Circuitos Digitais

Códigos Binários

Prof. Marcelo Grandi Mandelli

mgmandelli@unb.br

Códigos Binários

- Qualquer informação no computador é representada por códigos binários:
  - caracteres, números, símbolos, etc.

- Existem diversas alternativas para codificar elementos dependendo das características desejadas.

- Um código pode ser otimizado para
  - reduzir espaço de armazenamento necessário
  - representar informações de forma unívoca
  - ainda explorar redundâncias para deteção e correção de erros

Códigos Binários

- ASCII e Unicode
- Código BCD
- Código de Gray
- Códigos k-de-n
- Códigos de paridade
- Códigos de Hamming

ASCII e Unicode

- American Standard Code for Information Interchange (Código Padrão Americano para Troca de Informações)

- Padrão americano de codificação de caracteres proposto em 1963

- O código ASCII oficial usa 7 bits, o que permite 2^7 = 128 combinações:
  - 33 representam caracteres de controle como Line Feed ou Carriage Return, muito úteis, especialmente nos tempos em que teletipos – máquinas de escrever comandadas por computador – eram um periférico essencial para os operadores.
  - 95 caracteres “imprimíveis” do alfabeto Inglês americano

ASCII e Unicode

CARACTERES DE CONTROLE				SÍMBOLO GRÁFICO											
NOME	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA
NUL	0	0000000	00	espaço	32	0100000	20	@	64	1000000	40	`	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	−	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

ASCII e Unicode

- Não contemplava outras línguas → ASCII estendido para 8 bits (um byte) para representar 256 (2^8) caracteres → BRASCII no Brasil

- Unicode foi criado para suceder o ASCII, sendo um padrão de codificação capaz de representar caracteres de praticamente qualquer conjunto de caracteres das línguas vivas e mortas do mundo.

- Unicode permite diversas codificações em bits. A codificação mais importante – a mais utilizada na Internet – é a UTF-8, que utiliza entre 1 e 4 bytes para a representação de caracteres. Existe ainda UTF-16 e UTF-32.

ASCII e Unicode

- O Unicode contém o cojunto ASCII. Caracteres representados por um único byte têm a mesma representação em UTF-8 e em ASCII

Smileys & Emotion

face-smiling

№	Code	Browser	Appl	Goog	FB	Wind	Twtr	Joy	Sams	GMail	SB	DCM	KDDI	CLDR Short Name
1	U+1F600										—	—	—	grinning face
2	U+1F603													grinning face with big eyes
3	U+1F604											—	—	grinning face with smiling eyes
4	U+1F601													beaming face with smiling eyes
5	U+1F606										—		—	grinning squinting face

https://unicode.org/emoji/charts/full-emoji-list.html

Códigos BCD (Binary Coded Decimal)

- Associam os 10 algarismos decimais (0 a 9) a códigos de 4 bits (16 combinações possíveis)
- Diversas associações são utilizadas, com predominância da representação BCD natural
- Na tabela apresentada a seguir, os pesos associados a cada um dos quatro dígitos binários aparecem entre parênteses

Códigos BCD (Binary Coded Decimal)

Mais utilizado

Dígito	BCD natural (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

BCD Natural é igual ao código binário até o valor 9. Os valores entre 10 e 15 não são válidos.

Códigos BCD (Binary Coded Decimal)

Dígito	BCD natural (8	4	2	1)	Aiken (2	4	2	1)
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	1	0	0
7	0	1	1	1	1	1	0	1
8	1	0	0	0	1	1	1	0
9	1	0	0	1	1	1	1	1

Aiken é igual ao
código binário até o valor 4.
Os valores 5 a 9 são formados
Pela inversão dos bits dos
valores 4 a 0.

Códigos BCD (Binary Coded Decimal)

Excesso-de-três: simplifica a aritmética BCD

Dígito	BCD natural (8 4 2 1)	Aiken (2 4 2 1)	Stibitz (8 4 2 1) - 3
0	0 0 0 0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 0 1 0	0 1 0 1
3	0 0 1 1	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 1 1	1 0 0 0
6	0 1 1 0	1 1 0 0	1 0 0 1
7	0 1 1 1	1 1 0 1	1 0 1 0
8	1 0 0 0	1 1 1 0	1 0 1 1
9	1 0 0 1	1 1 1 1	1 1 0 0

Para converter pro dígito decimal: Stibitz - 3.

Códigos BCD (Binary Coded Decimal)

Excesso-de-três: simplifica a aritmética BCD

Dígito	BCD natural (8 4 2 1)	Aiken (2 4 2 1)	Stibitz (8 4 2 1) - 3	(7 4 2 1)	(6 4 2 -1)
0	0 0 0 0	0 0 0 0	0 0 1 1	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1	0 1 0 0	0 0 0 1	0 0 1 1
2	0 0 1 0	0 0 1 0	0 1 0 1	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1	0 1 1 0	0 0 1 1	0 1 0 1
4	0 1 0 0	0 1 0 0	0 1 1 1	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1	1 0 0 0	0 1 0 1	0 1 1 1
6	0 1 1 0	1 1 0 0	1 0 0 1	0 1 1 0	1 0 0 0
7	0 1 1 1	1 1 0 1	1 0 1 0	0 1 1 1	1 0 1 1
8	1 0 0 0	1 1 1 0	1 0 1 1	1 0 0 1	1 0 1 0
9	1 0 0 1	1 1 1 1	1 1 0 0	1 0 1 0	1 1 0 1

BCD Natural - Números > 9

- Para escrever números decimais > 9 em BCD, escreva o valor BCD correspondente para cada dígito
  - Exemplo

157

0001 0101 0111

Decimal	Binário			
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

Códigos BCD (Binary Coded Decimal)

- Algoritmo para soma de números BCD
  1. Efetuar a soma binária convencional dos dois números
  2. Adicionar 6 a cada nibble (grupo de 4 bits) que não seja um valor BCD válido
  3. Repetir o passo 2 até que todos os nibbles do resultado correspondam a valores BCD válidos

Códigos BCD (Binary Coded Decimal)

- Exemplo de soma de números BCD

7 4
2 9 +
1 0 3

0 1 1 1 0 1 0 0
0 0 1 0 1 0 0 1 +
1 0 0 1 $\boxed{1\ 1\ 0\ 1}$ ← *Nibble* não é BCD!
0 1 1 0 + ← Some 6 ao *nibble*

Nibble não é BCD! → $\boxed{1\ 0\ 1\ 0}$ 0 0 1 1
Some 6 ao *nibble* → 0 1 1 0 +

$1\ \underbrace{0\ 0\ 0\ 0}_{0}\ \underbrace{0\ 0\ 1\ 1}_{3}$

Códigos BCD (Binary Coded Decimal)

- Exemplo de soma de números BCD

$$
\begin{array}{rrr}
 & 7 & 4 \\
 & 8 & 8 \; + \\
\hline
1 & 6 & 2
\end{array}
$$

$$
\begin{array}{c|cccc|cccc|l}
 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & \\
 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & + \\
\hline
 & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{0} & \boxed{0} & \\
 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & + \\
\hline
1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & \\
1 & & & 6 & & & & 2 & &
\end{array}
$$

Nibble não é BCD! → 1 1 1 1

1 1 0 0 ← *Nibble* não é BCD!

Some 6 ao *nibble* → 0 1 1 0

0 1 1 0 ← Some 6 ao *nibble*

Código de Gray

- É um código numérico binário onde dois valores sucessivos diferem em somente um bit

- Também conhecido como código binário refletido, pois o código de Gray para n bits pode ser obtido a partir da reflexão do código de Gray para (n-1) bits em torno de um eixo situado ao término do código
  - Adiciona-se "0" como bit mais significativo (MSB - Most Significant Bit) acima do eixo
  - Adiciona-se "1" como MSB abaixo do eixo.

Código de Gray

1 bit

0
1

2 bits

0	0
0	1
1	1
1	0

Refletir

3 bits

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

Refletir

4 bits

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

Refletir

Código de Gray

- Conversão Binário → Gray
  - g_i = i-ésimo bit do código de Gray
    - g0 = MSB
  - b_i = i-ésimo bit do código binário
    - b_0 = MSB

$$g_0 = b_0$$

$$g_i = \begin{cases} b_i = b_{i-1} \rightarrow g_i = 0 \\ b_i \neq b_{i-1} \rightarrow g_i = 1 \end{cases}$$

Decimal	Binário	Gray
	b_0 b_1 b_2	g_0 g_1 g_2
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1
6	1 1 0	1 0 1
7	1 1 1	1 0 0

Código de Gray

- Conversão Binário → Gray
  - g_i = i-ésimo bit do código de Gray
    - g0 = MSB
  - b_i = i-ésimo bit do código binário
    - b_0 = MSB

Decimal	Binário	Gray
	b_0 b_1 b_2	g_0 g_1 g_2
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1
6	1 1 0	1 0 1
7	1 1 1	1 0 0

$$g_0 = b_0$$

$$g_i = \begin{cases} b_i = b_{i-1} \rightarrow & g_i = 0 \\ b_i \neq b_{i-1} \rightarrow & g_i = 1 \end{cases}$$

$$g_i = b_i \; XOR \; b_{i-1}$$

Código de Gray

- Exemplo → Converter 101 de Binário para Gray

Binário

b_0	b_1	b_2
1	0	1

↓

Código de Gray

g_0	g_1	g_2

Código de Gray

- Exemplo → Converter 101 de Binário para Gray

Binário

b_0	b_1	b_2
1	0	1

i = 0

Código de Gray

g_0	g_1	g_2
1		

$$g_0 = b_0 = 1$$

Código de Gray

- Exemplo → Converter 101 de Binário para Gray

Binário

b_0	b_1	b_2
1	0	1

i = 1

Código de Gray

g_0	g_1	g_2
1	1	

$$g_i = \begin{cases} b_i = b_{i-1} \;\rightarrow\; g_i = 0 \\ b_i \neq b_{i-1} \;\rightarrow\; g_i = 1 \end{cases}$$

$$g_1 = \begin{cases} b_1 = b_0 \;\rightarrow\; g_1 = 0 \\ b_1 \neq b_0 \;\rightarrow\; g_1 = 1 \end{cases}$$

Código de Gray

- Exemplo → Converter 101 de Binário para Gray

Binário

b_0	b_1	b_2
1	0	1

Código de Gray

g_0	g_1	g_2
1	1	1

i = 2

$$g_i = \begin{cases} b_i = b_{i-1} \ \rightarrow \ g_i = 0 \\ b_i \neq b_{i-1} \ \rightarrow \ g_i = 1 \end{cases}$$

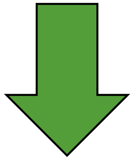
$$g_2 = \begin{cases} b_2 = b_1 \ \rightarrow \ g_2 = 0 \\ b_2 \neq b_1 \ \rightarrow \ g_2 = 1 \end{cases}$$

Código de Gray

- Exemplo → Converter 101 de Binário para Gray

Binário

b_0	b_1	b_2
1	0	1



Código de Gray

g_0	g_1	g_2
1	1	1

Decimal	Binário	Gray
	b_0 b_1 b_2	g_0 g_1 g_2
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1
6	1 1 0	1 0 1
7	1 1 1	1 0 0

Código de Gray

- Conversão Gray → Binário
  - g_i = i-ésimo bit do código de Gray
    - g0 = MSB
  - b_i = i-ésimo bit do código binário
    - b_0 = MSB

$$b_0 = g_0$$

$$b_i = \begin{cases} b_{i-1} = g_i \rightarrow b_i = 0 \\ b_{i-1} \neq g_i \rightarrow b = 1 \end{cases}$$

Decimal	Binário	Gray
	b_0 b_1 b_2	g_0 g_1 g_2
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1
6	1 1 0	1 0 1
7	1 1 1	1 0 0

Código de Gray

- Conversão Gray → Binário
  - g_i = i-ésimo bit do código de Gray
    - g0 = MSB
  - b_i = i-ésimo bit do código binário
    - b_0 = MSB

Decimal	Binário	Gray
	$b_0\ b_1\ b_2$	$g_0\ g_1\ g_2$
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1
6	1 1 0	1 0 1
7	1 1 1	1 0 0

$$b_0 = g_0$$

$$\boxed{b_i = \begin{cases} b_{i-1} = g_i \rightarrow \; b_i = 0 \\ b_{i-1} \neq g_i \rightarrow b = 1 \end{cases}}$$

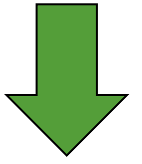
$$b_i = b_{i-1}\ XOR\ g_i$$

Código de Gray

- Exemplo → Converter 101 de Gray para Binário

Código de Gray

g_0	g_1	g_2
1	0	1



Binário

b_0	b_1	b_2

Código de Gray

- Exemplo → Converter 101 de Gray para Binário

Código de Gray

g_0	g_1	g_2
1	0	1

i = 0

Binário

b_0	b_1	b_2
1		

$$b_0 = g_0 = 1$$

Código de Gray

- Exemplo → Converter 101 de Gray para Binário

Código de Gray	g_0	g_1	g_2
	1	0	1

Binário	b_0	b_1	b_2
	1	1	

i = 1

$$b_i = \begin{cases} b_{i-1} = g_i \;\rightarrow\; b_i = 0 \\ b_{i-1} \neq g_i \;\rightarrow\; b_i = 1 \end{cases}$$

$$b_1 = \begin{cases} b_0 = g_1 \;\rightarrow\; b_1 = 0 \\ b_o \neq g_1 \;\rightarrow\; b_1 = 1 \end{cases}$$

Código de Gray

- Exemplo → Converter 101 de Gray para Binário

Código de Gray	g_0	g_1	g_2
	1	0	1

Binário	b_0	b_1	b_2
	1	1	0

i = 2

$$b_i = \begin{cases} b_{i-1} = g_i \ \rightarrow \ b_i = 0 \\ b_{i-1} \neq g_i \ \rightarrow \ b_i = 1 \end{cases}$$

$$b_2 = \begin{cases} b_1 = g_2 \ \rightarrow \ b_2 = 0 \\ b_1 \neq g_2 \ \rightarrow \ b_2 = 1 \end{cases}$$

Código de Gray

- Exemplo → Converter 101 de Gray para Binário

Código de Gray

g_0	g_1	g_2
1	0	1

Binário

b_0	b_1	b_2
1	1	0

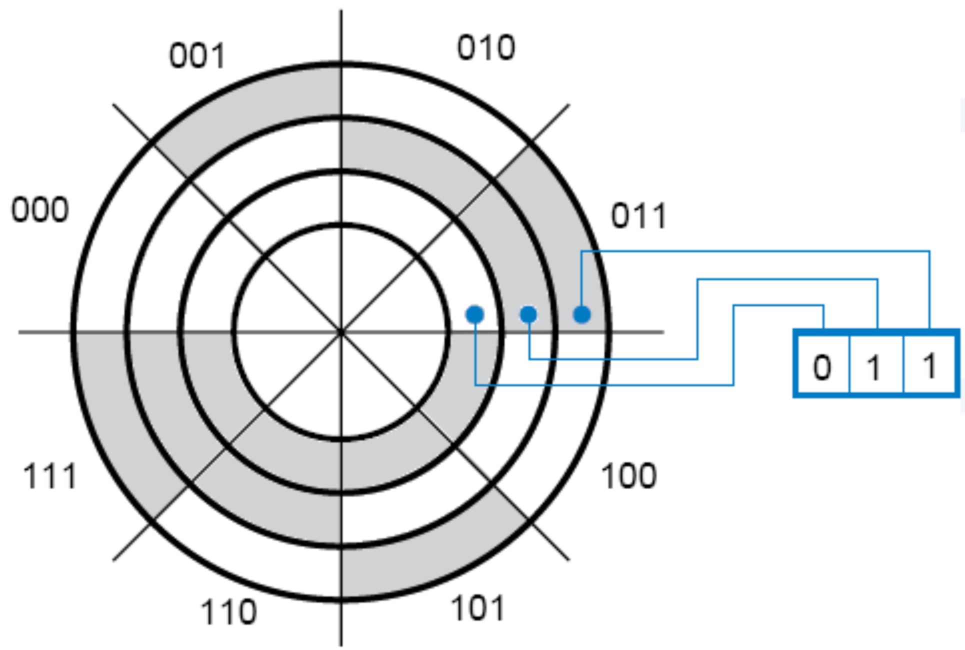
Decimal	Binário			Gray		
	b_0	b_1	b_2	g_0	g_1	g_2
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

Código de Gray

- Principais utilizações
  - Codificadores mecânicos
    - Pequenas mudanças de posição afetam apenas um único bit, diferentemente de certas situações que ocorrem com o código binário tradicional
  - Mapas de Karnaugh
    - O ordenamento das células é feito segundo o código de Gray, para possibilitar as simplificações booleanas
    - Visto nas próximas aulas

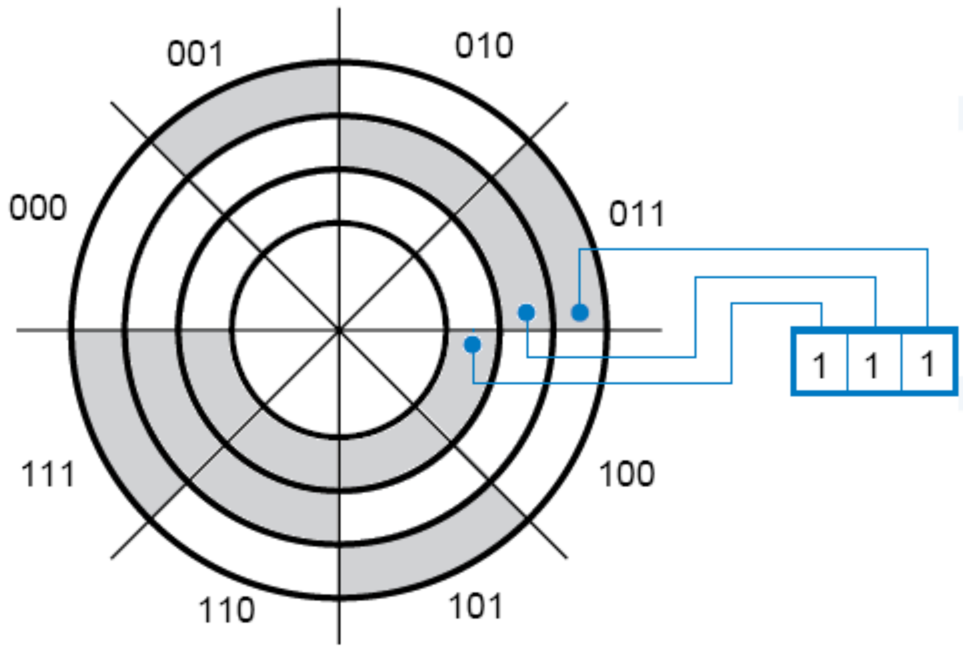
Código de Gray

- Ex: Codificador binário para um sistema mecânico rotacional



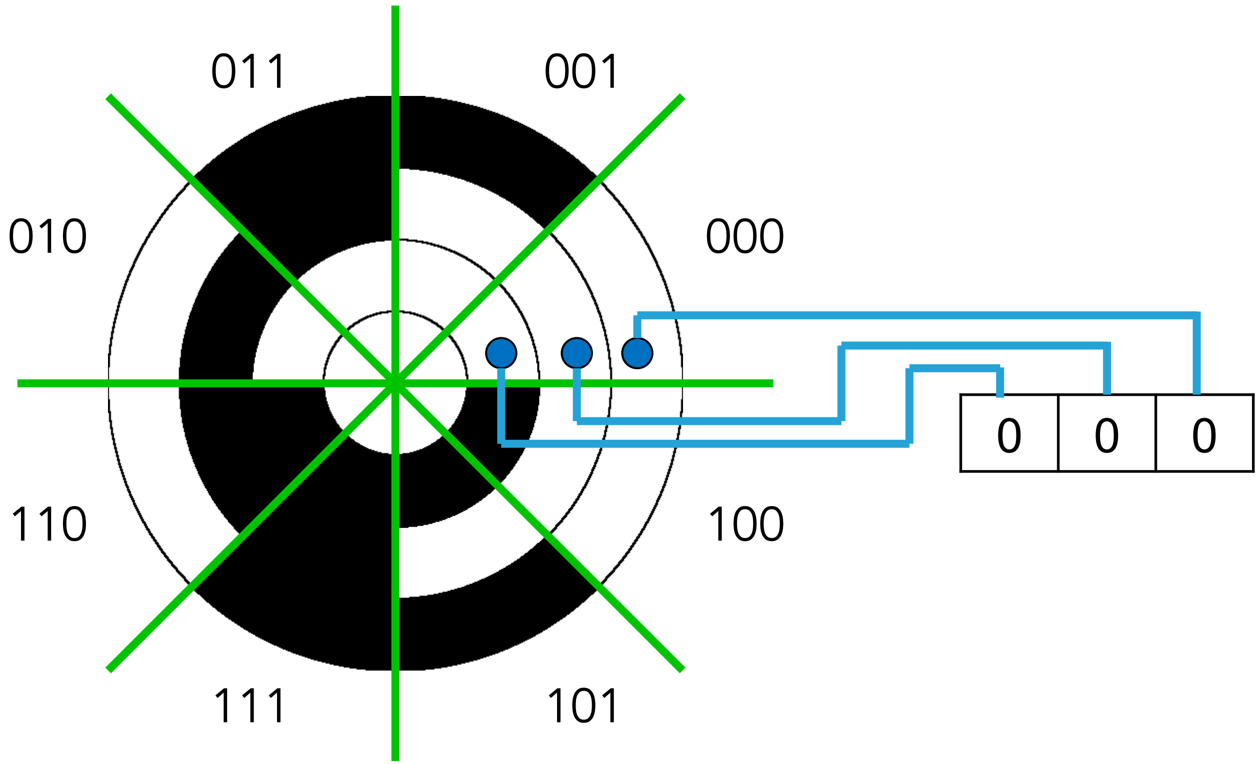
Código de Gray

- Ex: Codificador binário para um sistema mecânico rotacional
  - Caso haja desbalanceamento nas agulhas o erro produzido pode ser grande:



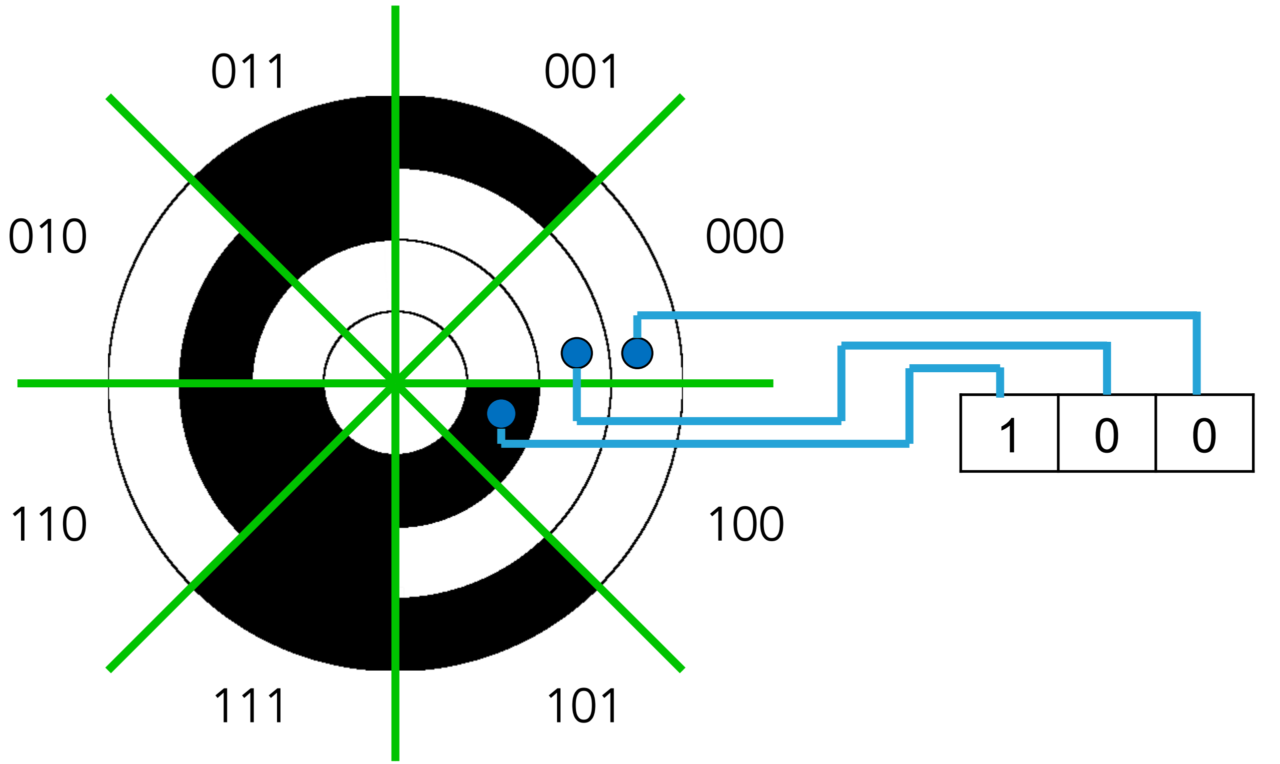
Código de Gray

- Ex: Codificador Gray para um sistema mecânico rotacional



Código de Gray

- Ex: Codificador Gray para um sistema mecânico rotacional



Código k de n

- São códigos ponderados constituídos por n bits
  - k bits são 1
  - n-k bits são 0

- Normalmente utilizados em detecção de erros transmissões de dados

- Número de combinações possíveis

$$\frac{n!}{k!(n-k)!}$$

Código k de n

- Exemplo: código 74210, ou 2 de 5
  - A detecção de erros pode ser feita simplesmente conferindo se o número de 1s for diferente de 2

Dígito decimal	(7	4	2	1	0)
0	1	1	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0

Caso especial

Código k de n

- Exemplo: código 2 de 7 bits ponderado
  - (50 43210), ou biquinário

Dígito decimal	(5	0	4	3	2	1	0)
0	0	1	0	0	0	0	1
1	0	1	0	0	0	1	0
2	0	1	0	0	1	0	0
3	0	1	0	1	0	0	0
4	0	1	1	0	0	0	0
5	1	0	0	0	0	0	1
6	1	0	0	0	0	1	0
7	1	0	0	0	1	0	0
8	1	0	0	1	0	0	0
9	1	0	1	0	0	0	0

Código k de n

- Exemplo: código 2 de 7 bits ponderado
  - Detecção de erros: número total de 1s é igual a 2, nos primeiros dois bits só há um 1, nos últimos cinco bits só há um 1

Dígito decimal	(5	0	4	3	2	1	0)
0	0	1	0	0	0	0	1
1	0	1	0	0	0	1	0
2	0	1	0	0	1	0	0
3	0	1	0	1	0	0	0
4	0	1	1	0	0	0	0
5	1	0	0	0	0	0	1
6	1	0	0	0	0	1	0
7	1	0	0	0	1	0	0
8	1	0	0	1	0	0	0
9	1	0	1	0	0	0	0

Códigos de Paridade

- Em códigos de paridade simples acrescenta-se um bit à palavra de tal forma que a paridade seja par ou ímpar

- O objetivo é a detecção de erros simples na transmissão de dados

Códigos de Paridade

Paridade

Código	Par	Nº de "1"s
0 0 0	0	0
0 0 1	1	2
0 1 0	1	2
0 1 1	0	2
1 0 0	1	2
1 0 1	0	2
1 1 0	0	2
1 1 1	1	4

Um circuito gerador de paridade par pode ser implementado apenas com uma porta XOR

Códigos de Paridade

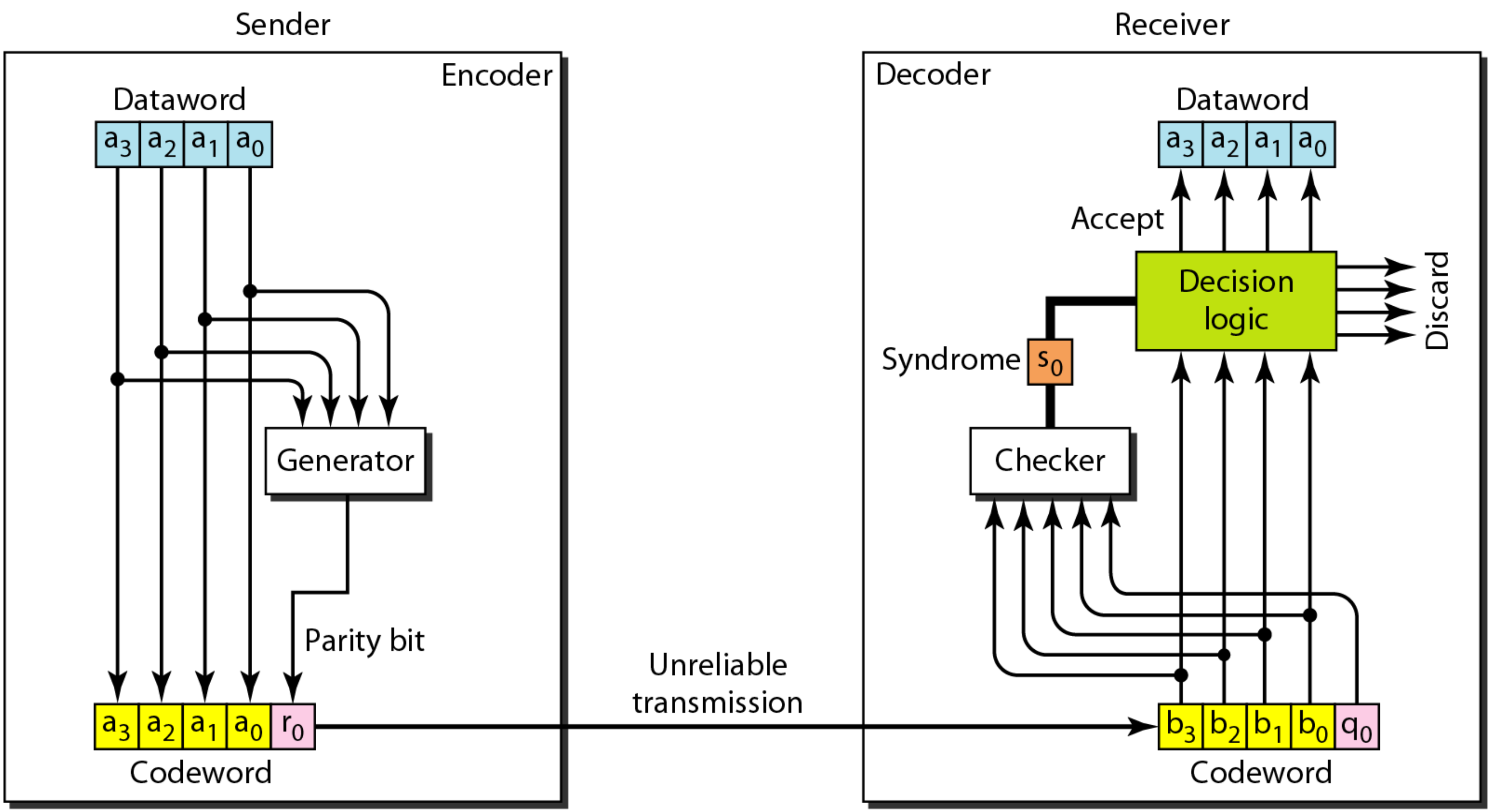
Paridade

Código	Ímpar	Nº de "1"s
0 0 0	1	1
0 0 1	0	1
0 1 0	0	1
0 1 1	1	3
1 0 0	0	1
1 0 1	1	3
1 1 0	1	3
1 1 1	0	3

Um circuito gerador de paridade ímpar pode ser implementado apenas com uma porta XNOR

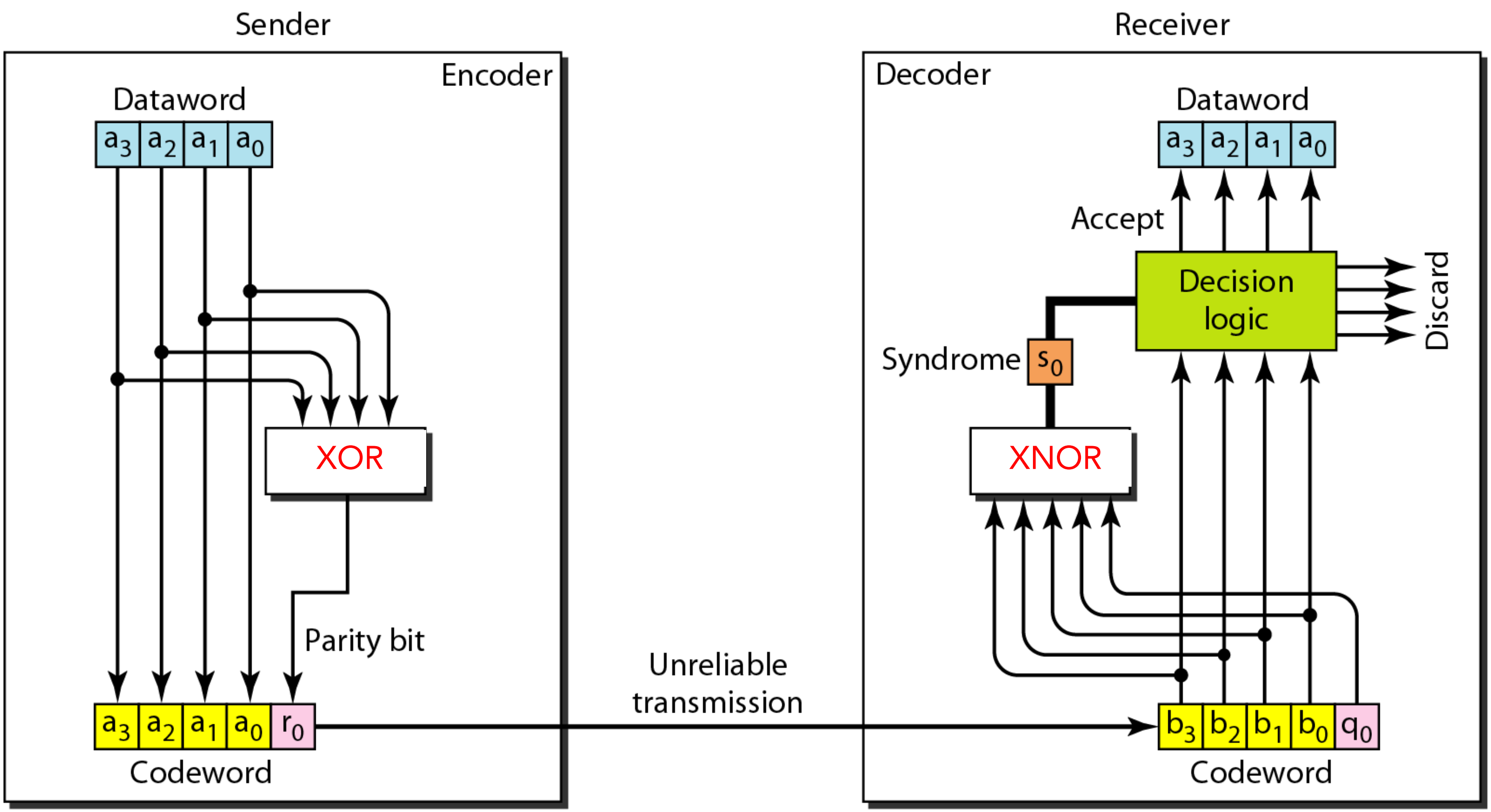
Códigos de Paridade

- Paridade em transmissão de dados



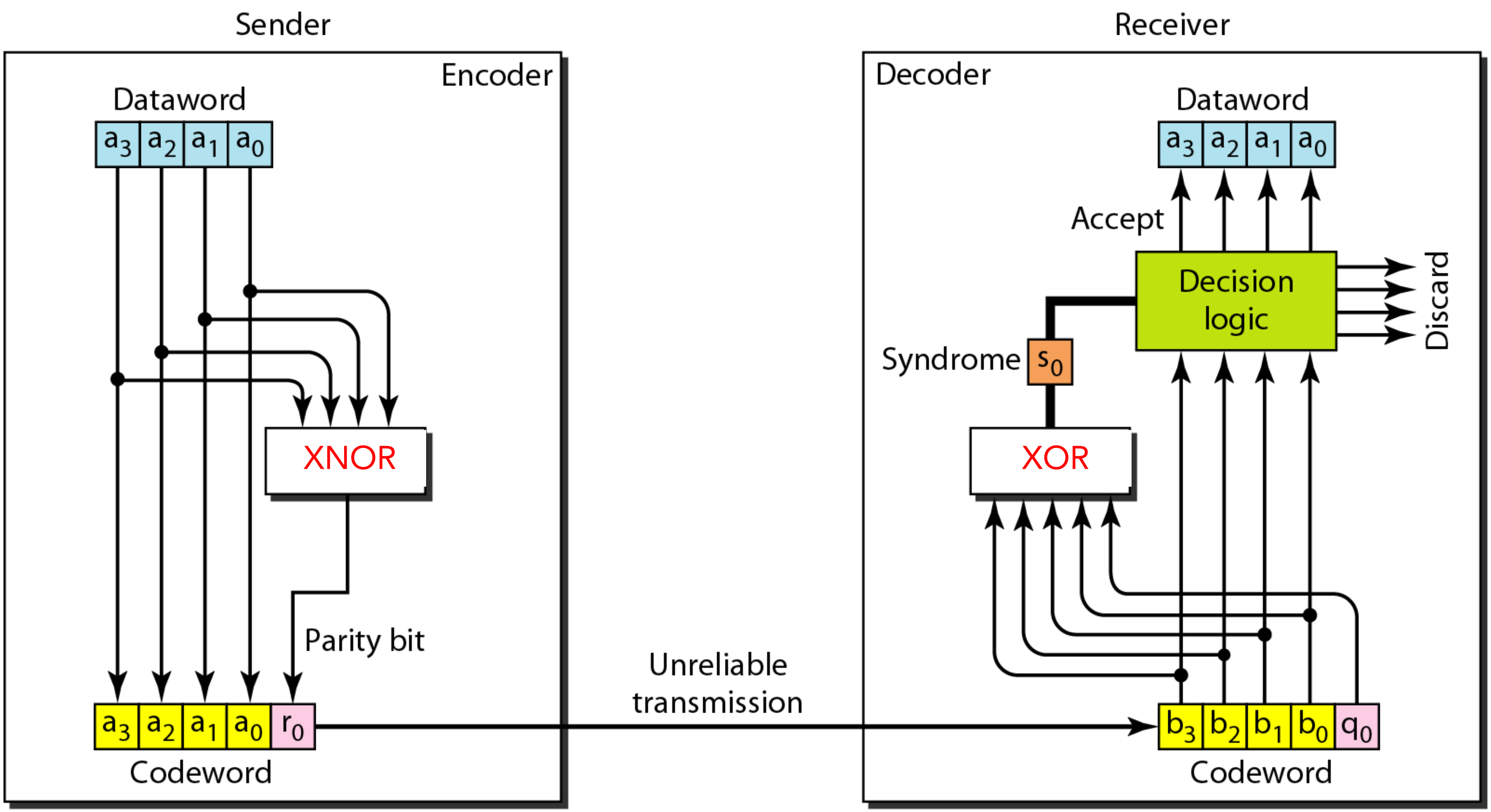
Códigos de Paridade

- Se paridade for PAR e considerando que se S_0=1 aceita os dados



Códigos de Paridade

- Se paridade for ÍMPAR e considerando que se $S_0=1$ aceita os dados



Código de Hamming

- Códigos de Hamming utilizam vários bits de paridade para
  - a detecção e correção de erros simples (em apenas um bit da palavra)
  - a detecção (mas não a correção) de erros duplos

- Código de Hamming terá:
  - d bits de dados
  - p bits de paridade

Código de Hamming

- Com d bits de dados, precisamos que a paridade seja:

$$2^p \geq d + p + 1, p \geq 2$$

Exemplo

- 4 bits de dados

$$2^p \geq d + p + 1, p \geq 2$$

  - Começamos com p = 2

$$2^2 \geq 4 + 2 + 1$$
$$4 \geq 7$$

  - Agora vamos tentar p = 3

$$2^3 \geq 4 + 3 + 1$$
$$8 \geq 8$$ 😀

Código de Hamming (7,4)

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7

Definimos que a paridade será PAR

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7

Teremos 3 bits de paridade → 7 - 4 = 3

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7
	P_1	P_2		P_4			

Os bits de paridade serão colocados nos bits que representam potências de 2: 1, 2, 4

Assim teremos as paridades: $P_1P_2P_4$

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7
	P_1	P_2	1	P_4	1	0	1

Os bits de dados são colocados de forma ordenada nos espaços restantes

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7
	P_1	P_2	1	P_4	1	0	1
P_1	P_1		1		1		1

Para calcular a paridade P_1 :

- Começamos a partir da posição de P_1 (1)
- Pegamos 1 bit
- Pulamos 1 bit
- Repetimos até o fim

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7
	P_1	P_2	1	P_4	1	0	1
P_1	P_1		1		1		1
P_2		P_2	1			0	1

Para calcular a paridade P_2 :

- Começamos a partir da posição de P_2 (2)
- Pegamos 2 bits
- Pulamos 2 bits
- Repetimos até o fim

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7
	P_1	P_2	1	P_4	1	0	1
P_1	P_1		1		1		1
P_2		P_2	1			0	1
P_4				P_4	1	0	1

Para calcular a paridade P_4 :

- Começamos a partir da posição de P_4 (4)
- Pegamos 4 bits
- Pulamos 4 bits
- Repetimos até o fim

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7
	1	P_2	1	P_4	1	0	1
P_1	1		1		1		1
P_2		P_2	1			0	1
P_4				P_4	1	0	1

Definimos a paridade como PAR :

$$P_1 = 1$$

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7
	1	0	1	P_4	1	0	1
P_1	1		1		1		1
P_2		0	1			0	1
P_4				P_4	1	0	1

Definimos a paridade como PAR :

$$P_2 = 0$$

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	0	1
P_1	1		1		1		1
P_2		0	1			0	1
P_4				0	1	0	1

Definimos a paridade como PAR :

$$P_4 = 0$$

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 1101

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	0	1
P_1	1		1		1		1
P_2		0	1			0	1
P_4				0	1	0	1

O valor a ser transmitido será : 1010101

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 1010111

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	1	1

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 1010111

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	1	1

BIT 6 CONTÉM ERRO!

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 1010111

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	1	1

Deveremos recalcular as paridades para detectar um erro!

Definimos a paridade como PAR

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 1010111

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	1	1
P_1	1		1		1		1

P_1 → 1 1 1 1 → Valor PAR de 1s → 😀

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 1010111

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	1	1
P_1	1		1		1		1
P_2		0	1			1	1

P_1 → 1 1 1 1 → Valor PAR de 1s → 😀

P_2 → 0 1 1 1 → Valor ÍMPAR de 1s → ☹️

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 1010111

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	1	1
P_1	1		1		1		1
P_2		0	1			1	1
P_4				0	1	1	1

P_1 → 1 1 1 1 → Valor PAR de 1s → 😀

P_2 → 0 1 1 1 → Valor ÍMPAR de 1s → ☹️

P_4 → 0 1 1 1 → Valor ÍMPAR de 1s → ☹️

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 1010111

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	1	1

Como descobrir o bit errado?

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 1010111

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	1	1

As paridades erradas foram a 2 e a 4:

BIT ERRADO = 2 + 4 = 6

P_1 → 1 1 1 1 → Valor PAR de 1s → 😀

P_2 → 0 1 1 1 → Valor ÍMPAR de 1s → ☹️

P_4 → 0 1 1 1 → Valor ÍMPAR de 1s → ☹️

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 1010111

Bit	1	2	3	4	5	6	7
	1	0	1	0	1	1	1

0

O bit 6 é 1, então deveria ser 0!!

Código de Hamming – Outro Exemplo

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7

Definimos que a paridade será PAR

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7

Teremos 3 bits de paridade → 7 - 4 = 3

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7
	P_1	P_2		P_4			

Os bits de paridade serão colocados nos bits que representam potências de 2: 1, 2, 4

Assim teremos as paridades: $P_1P_2P_4$

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7
	P_1	P_2	0	P_4	1	1	1

Os bits de dados são colocados de forma ordenada nos espaços restantes

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7
	P_1	P_2	0	P_4	1	1	1
P_1	P_1		0		1		1

Para calcular a paridade P_1 :

- Começamos a partir da posição de P_1 (1)
- Pegamos 1 bit
- Pulamos 1 bit
- Repetimos até o fim

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

	001	010	011	100	101	110	111
Bit	1	2	3	4	5	6	7
	P_1	P_2	0	P_4	1	1	1
P_1	P_1		0		1		1

Outra forma de definir os bits para cálculo da paridade P_1 :

- Escreva cada bit da mensagem em seu valor binário (como apresentado em amarelo)
- A paridade P_1 é calculada utilizando-se os bits que possuem 1 no bit menos significativo (demarcados em azul).

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7
	P_1	P_2	0	P_4	1	1	1
P_1	P_1		0		1		1
P_2		P_2	0			1	1

Para calcular a paridade P_2 :

- Começamos a partir da posição de P_2 (2)
- Pegamos 2 bits
- Pulamos 2 bits
- Repetimos até o fim

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

	001	010	011	100	101	110	111
Bit	1	2	3	4	5	6	7
	P_1	P_2	0	P_4	1	1	1
P_1	P_1		0		1		1
P_2		P_2	0			1	1

Outra forma de definir os bits para cálculo da paridade P_2 :

- Escreva cada bit da mensagem em seu valor binário (como apresentado em amarelo)
- A paridade P_2 é calculada utilizando-se os bits que possuem 1 no bit do meio (demarcados em azul).

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7
	P_1	P_2	0	P_4	1	1	1
P_1	P_1		0		1		1
P_2		P_2	0			1	1
P_4				P_4	1	1	1

Para calcular a paridade P_4 :

- Começamos a partir da posição de P_4 (4)
- Pegamos 4 bits
- Pulamos 4 bits
- Repetimos até o fim

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

	001	010	011	100	101	110	111
Bit	1	2	3	4	5	6	7
	P_1	P_2	0	P_4	1	1	1
P_1	P_1		0		1		1
P_2		P_2	0			1	1
P_4				P_4	1	1	1

Outra forma de definir os bits para cálculo da paridade P_4 :

- Escreva cada bit da mensagem em seu valor binário (como apresentado em amarelo)
- A paridade P_4 é calculada utilizando-se os bits que possuem 1 no bit mais significativo (demarcados em azul).

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7
	0	P_2	0	P_4	1	1	1
P_1	0		0		1		1
P_2		P_2	0			1	1
P_4				P_4	1	1	1

Definimos a paridade como PAR :

$P_1 = 0$

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7
	0	0	0	P_4	1	1	1
P_1	0		0		1		1
P_2		0	0			1	1
P_4				P_4	1	1	1

Definimos a paridade como PAR :

$P_2 = 0$

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7
	0	0	0	1	1	1	1
P_1	0		0		1		1
P_2		0	0			1	1
P_4				1	1	1	1

Definimos a paridade como PAR :

$$P_4 = 1$$

Código de Hamming

- Código de Hamming (7,4)
  - Exemplo → 4 bits de dados: 0111

Bit	1	2	3	4	5	6	7
	0	0	0	1	1	1	1
P_1	0		0		1		1
P_2		0	0			1	1
P_4				1	1	1	1

O valor a ser transmitido será : 0001111

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 0001111

Bit	1	2	3	4	5	6	7
	0	0	0	0	1	1	1

Exemplo: BIT 4 CONTÉM ERRO!

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 0001111

Bit	1	2	3	4	5	6	7
	0	0	0	0	1	1	1

Deveremos recalcular as paridades para detectar um erro!

Definimos a paridade como PAR

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 0001111

Bit	1	2	3	4	5	6	7
	0	0	0	0	1	1	1
P_1	0		0		1		1
P_2		0	0			1	1
P_4				0	1	1	1

P_1 → 0 0 1 1 → Valor PAR de 1s → 😀

P_2 → 0 0 1 1 → Valor PAR de 1s → 😀

P_4 → 0 1 1 1 → Valor ÍMPAR de 1s → ☹️

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 0001111

Bit	1	2	3	4	5	6	7
	0	0	0	0	1	1	1

A paridade errada foi a 4:

BIT ERRADO = 4

P_1 → 0 0 1 1 → Valor PAR de 1s → 😀

P_2 → 0 0 1 1 → Valor PAR de 1s → 😀

P_4 → 0 1 1 1 → Valor ÍMPAR de 1s → ☹️

Código de Hamming

- Código de Hamming (7,4) → CORREÇÃO
  - Exemplo → Valor recebido = 0001111

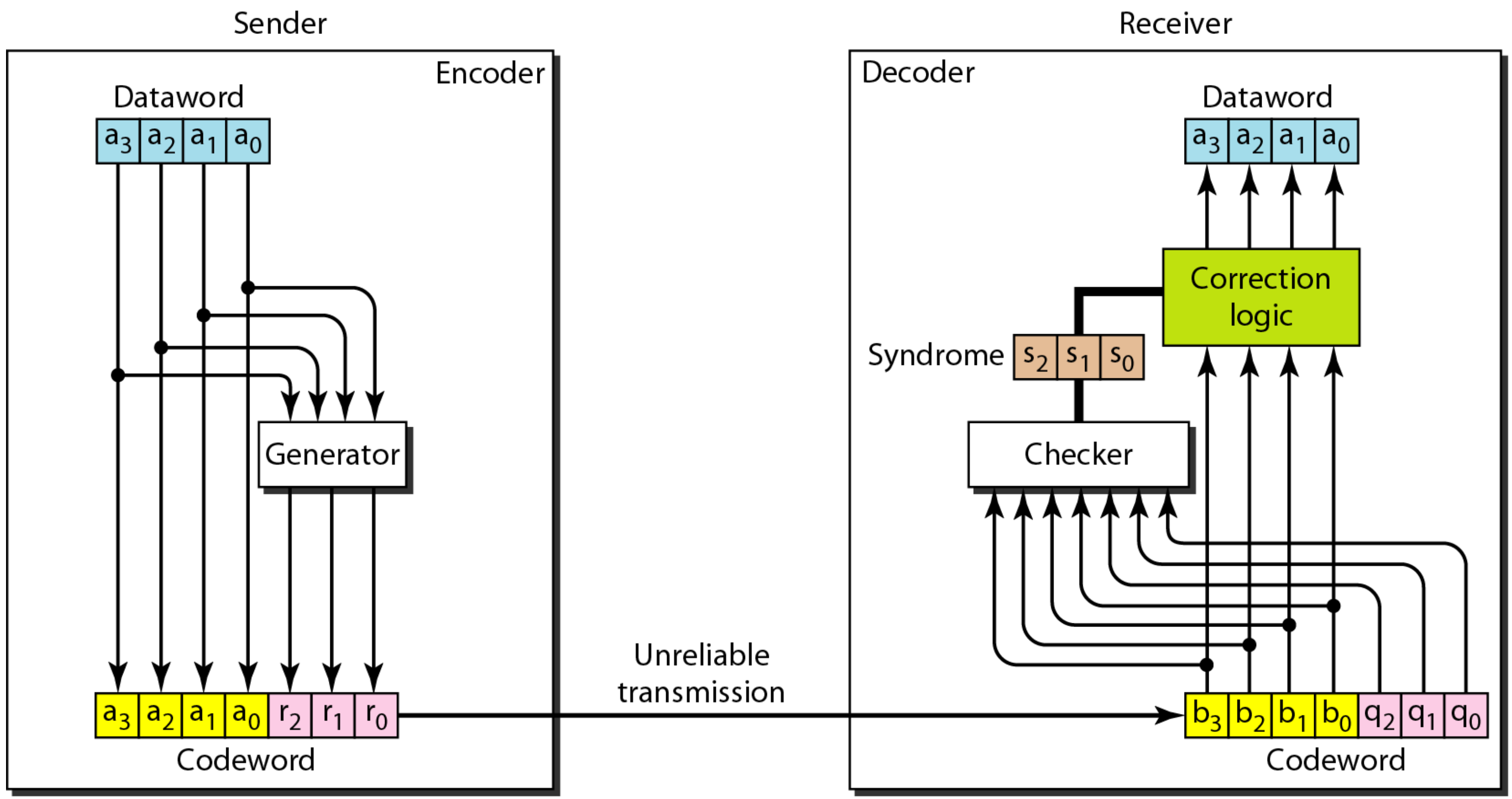
Bit	1	2	3	4	5	6	7
	0	0	0	0	1	1	1

1

O bit 4 é 0, então deveria ser 1!!

Código de Hamming

- Codificador/decodificador para o Código de Hamming



Código de Hamming

- *Códigos Hamming até um máximo de 255 bits*

Combinações de parâmetros do códigos de Hamming		
d	p	d + p
Bits de dados	Bits de paridade	Total da mensagem
1	2	3
4	3	7
11	4	15
26	5	31
57	6	63
120	7	127
247	8	255