PCS 3115

Sistemas Digitais I

Códigos para Detecção e Correção de Erros

Prof. Dr. Marcos A. Simplicio Jr.

versão: 3.0 (Jan/2016)

Adaptado por Glauber (2018)

Códigos para Detecção de Erros

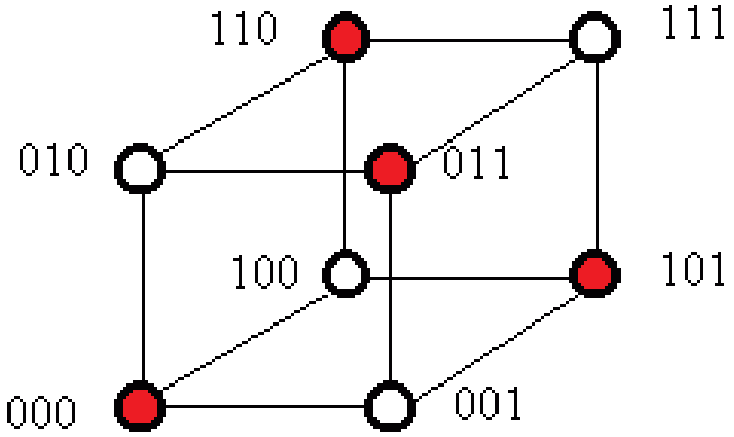
- **Erro**: dado alterado por causas físicas; pode ser temporário ou permamente.
- **Modelo de Independente Erros**: cada erro afeta exatamente um bit. Erros são independentes.
- **Código para Detecção de Erros:** n bits, mas menos de 2^n códigos válidos. Se uma cadeia de n bits é um código inválido, ela contém um erro (pelo menos um bit foi alterado).
- Alterar um bit em um código válido deve levar a um código não válido.

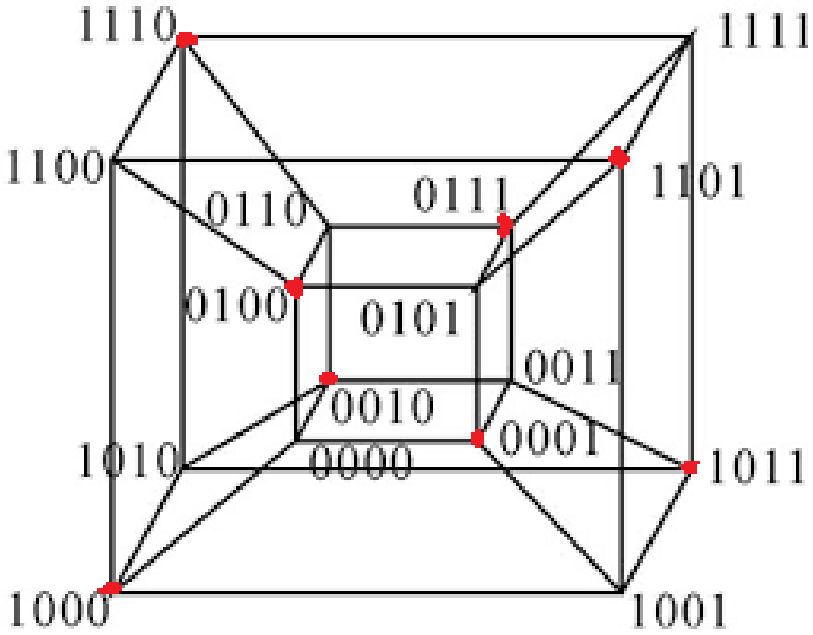
Paridade

- Código com n+1 bits, n bits de informação e 1 bit de paridade.
- **Bit de paridade**: igual a 1 sse há um número ímpar de 1’s entre os bits de informação (código de paridade par)
- Para cada uma das 2^n configurações dos bits de infomação, há apenas um valor pro bit de paridade que produz um código válido.
- Logo, temos 2^n códigos válidos e 2^n inválidos.
- Alterando apenas um bit em um código válido, tem-se um código não válido.

Distância de Hamming

- **Distância de Hamming**: número de bits diferentes entre duas cadeias de bits.
- Um código detecta todos os erros isolados se a distância mínima entre pares de códigos válidos é maior ou igual a 2.





Gerador/Detector de Paridade

- Recap: portas lógicas de OU-exclusivo (XOR)
  - $X \oplus Y = (X' \cdot Y) + (X \cdot Y')$
  - Resultados (equivalentes)
    - 1 se **apenas uma** das entradas for 1, 0 caso contrário
    - 1 se **ambas as entradas são diferentes**, 0 caso contrário

- Tabela-verdade

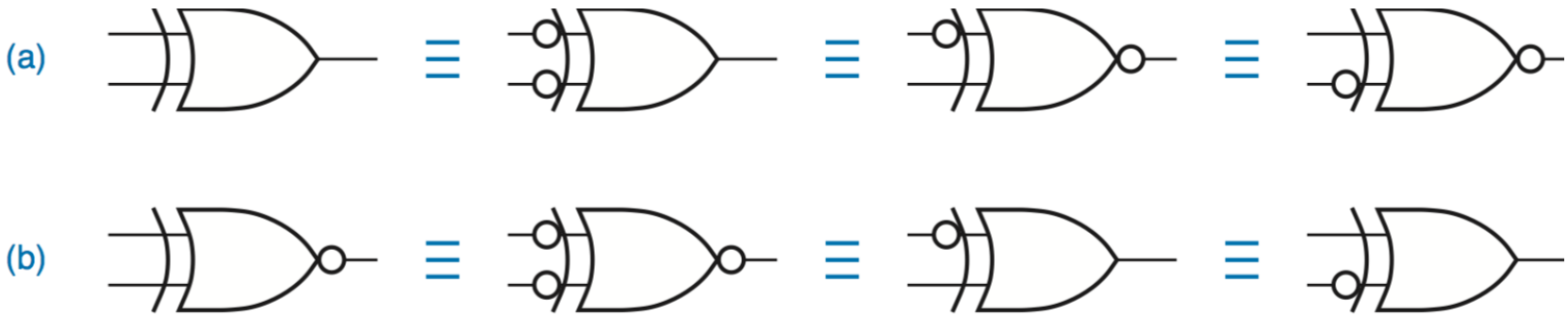
X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0

- Porta lógica

(representação gráfica)

Gerador/Detector de Paridade

- Símbolos alternativos para XOR (a) e XNOR (b)



Gerador/Detector de Paridade

- Exemplo: 2 bits
- $\sum_{odd}$: indica que entrada tem número ímpar de bits 1
- $\sum_{even}$: indica que entrada tem número par de bits 1

iguais

X	Y	X⊕Y	$\sum_{odd}$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

X	Y	(X⊕Y)'	$\sum_{even}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	1	1

Gerador/Detector de Paridade

- E para mais bits...?
  - Ex.: 3 bits

X	Y	Z			$\sum_{odd}$	# 1s
0	0	0			0	0
0	0	1			1	1
0	1	0			1	1
0	1	1			0	2
1	0	0			1	1
1	0	1			0	2
1	1	0			0	2
1	1	1			1	3

Gerador/Detector de Paridade

- E para mais bits...?
  - Ex.: 3 bits

X	Y	Z	W=Y⊕Z		$\sum_{odd}$	# 1s
0	0	0	0		0	0
0	0	1	1		1	1
0	1	0	1		1	1
0	1	1	0		0	2
1	0	0	0		1	1
1	0	1	1		0	2
1	1	0	1		0	2
1	1	1	0		1	3

Gerador/Detector de Paridade

- E para mais bits...?
  - Ex.: 3 bits

X			W=Y⊕Z	X⊕W	$\sum_{odd}$	# 1s
0			0	0	0	0
0			1	1	1	1
0			1	1	1	1
0			0	0	0	2
1			0	1	1	1
1			1	0	0	2
1			1	0	0	2
1			0	1	1	3

Gerador/Detector de Paridade

- E para mais bits...?
  - Basta cascatear os geradores de paridade de 2 bits!

X	Y	Z	W=Y⊕Z	X⊕W	$\sum_{odd}$	# 1s
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	1	1	1	1
0	1	1	0	0	0	2
1	0	0	0	1	1	1
1	0	1	1	0	0	2
1	1	0	1	0	0	2
1	1	1	0	1	1	3

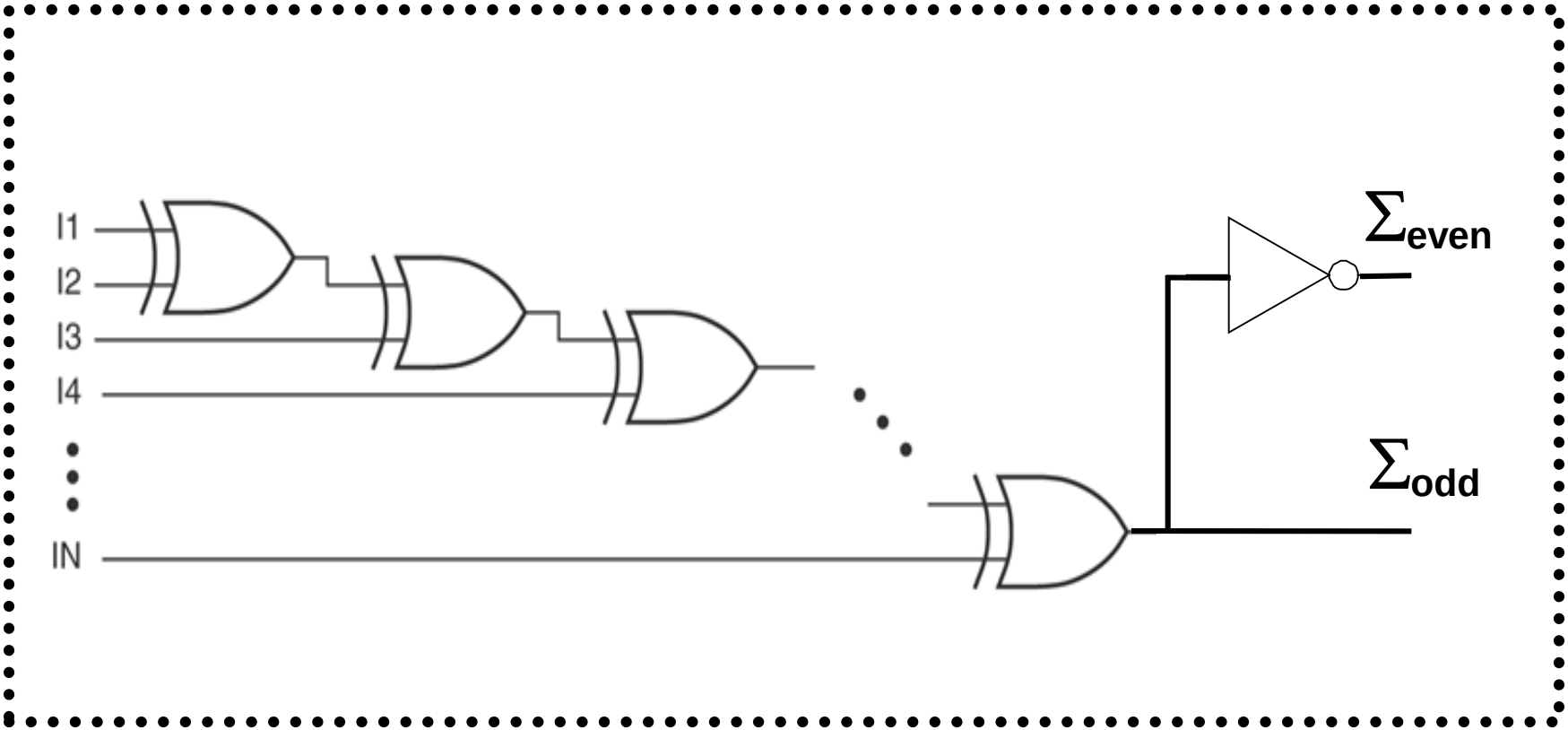
Gerador/Detector de Paridade

- De maneira geral, se X e Y representam a paridade de palavras x e y, então X ⊕ Y representa a paridade de xy (x concatenado com y)

- Se x e y têm um número par de 1's, então xy também têm paridade par; X=Y=0 e X⊕Y=0.
- Se x e y têm paridade ímpar, X=Y=1 e X⊕Y=0.
- Se x tem paridade par, e y, ímpar, então um número par de 1's, X⊕Y=1 e xy tem paridade ímpar.

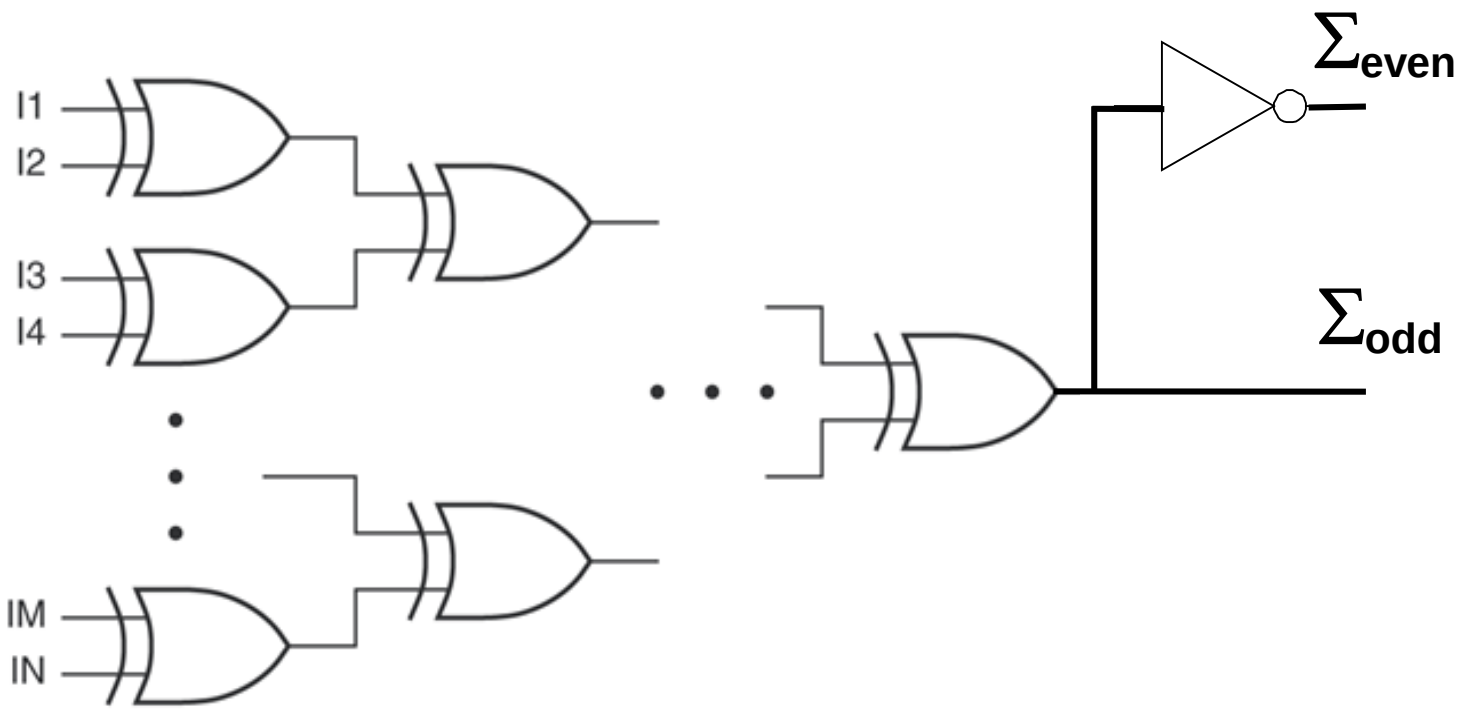
Gerador/Detector de Paridade

- E para mais bits...?
  - Basta cascatear os geradores de paridade de 2 bits!
  - Em série: atraso total de N ... ☹



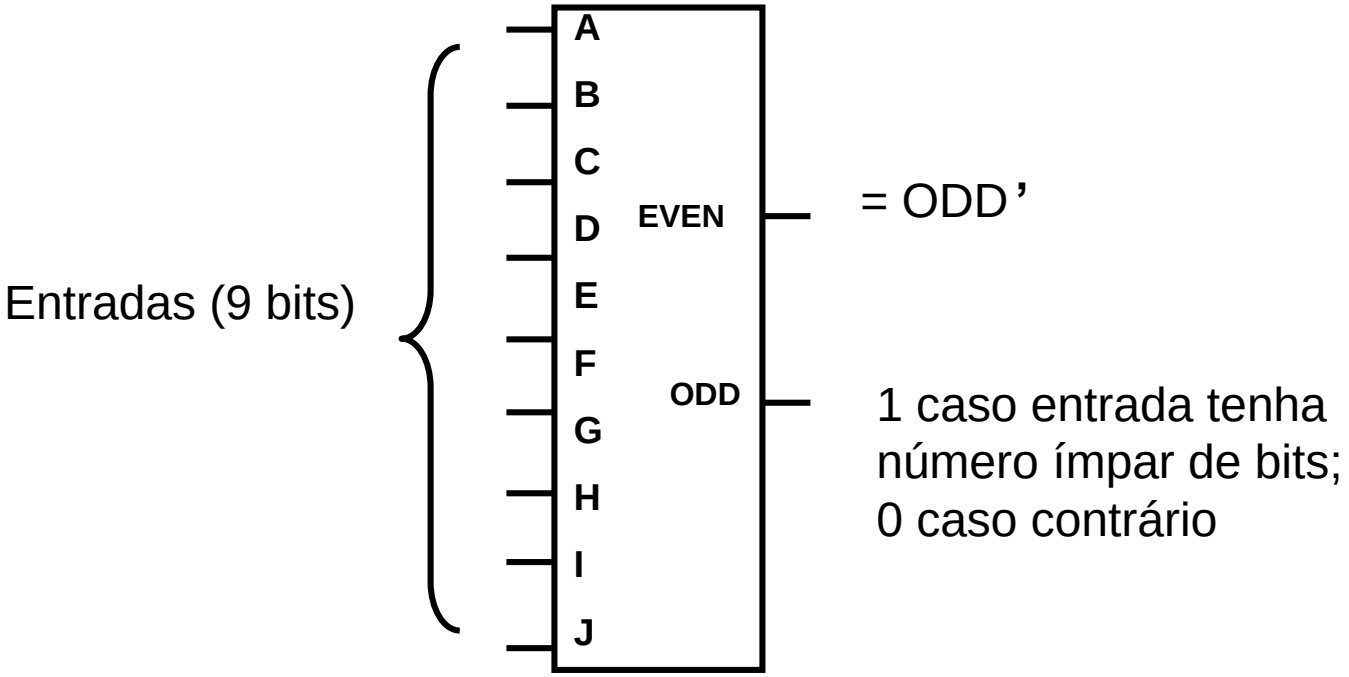
Gerador/Detector de Paridade

- E para mais bits...?
  - Basta cascatear os geradores de paridade de 2 bits!
  - Em árvore: atraso de ~lg(N) ☺



Gerador/Detector de Paridade (PAR)

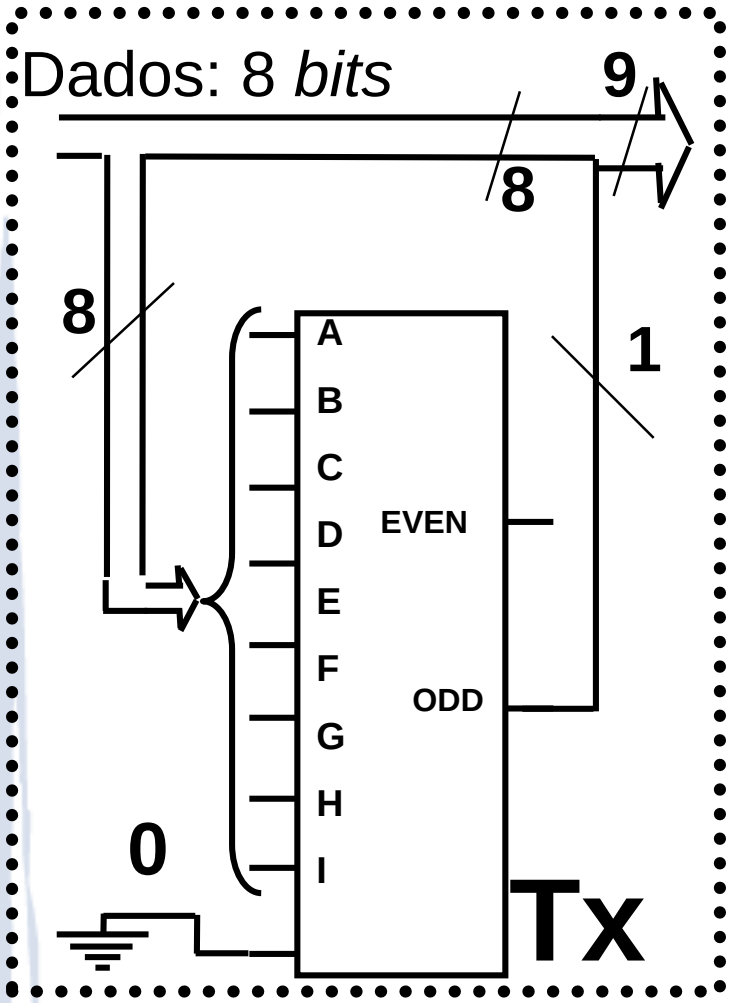
- Bloco gerador/detector de paridade 74x280

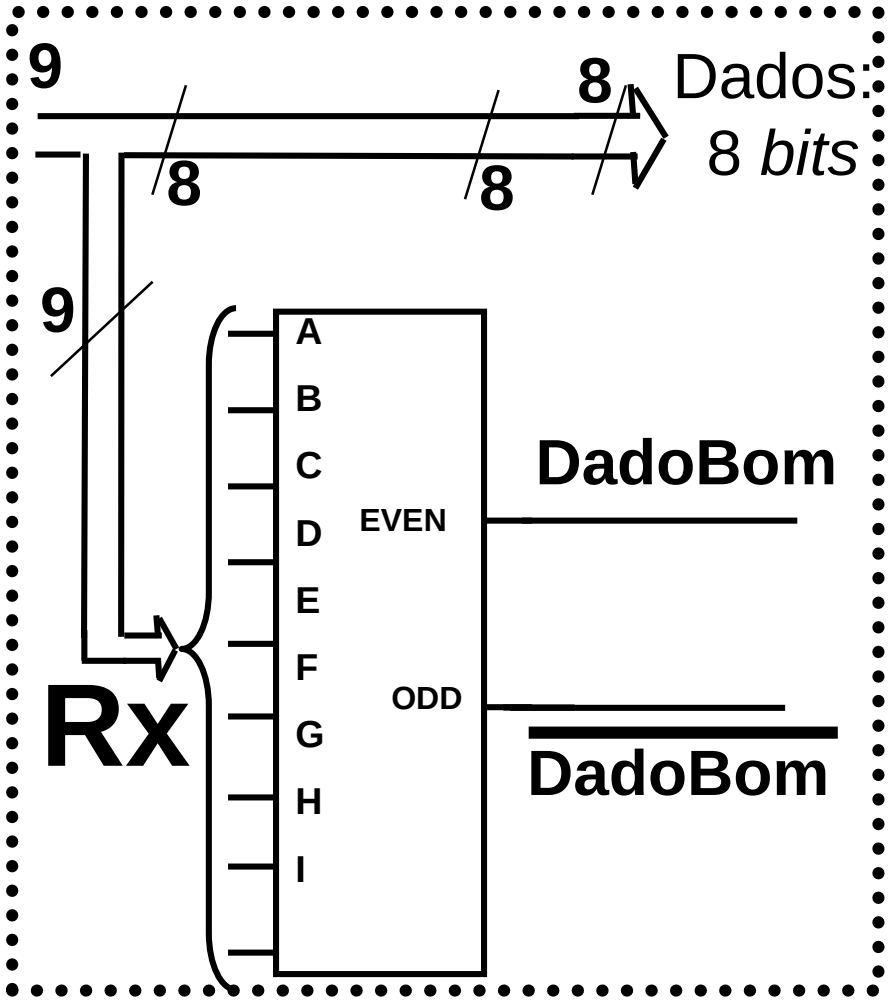


Gerador/Detector de Paridade

Notação: paridade par

Transmissão de **8 *bits*** e **verificação** na **Recepção [1/2]**:

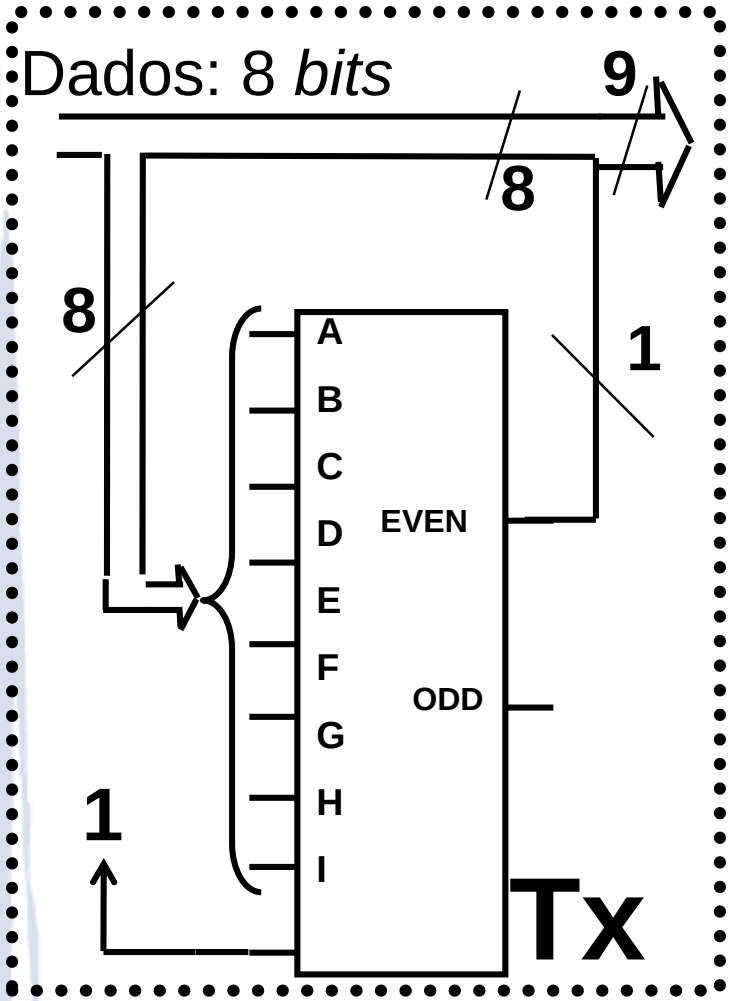


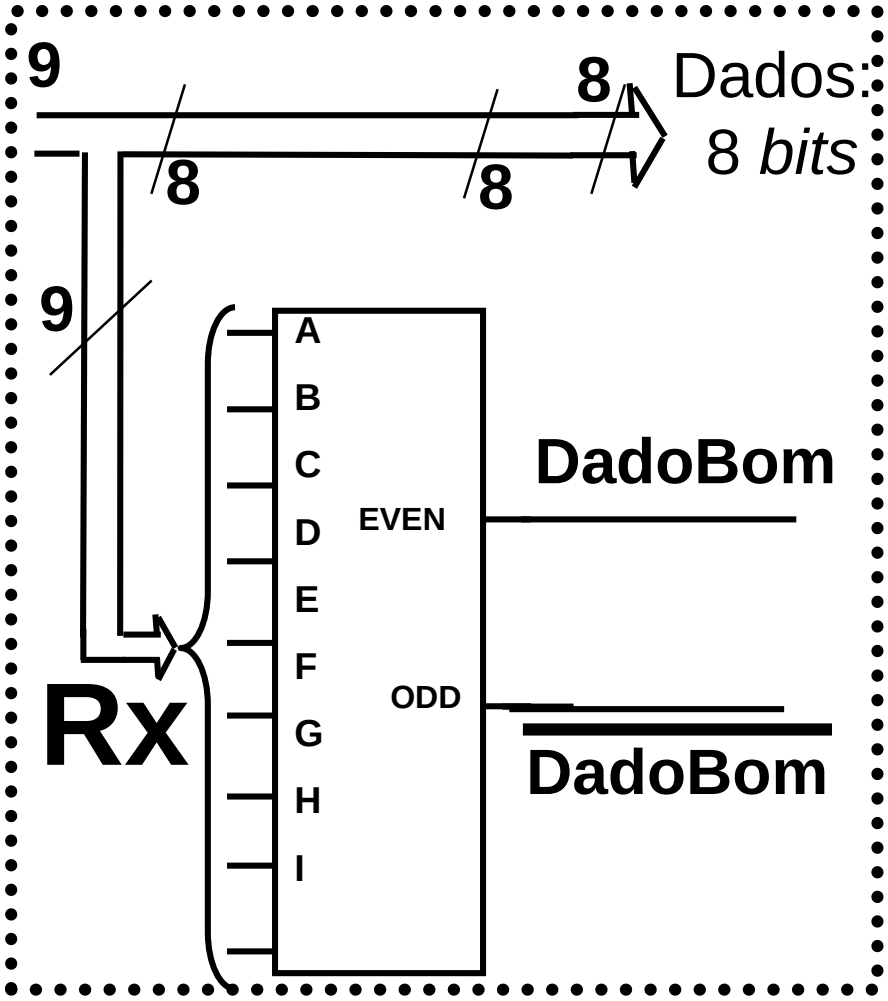


Gerador/Detector de Paridade

Notação: paridade par

Transmissão de **8 *bits*** e **verificação** na **Recepção [2/2]**:

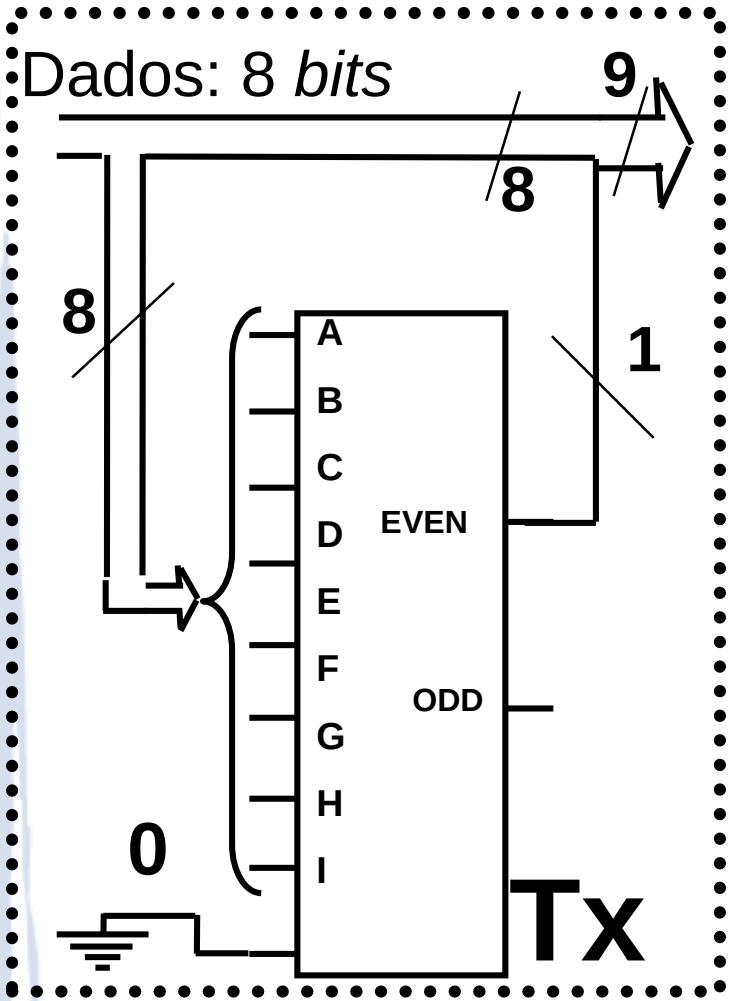


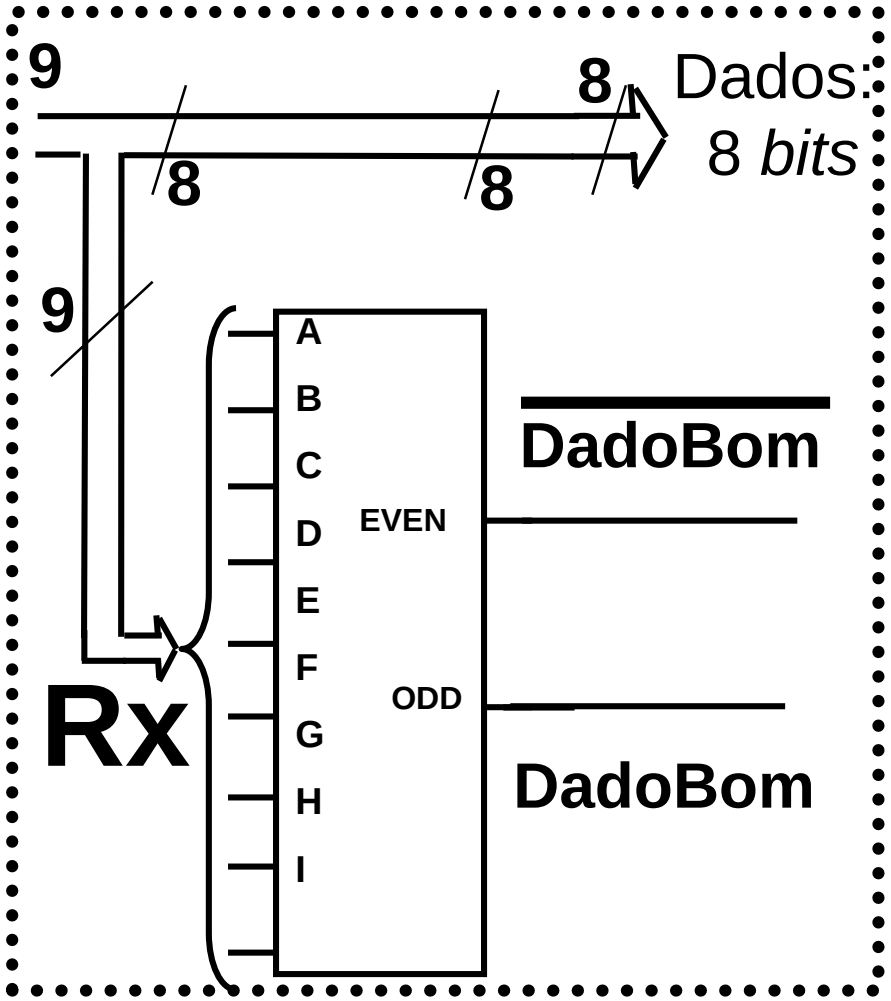


Gerador/Detector de Paridade

Notação: paridade ímpar

Transmissão de **8 *bits*** e **verificação** na **Recepção [1/2]**:

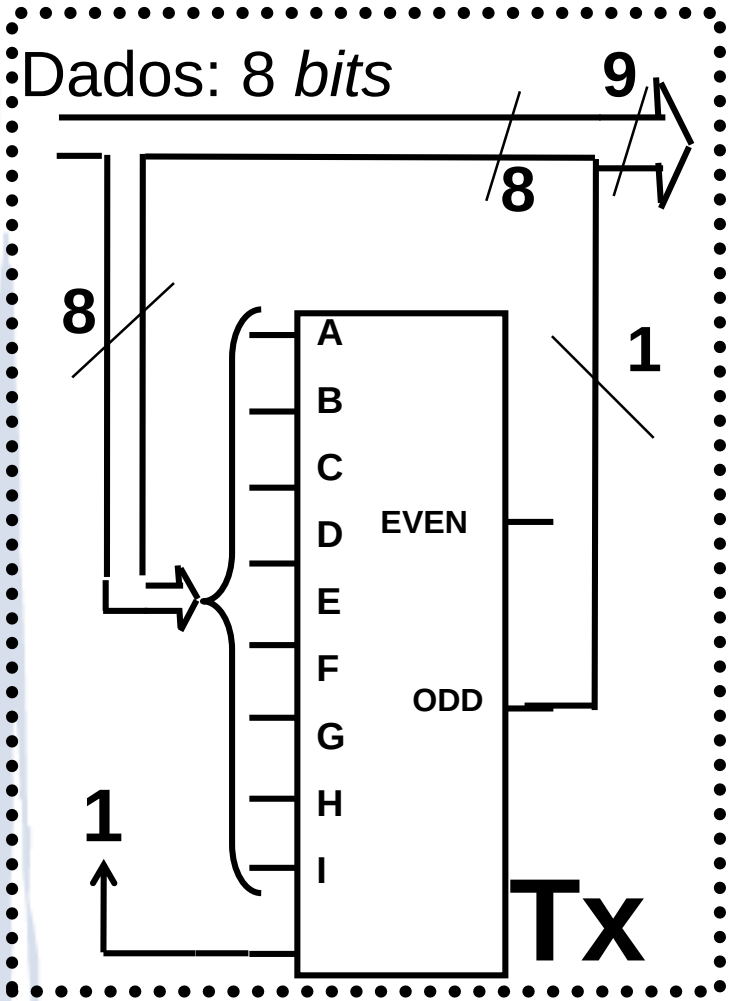


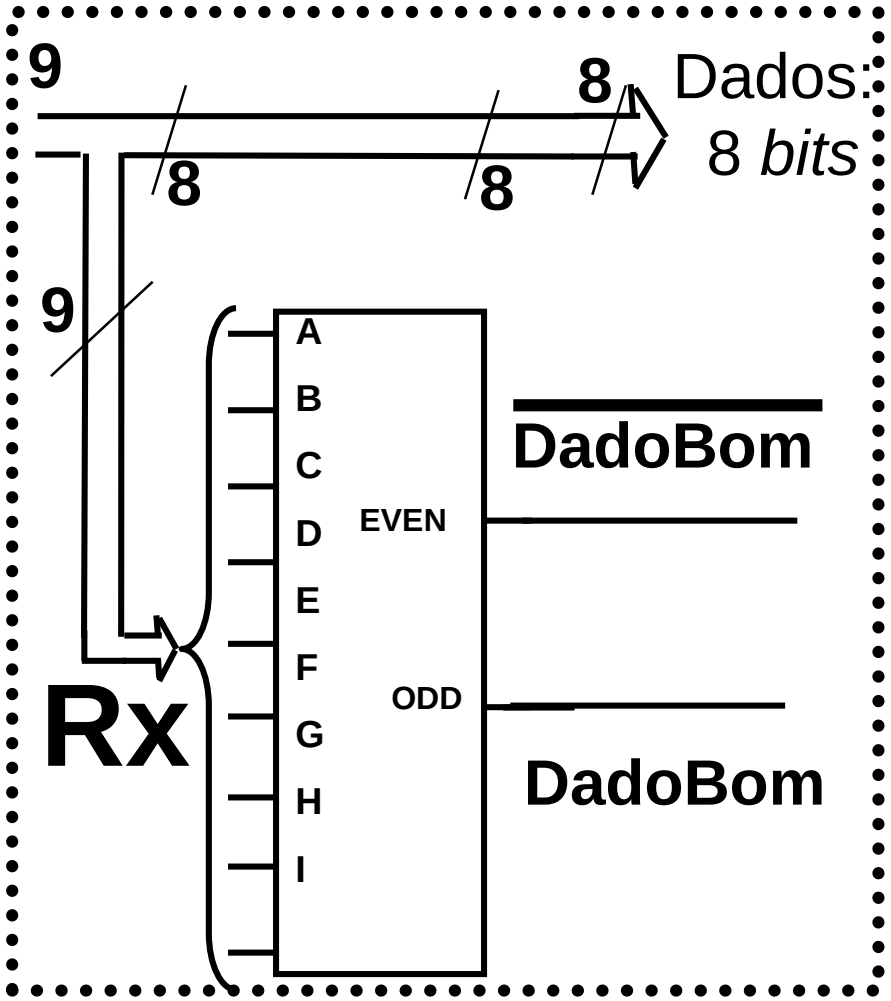


Gerador/Detector de Paridade

Notação: paridade ímpar

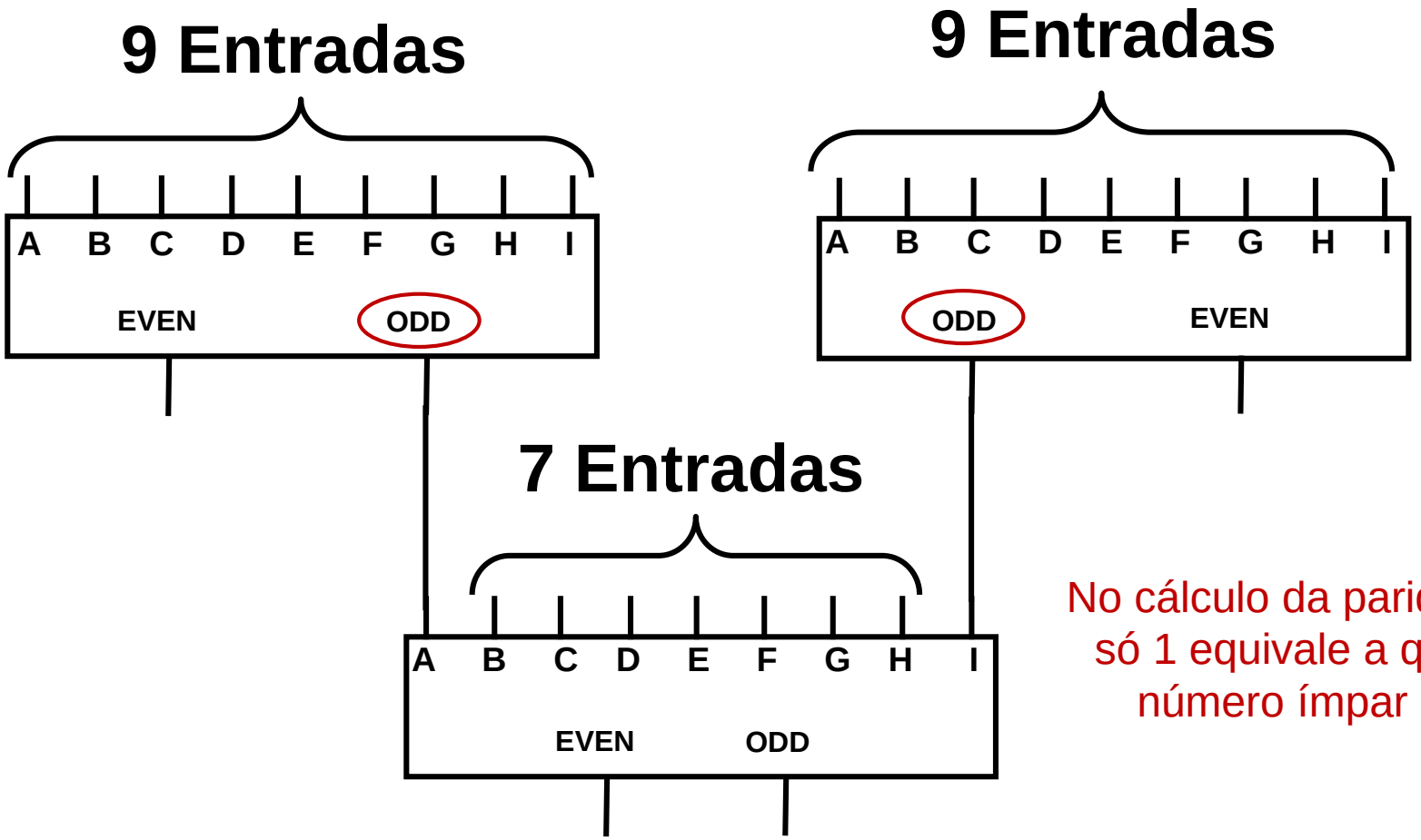
Transmissão de **8 *bits*** e **verificação** na **Recepção [2/2]**:





Gerador/Detector de Paridade

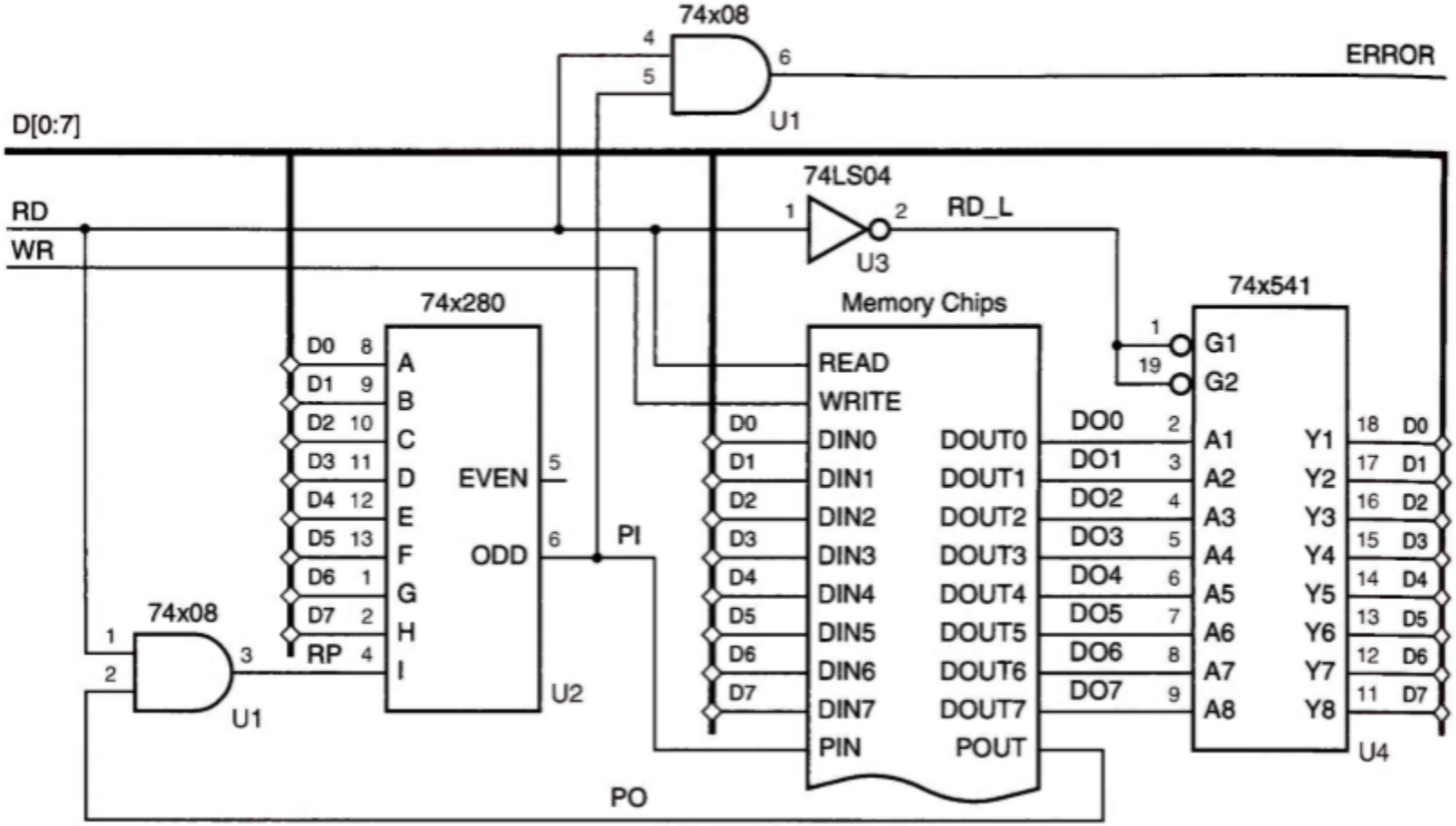
- **Associação** de Blocos Calculadores de Paridade:



No cálculo da paridade, um só 1 equivale a qualquer número ímpar de 1s

Gerador/Detector de Paridade

- Memória com Blocos Calculadores de Paridade:



Gerador/Detector de Paridade

- XOR de 3 bits em VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;

entity vxor3 is
    port (
        A, B, C: in STD_LOGIC;
        Y: out STD_LOGIC
    );
end vxor3;

architecture vxor3 of vxor3 is
begin
  Y <= A xor B xor C;
end vxor3;
```

Gerador/Detector de Paridade

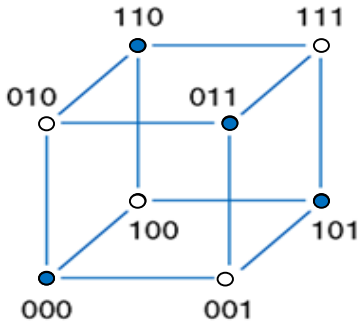
```
library IEEE;
use IEEE.std_logic_1164.all;

entity V74x280 is
    port (
        I: in STD_LOGIC_VECTOR (1 to 9);
        EVEN, ODD: out STD_LOGIC
    );
end V74x280;

architecture V74x280s of V74x280 is
component vxor3
 port (A, B, C: in STD_LOGIC; Y: out STD_LOGIC);
end component;
signal Y1, Y2, Y3, Y3N: STD_LOGIC;
begin
  U1: vxor3 port map (I(1), I(2), I(3), Y1);
  U2: vxor3 port map (I(4), I(5), I(6), Y2);
  U3: vxor3 port map (I(7), I(8), I(9), Y3);
  Y3N <= not Y3;
  U4: vxor3 port map (Y1, Y2, Y3, ODD);
  U5: vxor3 port map (Y1, Y2, Y3N, EVEN);
end V74x280s;
```

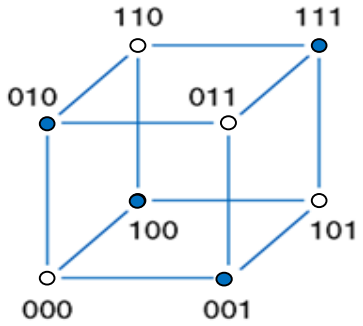
Verificador de paridade de 9 bits: abordagem **estrutural**

Detecção de erros: um pouco de teoria

- Nos códigos de Paridade, onde a distância mínima é 2, detectamos erros em **número ímpar de bits**

Paridade par

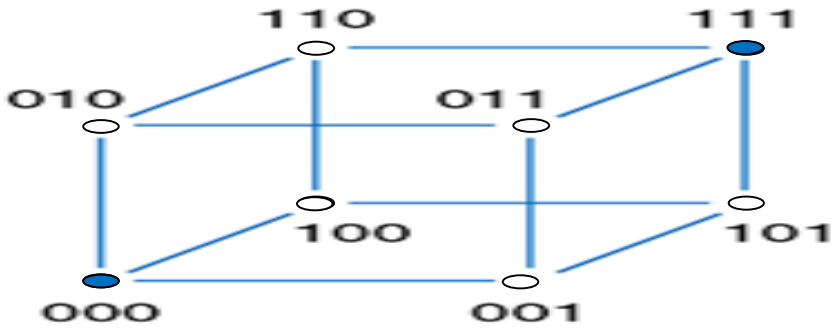




Paridade ímpar

- Códigos com **distância** mínima *m* conseguem detectar erros em até $d = m - 1$ bits!

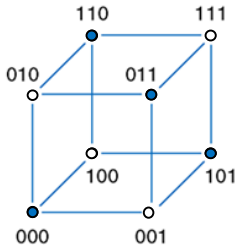
Ex: m=3



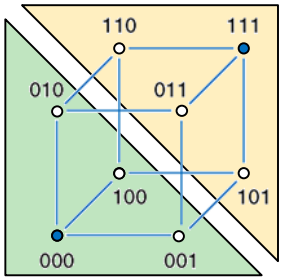
Correção de erros

- Como corrigir (não apenas detectar) erros?
  - Ideia: podemos corrigir uma palavra inválida para a palavra de código válida mais próxima dela!
- É possível corrigir erros de 1 bit com um código de distância 2?
  - Não: palavras inválidas são equidistantes das palavras válidas

000 ←? 010 →? 110

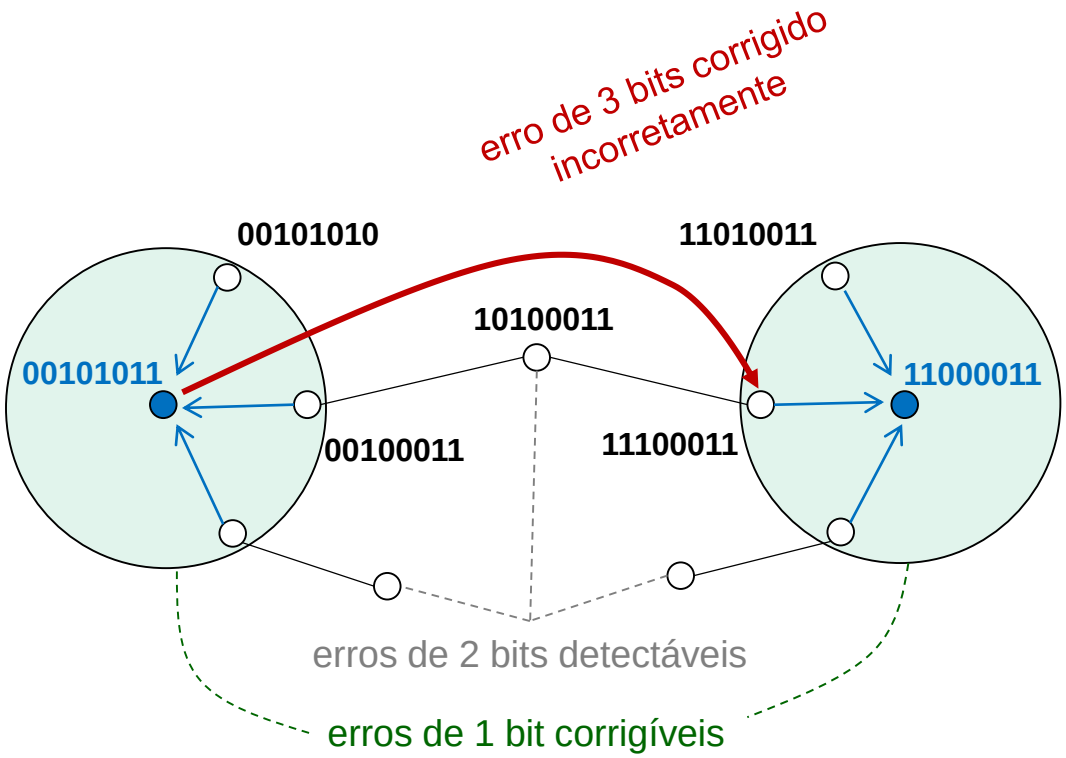


- E se a distância for 3?
  - Sim: 010, 100, 001 ➔ 000
    110, 011, 101 ➔ 111



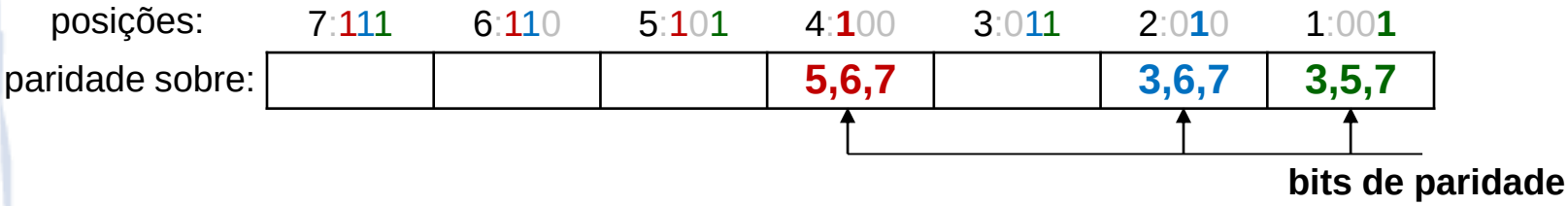
Correção de erros: um pouco de teoria

- Código com distância mínima m = 2*c*+*d*+1: corrige até *c* erros e é capaz de detectar *c*+*d* erros
  - Ex.: m = 4, podemos corrigir erros (c = 1, d = 1) ou apenas detectar erros (c = 0, d = 3)



Código de Hamming

- Código com as seguintes características:
  - Distância mínima m = 3
  - Palavras de até $(2^i - 1)$ bits, dentre eles *i* bits de verificação
- Método de construção:
  - Enumere os bits de 1 a $2^i - 1$
  - Posições que são potências de 2 são bits de paridade *p*
    - Ou seja, pos(p) = 2^n, para $0 \leq n < i$
  - Cada bit de paridade *p* abrange todos os bits para os quais o AND lógico da posição de *p* e do bit de informação for ≠ 0
    - Representando as posições em binário, cada bit de paridade corresponde ao **grupo** de posições



Código de Hamming

- A distância é no mínimo 3 porque
  - Trocar 1 bit na posição *j* qualquer leva a palavra inválida: posição j está associada a pelo menos um grupo
  - Trocar 2 bits nas posições *j* e *k* também: grupos envolvendo *j* **e** *k* não detectam erro, mas existe ao menos um grupo que não contém ambos *j* e *k*
    - Afinal, *j* e *k* diferem em pelo menos 1 bit

posições:	7:111	6:110	5:101	4:100	3:011	2:010	1:001
paridade sobre:	*		*	5,6,7		3,6,7	3,5,7

Exemplo:

erro em j = 7 ➔ invalida bits de paridade nas posições 4, 2 e 1

erro em j = 7 e k = 5➔ invalida bit de paridade na posição 2

Código de Hamming

- Correção de erros de 1 bit é simples (c = 1, d =0) :
  - Posição do bit em que houve a inversão é dada pela representação binária dos bits de paridade

posições:	7:111	6:110	5:101	4:100	3:011	2:010	1:001
paridade sobre:	*		*	5,6,7		3,6,7	3,5,7

Posição do erro	Bits de paridade afetados	Posição do erro	Bits de paridade afetados
7	4+2+1 = 7	3	2+1 = 3
6	4+2 = 6	2	2
5	4+1 = 5	1	1
4	4 = 4		

=

Código de Hamming

- Alguns detalhes adicionais
  - Distância 3 pode ser **estendida para distância 4**: basta adicionar um **bit de paridade calculado sobre todos os bits**
    - Obtém-se: (c = 1, d = 1), ou então (c=0, d=3)
  - Normalmente, em uma comunicação os bits de paridade são colocados nas **posições menos significativas** da palavra
    - Ou seja: bit de paridade da posição 2^i colocado na posição i

	Código de distância mínima 3		**Código de distância mínima 4**	
Bits de dados	Bits de paridade	Bits totais	Bits de paridade	Bits totais
1	2	3	3	4
≤ 4	3	≤ 7	4	≤ 8
≤ 11	4	≤ 15	5	≤ 16
≤ 26	5	≤ 31	6	≤ 32
≤ 57	6	≤ 63	7	≤ 64
≤ 120	7	≤ 127	8	≤ 128

Código de Hamming: Exercícios

1) Qual o Código de Hamming (distância mínima 3) com paridade par que representa a cadeia de informação 0101?

2) Se os bits de paridade nas posições 1, 2 e 8 indicam erro, qual bit está errado?

Código de Hamming: Exercícios

1) Qual o Código de Hamming (distância mínima 3) com paridade par que representa a cadeia de informação 0101?

➔ Comece construindo o código da direita para a esquerda, preenchendo os bits de informação e saltando os de paridade

➔ Preencha os bits de paridade usando a regra de abrangência previamente apresentada

7:111	6:110	5:101	**4**:100	3:011	**2**:010	**1**:001
d4	d3	d2	p3	d1	p2	p1
0	1	0	1	1	0	1

2) Se os bits de paridade nas posições 1, 2 e 8 indicam erro, qual bit está errado?

➔ 1+2+8 = 11

Gerador do Código de Hamming

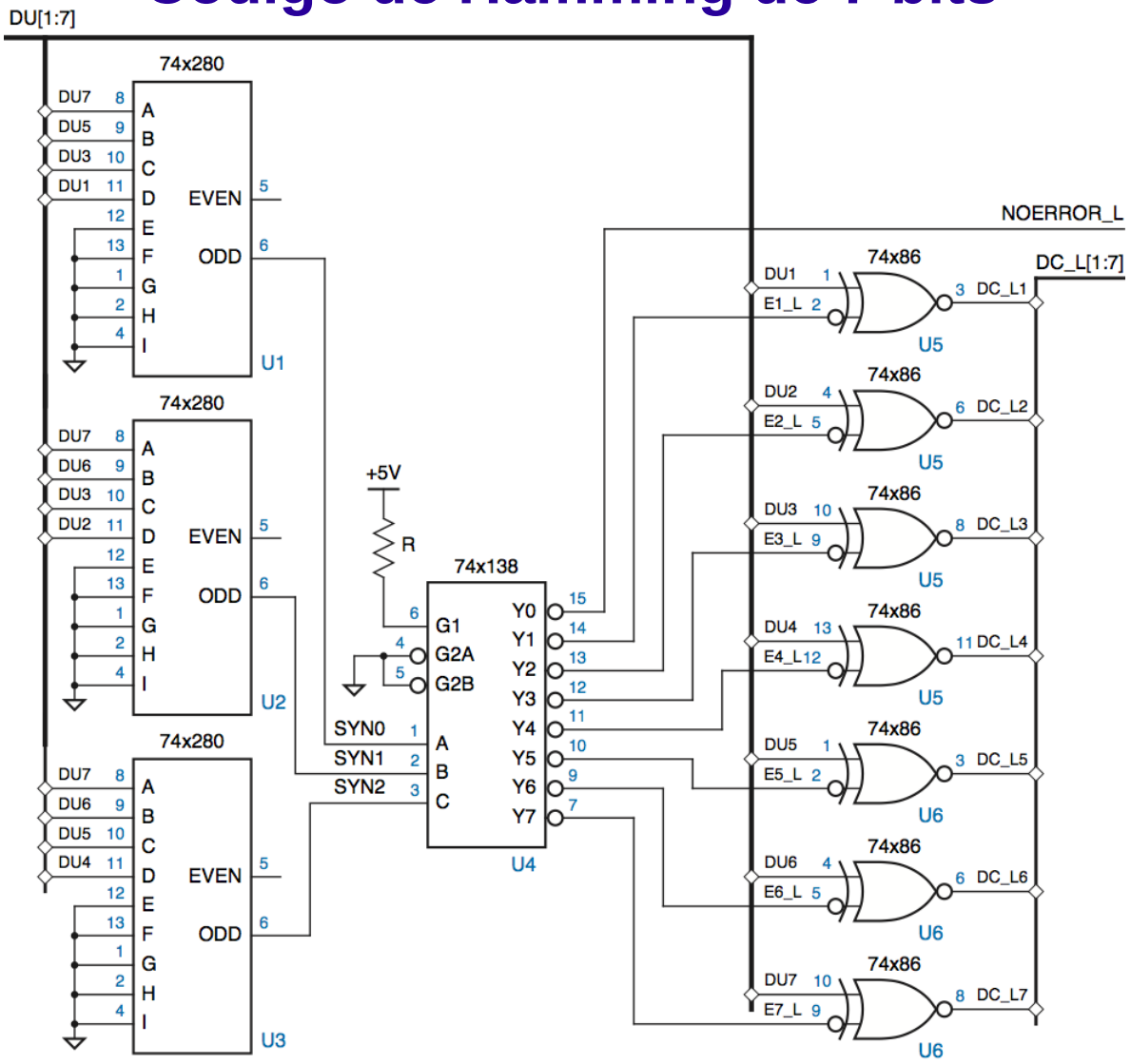
- **Código de Hamming:**
  - Basta fazer a interconexão correta entre os bits de entrada que entram na composição de cada bit de paridade

posições:	7:111	6:110	5:101	4:100	3:011	2:010	1:001
paridade sobre:				**5,6,7**		**3,6,7**	**3,5,7**

Posição do erro	Bits de paridade afetados	Posição do erro	Bits de paridade afetados
7	4+2+1 = 7	3	2+1 = 3
6	4+2 = 6	2	2
5	4+1 = 5	1	1
4	4 = 4		

Hamming: Detector/Corretor de Erros

Código de Hamming de 7 bits



Código de Hamming com Distância 4

- Distância 3 pode ser **estendida para distância 4**: basta adicionar um **bit de paridade (de redundância) calculado sobre todos os bits**
  - Usualmente o bit extra fica na posição 0.
  - Obtém-se: (c = 1, d = 1), ou então (c=0, d=3)

posições:	7:111	6:110	5:101	4:100	3:011	2:010	1:001	0
paridade sobre:				**5,6,7**		**3,6,7**	**3,5,7**	1…7

- Exemplo, bits de informação 1001, paridade par:

7:111	6:110	5:101	4:100	3:011	2:010	1:001	0
1	**0**	**0**	**1**	**1**	**0**	**0**	**1**

Código de Hamming: Exercícios

Qual o Código de Hamming com distância mínima 4 com paridade ímpar que representa os bits de informação 0101?

Liste as palavras do código de Hamming com 1 bit de informação.

Defina os grupos de paridade para um código de Hamming com 11 bits de informação.

CRC – Cyclic Redudancy Check

- Cadeias de bits representam **polinômios** d(x) com coeficientes 0 ou 1 – elementos de GF(2)
- $1010 = 1x^3 + 0x^2 + 1x + 0$
- Fixado um **polinômio gerador** p(x) de grau n, os bits de redundância representam o resto r(x) da divisão de $d(x).x^n$ por p(x).
- r(x), escrito com n bits, é concatenado à direita de d(x) para formar o código $d(x).x^n + r(x)$
- Todas operações no corpo finito de **Gallois**, onde 1+1=0 e 0-1=1, sem carry.
- **Exemplo**: p(x) = 101 e d(x)=11101, r(x)=?

CRC – Cyclic Redudancy Check

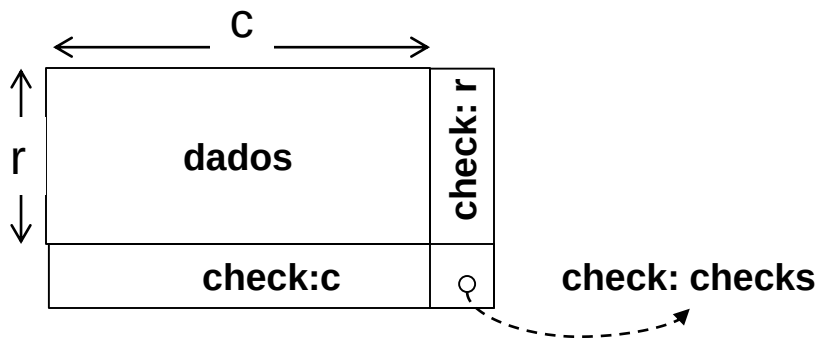
- Erro é detectado se a divisão do código por p(x) deixar resto diferente de zero.
- $d(x).x^n + r(x)$ dividido por p(x) deve deixar resto r(x)+r(x)=0.
- Exemplo, r(x)=1110111, p(x)=101.
- Simples implementação com XOR.
- Detecta erros em até n bits seguidos (comuns em discos e transmissões).

Código CRC: Exercício

Com o polinômio gerador p(x)=111, construa o código CRC para os bits de informação 101011. Em seguida, mostre que o código gerado é válido (sem erros).

Códigos Bidimensionais

- Bits são conceitualmente organizados em uma matriz, c colunas e r linhas.
- Usamos um código para acrescentar redundância a cada linha, e um código para colunas:



- Se os códigos tem distância mínima d_r (linhas) e d_c (colunas), qual a distância mínima resultante é $d_r d_c$
- Bit do canto pode ser de qualquer código.

Códigos Bidimensionais

- No caso mais simples, podemos usar paridade para colunas e linhas.
- Se uma coluna (ou linha) inteira é perdida, pode ser recuperada com os bits de redundância.
- Utilização no esquema RAID (Redundant Array of Inexpensive Disks):
  - n HDs de informação, 1 HD de paridade.
  - A perda de um HD inteiro é recuperável!.

Códigos Bidimensionais: Exercício

Usando código de paridade para linhas e colunas, mostre o código bidimensional para os bits de informação 101010101

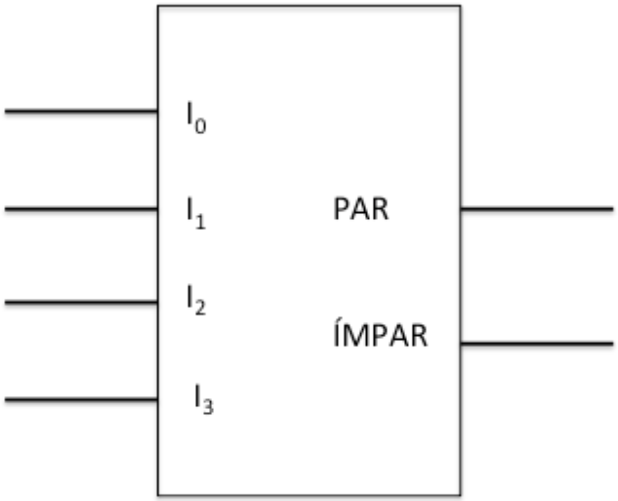
Suponha que, em uma transmissão dos 16 bits, a primeira linha de bits foi perdida. Mostre como recuperá-la.

Mostre como construir um código com distância mínima igual a 6 para 4 bits de informação. Liste as palavras válidas do código.

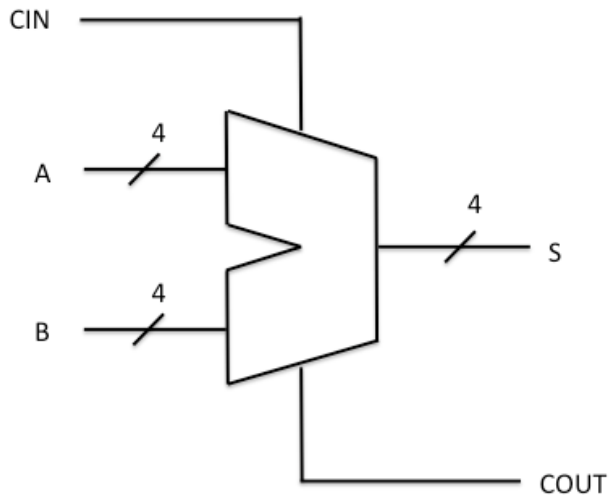
Qual código de distância 4 têm mais bits de informação por bit redundante, um bidimensional ou Hamming?

Exercício (PREC 2016)

- Seja A um número binário de 4 bits. Projete um circuito que calcule $A - 1_{10}$ caso A tenha paridade par e $A - 2_{10}$ caso A tenha paridade ímpar. A subtração deve ser calculada em Complemento de 2. Utilize (somente!) um gerador de paridade de 4 bits (Figura 1) e um somador completo de 4 bits (Figura 2).



Gerador de Paridade



Somador Completo de 4 bits

Exercício (PSUB 2017)

- Um sistema usa códigos de Hamming para corrigir erros de transmissão. Em um certo momento, deseja-se enviar a sequência de bits “1010”. Responda:

  a) Qual o número mínimo de bits que serão necessários para representar a palavra de código resultante caso a distância de código desejada seja 3? Qual a palavra de código resultante, considerando paridade par?

  b) Qual o número mínimo de bits que serão necessários para representar a palavra de código resultante caso a distância de código desejada seja 4? Qual a palavra de código resultante, considerando paridade par?

  c) Suponha que seja usado o código com distância 3, como no item (a), que a palavra recebida tenha o número de bits determinado naquele item e que todos esses bits sejam 1s exceto pelo bit mais significativo, que tem valor 0. Por exemplo, se sua resposta no item (a) foi que são necessários 4 bits para representar a informação, então a palavra recebida foi “0111”. Essa palavra código é válida? Caso não seja, para qual palavra código ela será corrigida de acordo com o método de correção de Hamming?

Exercício (PSUB 2017)

- Um sistema usa códigos de Hamming para corrigir erros de transmissão. Em um certo momento, deseja-se enviar a sequência de bits “1010”. Responda:

a) Qual o número mínimo de bits que serão necessários para representar a palavra de código resultante caso a distância de código desejada seja 3? Qual a palavra de código resultante, considerando paridade par?

7 bits	7	6	5	4	3	2	1
Palavra:	1	0	1	**0**	0	**1**	**0**
	1	0	1	**0**			
	1	0			0	**1**	
	1		1		0		**0**
Também aceito:	1	0	1	0	**0**	**1**	**0**
							paridade

Exercício (PSUB 2017)

- Um sistema usa códigos de Hamming para corrigir erros de transmissão. Em um certo momento, deseja-se enviar a sequência de bits “1010”. Responda:

b) Qual o número mínimo de bits que serão necessários para representar a palavra de código resultante caso a distância de código desejada seja 4? Qual a palavra de código resultante, considerando paridade par?

7 bits	7	6	5	4	3	2	1	0
Palavra:	1	0	1	**0**	0	**1**	**0**	**1**
	1	0	1	**0**				
	1	0			0	**1**		
	1		1		0		**0**	
Também aceito:	1	0	1	0	**0**	**1**	**0**	**1**
								paridade

Exercício (PSUB 2017)

c) Suponha que seja usado o código com distância 3, como no item (a), que a palavra recebida tenha o número de bits determinado naquele item e que todos esses bits sejam 1s exceto pelo bit mais significativo, que tem valor 0. Por exemplo, se sua resposta no item (a) foi que são necessários 4 bits para representar a informação, então a palavra recebida foi “0111”. Essa palavra código é válida? Caso não seja, para qual palavra código ela será corrigida de acordo com o método de correção de Hamming?

7 bits	7	6	5	4	3	2	1
Palavra:	0	1	1	**1**	1	**1**	**1**
	0	1	1	**0 (erro)**			
	0	1			1	**0 (erro)**	
	0		1		1		**0 (erro)**

Erro na posição 4+2+1 = 7 ➔ correção para 1111111

Exercício (PREC 2017)

- Deseja-se implementar um sistema que tenha suporte a números de 0 a 7, usando um código numérico. Preencha a seguinte tabela de códigos para cada um dos digitos possíveis, usando o código indicado na primeira linha da tabela.

Dígito Decimal	Bits correspondentes a **código BCD**	Bits para **código Gray**	**Bit de paridade par** para código BCD	**Bit de paridade par** para código Gray
0	000			
1				
2				
3				
4				
5				
6				
7				

Exercício (PREC 2017)

- Deseja-se implementar um sistema que tenha suporte a números de 0 a 7, usando um código numérico. Preencha a seguinte tabela de códigos para cada um dos digitos possíveis, usando o código indicado na primeira linha da tabela.

Dígito Decimal	Bits correspondentes a **código BCD**	Bits para **código Gray**	**Bit de paridade par** para código BCD	**Bit de paridade par** para código Gray
0	000	000	0	0
1	001	001	1	1
2	010	011	1	0
3	011	010	0	1
4	100	110	1	0
5	101	111	0	1
6	110	101	0	0
7	111	100	1	1