduł (ZM):

W tym kodzie bit MSB definiuje znak liczby: 0 = '+', 1 = '-', pozostałe bity kodowane jak w kodzie naturalnym:

np.:  $00001_{ZM} = +1_{10}$ ,  $10001_{ZM} = -1_{10}$

Wady - niejednoznaczność określenia zera (tzw. zero dodatnie i ujemne):  $1000000_{ZM} = 0000000_{ZM}$ , odmiennie realizowane dodawanie i odejmowanie (przeniesienie i pożyczka).

# Kodowanie liczb binarnych ze znakiem

Kod znak - uzupełnienie do 2 (ZU2):

liczba dodatnia:  $A_{ZU2} = A_{ZM}$ , przy czym MSB = 0.

liczba ujemna: MSB = 1, pozostałe bity powstają przez zanegowanie odpowiednich bitów kodowanych naturalnie i dodanie 1.

np.:  $+13_{10} = 01101_{ZU2}$     $-13_{10} = 10011_{ZU2}$

Zalety – jednoznacznie określone zero: 000000ZU2, stąd większy zakres kodowanych liczb.

Wada – zakres zmienności liczb jest niesymetryczny,  
np.: dla  $n = 4$ ,  $A = \{-8, -7, \dots, 7\}$ .

# Porównanie kodów ze znakiem

## Przykłady liczb dwójkowych ze znakiem

Liczba dziesiętna	ZM	ZU1	ZU2
-8 (-1)			1.000
-7 (-0.875)	1.111	1.000	1.001
-6 (-0.75)	1.110	1.001	1.010
-5 (-0.625)	1.101	1.010	1.011
-4 (-0.5)	1.100	1.011	1.100
-3 (-0.375)	1.011	1.100	1.101
-2 (-0.25)	1.010	1.101	1.110
-1 (-0.125)	1.001	1.110	1.111
-0	1.000	1.111	
0			0.000
+0	0.000	0.000	
1 (0.125)	0.001	0.001	0.001
...	...	...	...
7 (0.875)	0.111	0.111	0.111

# Kod alfanumeryczny ASCII

Siedmiobitowy kod alfanumeryczny ASCII  
ND – liczba dziesiętna, HEX – liczba szesnastkowa

ND	HEX	Znak	ND	HEX	Znak	ND	HEX	Znak	ND	HEX	Znak
0	00		32	20	SP	64	40	@	96	60	`
1	01	☺	33	21	!	65	41	A	97	61	a
2	02	☻	34	22	"	66	42	B	98	62	b
3	03	♥	35	23	#	67	43	C	99	63	c
4	04	♦	36	24	\$	68	44	D	100	64	d
5	05	♣	37	25	%	69	45	E	101	65	e
6	06	♠	38	26	&	70	46	F	102	66	f
7	07	●	39	27	,	71	47	G	103	67	g
8	08	■	40	28	(	72	48	H	104	68	h
9	09	○	41	29	)	73	49	I	105	69	i
10	0A	▣	42	2A	*	74	4A	J	106	6A	j
11	0B	♂	43	2B	+	75	4B	K	107	6B	k
12	0C	♀	44	2C	,	76	4C	L	108	6C	l
13	0D	♪	45	2D	-	77	4D	M	109	6D	m
14	0E	♫	46	2E	.	78	4E	N	110	6E	n
15	0F	¤	47	2F	/	79	4F	O	111	6F	o
16	10	▶	48	30	0	80	50	P	112	70	p
17	11	◀	49	31	1	81	51	Q	113	71	q
18	12	↑	50	32	2	82	52	R	114	72	r
19	13	!!	51	33	3	83	53	S	115	73	s
20	14	¶	52	34	4	84	54	T	116	74	t
21	15	§	53	35	5	85	55	U	117	75	u
22	16	▬	54	36	6	86	56	V	118	76	v
23	17	▬	55	37	7	87	57	W	119	77	w
24	18	↑	56	38	8	88	58	X	120	78	x
25	19	↓	57	39	9	89	59	Y	121	79	y
26	1A	→	58	3A	:	90	5A	Z	122	7A	z
27	1B	←	59	3B	;	91	5B	[	123	7B	{
28	1C	↶	60	3C	<	92	5C	\	124	7C	
29	1D	↷	61	3D	=	93	5D	]	125	7D	}
30	1E	▲	62	3E	>	94	5E	^	126	7E	~
31	1F	▼	63	3F	?	95	5F	-	127	7F	DEL

# Rozszerzony ASCII – strony kodowe

POLSKIE LITERY W NAJCZĘŚCIEJ STOSOWANYCH KODACH

Oznaczenia: ND – liczba dziesiętna, EN – IBM English (*Code page 437*),  
 L2 – IBM Latin-2 (*Code page 852*), EE – IBM Windows EE (*Code page 1250*),  
 ISO – ISO Latin-2 (*ISO-8859-2*)

ND	EN	L2	EE	ISO	ND	EN	L2	EE	ISO	ND	EN	L2	EE	ISO
128	Ç	Ç			160	á	á			192	ł	ł		
129	ü	ü			161	í	í	A		193	ł	ł		
130	é	é			162	ó	ó			194	ł	ł		
131	â	â			163	ú	ú	Ł	Ł	195	ł	ł		
132	ä	ä			164	ñ	ñ	A		196	—	—		
133	à	ú			165	Ñ	ñ	A		197	+	+		
134	å	é			166	²	Z	A	Ś	198	ƒ	Ã	Ć	Ć
135	ç	ç			167	²	ž			199	¶	ă		
136	ê	ł			168	ż	E			200	£	£		
137	ë	ë			169	¬	€			201	¤	¤		
138	è	Ó			170	¬	¬			202	£	£	E	E
139	í	ő			171	½	ž			203	—	—	E	E
140	î	í	S		172	¼	Č		Ž	204	ƒ	ƒ		
141	ì	Ž			173	i	š			205	=	=		
142	Ä	Ä			174	“	“			206	±	±		
143	Å	Ć	Ž		175	»	»	Ž	Ž	207	±	¤		
144	É	É			176	...	...			208	„	đ		
145	æ	Ł			177	...	...			209	—	đ	N	N
146	Æ	Í			178	...	...			210	—	đ		
147	ô	ô			179			I	I	211	£	ë	Ó	Ó
148	ö	ö			180					212	£	đ		
149	ò	L			181		A			213	£	ň		
150	û	ł			182		À		ś	214	£	í		
151	ù	Ś			183		È			215	+	í		
152	ij	ś			184		Ş			216	+	ě		
153	Ö	Ö			185		ä			217	—	—		
154	Ü	Ü			186					218	—	—		
155	c	ł	s		187					219	—	—		
156	£	t			188					220	—	—		
157	¥	Ł			189	—	Ż			221	—	—	T	T
158	R	X			190	—	ż			222	—	—	Ú	Ú
159	f	č	z		191	—	—	ż	ż	223	—	—		

# UNICODE

Standard zapisu znaków przy pomocy liczb 2-bajtowych (65536 słów). Pozwala na zdefiniowanie znaków diakrytycznych wszystkich znanych języków świata (łącznie z arabskim, odmianami chińskiego i in.)

10100101001101010101101100010110101111010100101001010100101010010101001010100110010101001

Odmiany: UTF-16, UTF-8, UTF-7 i in.

UTF-8 jest schematem kodowania znaków Unicode o zmiennej długości łańcucha, zgodnym ze standardem ASCII.

# Schemat kodowania UTF-8

1 bajt: 0xxxxxx;

2 bajty: 110xxxxx 10xxxxxx;

3 bajty: 1110xxxx 10xxxxxx 10xxxxxx;

4 bajty: 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx;

(Ex.)

Kodowanie znaku K (U+004B)

Zakres 0000-008F (ASCII)

004B h = 0100 1011 bin

UTF-8: 01001011

Kodowanie znaku ♂ (U+2642):

Zakres 0800-FFFF (3 bajty).

2642 h = 0010 0110 0100 0010<sub>2</sub>

UTF-8: 11100010 10011001 10000010

# Kody detekcyjne i kontrolne

Zakłócenia sygnału wymuszają stosowanie kodów kontrolnych:

- detekcyjnych – wykrywających błędy transmisji,
- kontrolnych – wykrywających i korygujących błędy transmisji.



# Kontrola parzystości

Najprostszy kod detekcyjny wykorzystuje transmisję sumy kontrolnej – bitu parzystości:

Binarny								Hex	Bit parzystości	
1	0	0	0	0	0	1	0	1	0x85	1
0	0	0	0	0	0	1	1	1	0x07	1
1	0	1	0	0	0	0	1	1	0xA3	0
1	0	0	1	0	1	0	1	1	0x95	0
0	0	0	0	0	0	1	0	1	0x05	0
1	0	1	0	0	0	0	1	1	0xA3	0
0	0	0	1	0	1	0	1	1	0x15	1
0	0	1	0	1	0	0	1	1	0x29	1
								0xC3		

Cechy:

- kod nadmiarowy,
- niewrażliwy na parzystą liczbę błędów transmisji.