Circuitos Lógicos

Profa. Grace S. Deaecto

Faculdade de Engenharia Mecânica / UNICAMP
13083-860, Campinas, SP, Brasil.
grace@fem.unicamp.br

Segundo Semestre de 2013

NOTA AO LEITOR

Estas notas de aula foram inteiramente baseadas nas seguintes referências :

- T. Floyd, "*Digital Fundamentals*", 10th Edition, Prentice Hall, 2009.
- R. J. Tocci, N. S. Widmer, G. L. Moss, "*Sistemas Digitais : Princípios e Aplicações*", Prentice-Hall, 2007.
- I. V. Iodeta, F. G. Capuano, "*Elementos de Eletrônica Digital*", Editora Érica, 2006.
- V. A. Pedroni, "*Circuit Design and Simulation with VHDL*", 2^{nd} Edition, MIT, 2010.

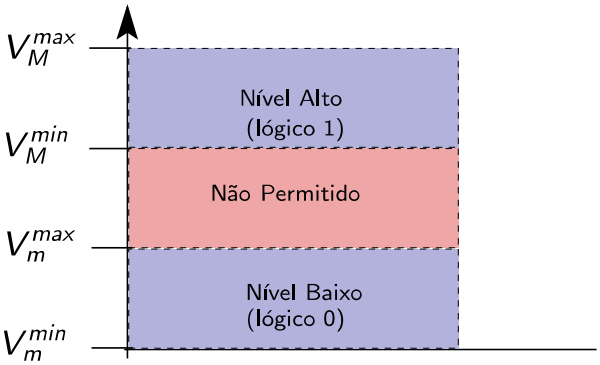
1. Introdução - Circuitos Lógicos
   - Sinais analógicos e digitais
   - Sistemas de numeração
   - Conversão numérica
   - Números com sinais
   - Operações com números binários
   - Códigos binários

Sinais analógicos e digitais

- Os circuitos eletrônicos podem ser divididos em duas categorias : circuitos analógicos e circuitos digitais.
- Circuitos analógicos processam grandezas contínuas no tempo.
  - A maior parte dos eventos da natureza são sinais analógicos tais como : temperatura, pressão, tempo, etc.
- Circuitos digitais processam grandezas discretas no tempo e na amplitude.
  - Apenas dois estados são possíveis 0 ou 1. Estes estados são chamados bits e, geralmente são uma representação codificada da informação original.

Sinais analógicos e digitais

- Os dois possíveis estados são representados por dois níveis de tensão alto ou baixo.



- Para circuitos digitais do tipo CMOS temos

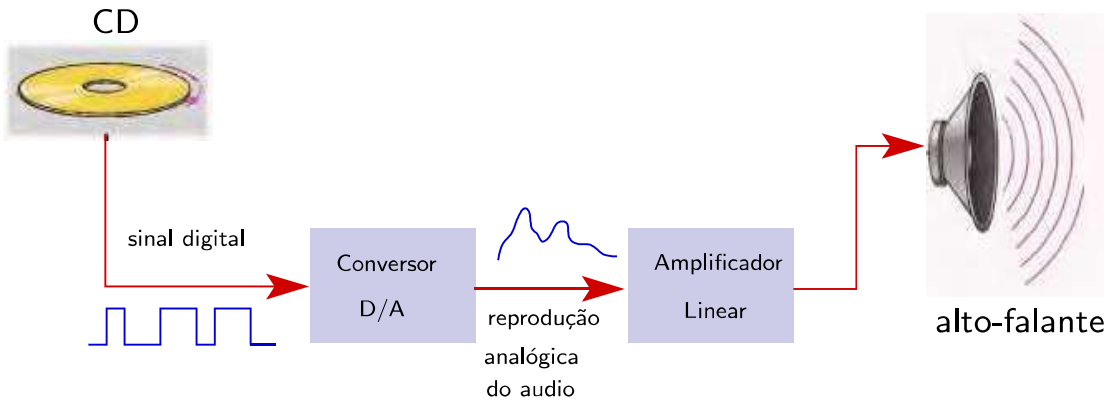
$$2 \leq v(t) \leq 3.3 \text{ [V]} \quad \Rightarrow \quad \text{Nível Alto}$$

$$0 \leq v(t) \leq 0.8 \text{ [V]} \quad \Rightarrow \quad \text{Nível Baixo}$$

Tensões entre 0.8 [V] e 2 [V] são inaceitáveis.

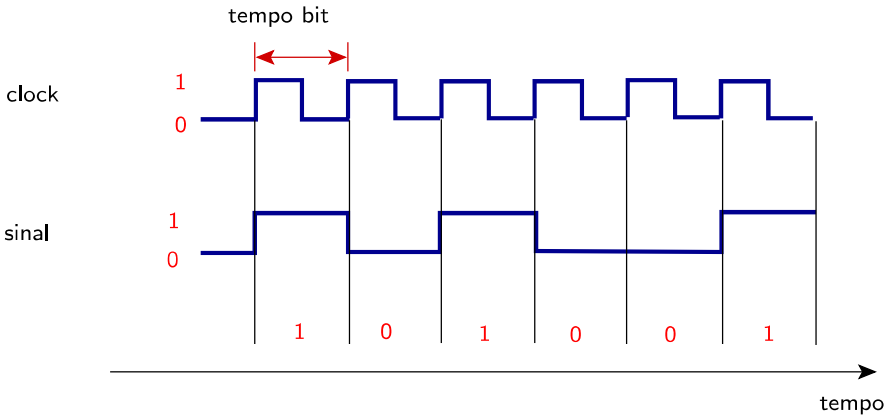
Sinais analógicos e digitais

- Exemplo de um sistema que utiliza métodos analógicos e digitais.



Sinais analógicos e digitais

- As formas de ondas encontradas em sistemas digitais são compostas por uma série de pulsos (trem de pulsos). A figura a seguir apresenta o diagrama de tempo de um sinal digital.



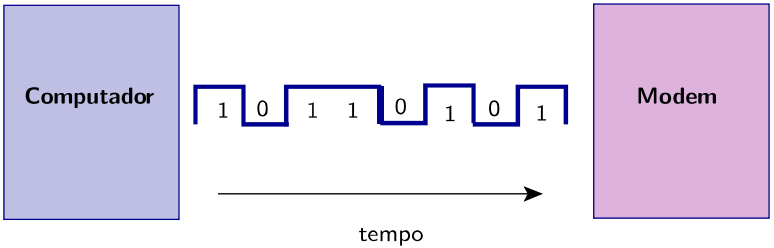
- O clock determina o sincronismo da forma de onda.

Transferência de dados

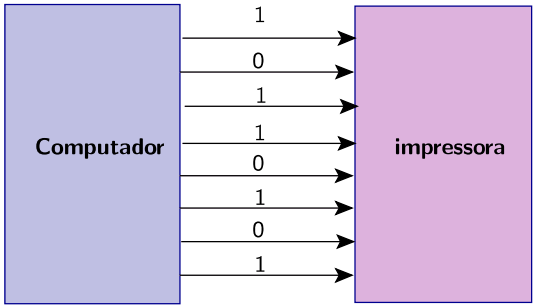
- Em sistemas digitais os dados representam um conjunto de bits que contém algum tipo de informação.
- A transmissão de dados pode ser realizada de duas maneiras diferentes : serial ou paralela.
  - transmissão serial : bit a bit (apenas uma linha de transmissão).
  - transmissão paralela : blocos de bits (número de linhas de transmissão é igual à quantidade de bits do bloco).

Transferência de dados

Transmissão Serial



Transmissão Paralela



Sistemas de numeração

- Os computadores e os circuitos digitais em geral utilizam o sistema de numeração binário para representar grandezas.
- Utilizando o sistema decimal seriam necessários 10 níveis de tensão diferentes, o que seria inviável.
- Como estamos habituados com o sistema de numeração decimal e, para entender com precisão o funcionamento de um circuito digital é importante relacionar ambos os sistemas de numeração.
- Representar uma grandeza no sistema binário requer um número maior de dígitos do que no caso decimal. Assim, para facilitar a manipulação de números binários utilizam-se os sistemas de numeração octal e hexadecimal.

Sistemas de numeração

- O sistema decimal possui dez símbolos

$$0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9$$

É um sistema posicional de base 10 que nos permite representar qualquer número utilizando potências de 10.

$$(234.5)_{10} = \mathbf{2} \times 10^2 + 3 \times 10^1 + 4 \times 10^0 + \mathbf{5} \times 10^{-1}$$

- O sistema binário contém somente dois símbolos

$$0,\ 1$$

É um sistema posicional de base 2 que nos permite representar qualquer número utilizando potências de 2.

$$(1011.1)_2 = \mathbf{1} \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + \mathbf{1} \times 2^{-1}$$

Sistemas de numeração

- O sistema hexadecimal possui dezesseis símbolos

$$0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ A,\ B,\ C,\ D,\ E,\ F$$

É um sistema posicional de base 16 que nos permite representar qualquer número utilizando potências de 16.

$$(356)_{16} = \mathbf{3} \times 16^2 + 5 \times 16^1 + \mathbf{6} \times 16^0$$

- O sistema octal contém oito símbolos

$$0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7$$

É um sistema posicional de base 8 que nos permite representar qualquer número utilizando potências de 8.

$$(54)_8 = \mathbf{5} \times 8^1 + \mathbf{4} \times 8^0$$

Ambos são formas compactas de representar sequências de bits!

Sistemas de numeração

- O dígito com a maior potência da base é denominado Most Significant Digit (MSD)
- O dígito com a menor potência da base é denominado Least Significant Digit (LSD)
- No sistema binário temos

  sequência de 8 bits = 1 byte
  sequência de 4 bits = 1 nibble

- Palavra é um conjunto de bits que representa uma certa informação.

Contagem

Decimal	Binário	Hexadecimal	Octal
0	0000	0	0
1	0001	1	1
2	0010	2	2
⋮	⋮	⋮	⋮
7	0111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17
16	10000	10	20

Contagem

- Considere um número com n dígitos representado na base $b \in \{2, 8, 10, 16\}$. Para a base b podemos contar

$$b^n \text{ números diferentes}$$

- O maior número sempre será igual a

$$b^n - 1 \text{ no sistema decimal}$$

Contagem

Exemplos : Qual o maior valor decimal que podemos contar com

- 10 dígitos na base 2

$$2^{10} - 1 = (1023)_{10}$$

- 3 dígitos na base 10

$$10^3 - 1 = (999)_{10}$$

- 3 dígitos na base 16

$$16^3 - 1 = (4095)_{10}$$

- 4 dígitos na base 8

$$8^4 - 1 = (4095)_{10}$$

Conversão numérica

A seguir será apresentado técnicas de conversão entre bases numéricas.

- Base $b \in \{2, 8, 16\}$ para base decimal - Basta somar os pesos de cada dígito. Exemplos :

$$\begin{array}{rcccl} (11011)_2 & = & 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0 & = & (27)_{10} \\ (16)_8 & = & 1 \times 8^1 + 6 \times 8^0 & = & (14)_{10} \\ (AC)_{16} & = & 10 \times 16^1 + 12 \times 16^0 & = & (172)_{10} \end{array}$$

- Base decimal para base $b \in \{2, 8, 16\}$ - Neste caso, existem dois procedimentos a serem adotados. Caso o número possua parte fracionária, esta deve ser separada da sua parte inteira cabendo a cada uma delas o procedimento a seguir.

Conversão numérica

- Para a parte inteira realizam-se divisões sucessivas do número decimal pela base b escolhida até que o quociente da divisão seja nulo. Os restos formam o número na base desejada, sendo que o primeiro obtido é o LSB e o último o MSB.

Exemplo : Converta $(27)_{10}$ para binário :

$$\begin{array}{rcrcrcl} 27/2 & = & 13 & , & resto & = & 1_{LSB} \\ 13/2 & = & 6 & , & resto & = & 1 \\ 6/2 & = & 3 & , & resto & = & 0 \\ 3/2 & = & 1 & , & resto & = & 1 \\ 1/2 & = & 0 & , & resto & = & 1_{MSB} \end{array}$$

$$(27)_{10} = (11011)_2$$

Conversão numérica

- Para a parte fracionária realizam-se multiplicações sucessivas da parte fracionária do número decimal pela base b escolhida até que a mesma seja nula. As partes inteiras da multiplicação formam o número na base desejada, sendo que a primeira parte inteira obtida é o MSB e a última o LSB.

Exemplo : Converta $(0.375)_{10}$ para binário :

$$\begin{array}{rcrcrcl} 0.375 \times 2 & = & 0.750 & , & p.\ inteira & = & \mathbf{0}_{MSB} \\ 0.750 \times 2 & = & 1.500 & , & p.\ inteira & = & 1 \\ 0.500 \times 2 & = & 1.000 & , & p.\ inteira & = & \mathbf{1}_{LSB} \end{array}$$

$$(0.375)_{10} = (0.011)_2$$

Usando o mesmo raciocínio temos

$$(27.375)_{10} = (1B.6)_{16}$$

Conversão numérica

- Conversão de binário para hexadecimal - Basta agrupar os bits em grupo de quatro e converter cada grupo.

Exemplo :

$$(1001\ 0111\ 1011\ 0101)_2 = (97B5)_{16}$$

- Conversão de hexadecimal para binário - A conversão se faz dígito a dígito.

Exemplo :

$$(4C)_{16} = (0100\ 1100)_2$$

Para a base octal a ideia é semelhante, mas fazendo-se agrupamentos de 3.

Regras básicas da adição e da subtração

- As quatro regras básicas da adição são :

$$\begin{aligned} 0+0 &= 0 \\ 0+1 &= 1 \\ 1+0 &= 1 \\ 1+1 &= 10 \quad , \quad 1 \text{ é chamado carry in ou vai um} \end{aligned}$$

- As quatro regras básicas da subtração são :

$$\begin{aligned} 0-0 &= 0 \\ 0-1 &= 1 \quad , \quad \text{com empréstimo de 1 da coluna seguinte} \\ 1-0 &= 1 \\ 1-1 &= 0 \end{aligned}$$

As operações de multiplicação e divisão são realizadas como nos decimais.

Números com sinais

Para números com sinais, em uma palavra de 1 byte, o primeiro bit representa o sinal do número, sendo 0 se positivo e 1 se negativo. Existem duas maneiras de se representar números com sinais :

- Forma sinal-magnitude : Neste caso o primeiro bit é o bit do sinal e os restantes representam a magnitude.

  $(+35)_{10} = (00100011)_2 \quad , \quad (-35)_{10} = (10100011)_2$

  Para converter em decimal, basta converter a magnitude e verificar o primeiro bit para atribuição do sinal.

  $(10100011)_2 = -(1 \times 2^5 + 1 \times 2^1 + 1 \times 2^0) = (-35)_{10}$

Números com sinais

- Forma complemento de 2 : Neste caso, quando o número for negativo, realizamos o complemento de 2 de seu correspondente positivo. O complemento de 2 é obtido trocando-se 0s por 1s e vice-versa e somando-se 1 ao resultado.

$$(+35)_{10} = (00100011)_2 \quad , \quad (-35)_{10} = (11011101)_2$$

Para realizar a conversão para decimal atribui-se valor negativo à parcela do peso do bit de sinal.

$$(11011101)_2 =$$
$$-1 \times 2^7 + 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 = (-35)_{10}$$

Números com sinais

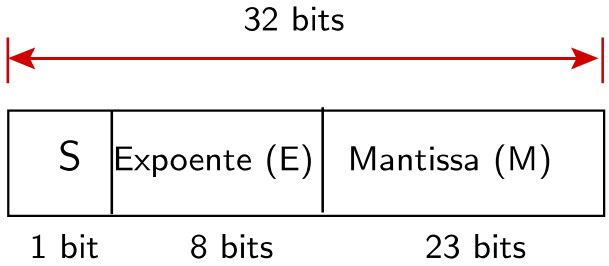
- Com a representação em complemento de 2 o intervalo de valores para números de n bits é

$$\text{Intervalo} : -(2^{n-1}) \; a \; (2^{n-1} - 1)$$

  Logo, para 8 bits podemos representar números de -128 a 127.
- Note que quanto maior é o número, maior é a quantidade de bits necessária para a sua representação.
- Para representar números muito grandes ou muito pequenos utiliza-se a representação em ponto flutuante definida pelo padrão IEEE-754.
- Este padrão possui três formatos : precisão única (32 bits), dupla precisão (64 bits) e precisão estendida (80 bits).

Números com sinais

- Para precisão única, a estrutura é a seguinte :



- S é o bit do sinal
- E é o expoente "polarizado" que é obtido somando-se 127 ao expoente verdadeiro. A ideia é de permitir um intervalo grande de números sem a necessidade de reservar um bit de sinal para o expoente. O intervalo de valores para o expoente é de -126 a +128.
- M é a mantissa e representa a parte fracionária do número. O dígito mais a esquerda é o mais significativo.

Números com sinais

- Duas exceções : O número 0.0 é representado por todos os bits iguais a 0 e, ∞ é representado por 1s nos bits do expoente e 0s nos bits da mantissa.
- O número escrito em ponto flutuante é dado por :

$$\text{Número} : (-1)^S \times (1 + M) \times 2^{(E-127)}$$

Exemplo : Represente o número positivo 1011010010001 em ponto flutuante. Primeiramente, escrevemos a igualdade

$$1011010010001 = 1.011010010001 \times 2^{12}$$

e, portanto, temos

$$\begin{aligned} S &= 0 \\ E &= 12 + 127 = (139)_{10} = (10001011)_2 \\ M &= (01101001000100000000000)_2 \end{aligned}$$

Operação de Adição

- Dois números positivos :

$$\begin{array}{r} 00000110 \\ +00000101 \\ \hline 00001011 \end{array} \quad , \quad \begin{array}{r} 6 \\ +5 \\ \hline 11 \end{array}$$

- Um positivo e outro negativo, três situações :

$$\begin{array}{r} 00001111 \\ +11111010 \\ \hline 1\ 00001001 \end{array} \quad , \quad \begin{array}{r} 15 \\ -6 \\ \hline 9 \end{array} \quad \text{o carry é descartado}$$

$$\begin{array}{r} 00010000 \\ +11101000 \\ \hline 11111000 \end{array} \quad , \quad \begin{array}{r} 16 \\ +-24 \\ \hline -8 \end{array}$$

$$\begin{array}{r} 11111011 \\ +11110111 \\ \hline 1\ 11110010 \end{array} \quad , \quad \begin{array}{r} -5 \\ +-9 \\ \hline -14 \end{array} \quad \text{o carry é descartado}$$

Operação de Adição

- Condição de overflow : Pode ocorrer somente quando os dois números são positivos ou ambos são negativos. A situação de overflow ocorre quando o bit do sinal resultante for diferente do bit dos números somados.

$$\begin{array}{r} 01111101 \\ +00111010 \\ \hline 10110111 \end{array} \quad , \quad \begin{array}{r} 125 \\ +58 \\ \hline 183 \end{array}$$

- Operação de subtração : É um caso especial da adição, por exemplo, para calcular $z = x - y$ basta fazer $z = x + (-y)$ e aplicar as regras da adição.

Operação de Multiplicação

- A multiplicação é realizada com os números na sua forma verdadeira (sem complemento de dois em caso de número negativo).
- Quando ambos os números são positivos ou ambos negativos (faz-se a conversão para a forma verdadeira caso se encontrem em complemento de dois) basta realizar a multiplicação como no caso decimal. Ao bit do sinal é atribuído 0.
- Quando um é negativo, na sua forma verdadeira, e o outro positivo, faz-se o complemento de dois do resultado e atribui-se 1 ao bit de sinal.

Operação de Multiplicação

- *Exemplo* : : Deseja-se multiplicar 01010011 por 11000101. Como os sinais são diferentes o bit do sinal é 1. Realiza-se a multiplicação de

$$01010011 \times 00111011 = 1001100100001$$

Uma vez que o sinal é negativo, deve-se obter o complemento de dois do número e adicionar o bit de sinal

Resultado : 1 0110011011111

- Operação de divisão : O raciocínio é o mesmo realizado na multiplicação, ou seja, considera-se os números na sua forma verdadeira e analisa-se o sinal do resultado.

Código BCD

Codificar - significa transformar um conjunto de símbolos conhecidos em códigos.

- No código BCD (Binary Coded Decimal) cada dígito decimal é representado por seu equivalente em binário sendo necessários 4 bits para cada dígito decimal de 0 a 9.
- A ideia é criar um ambiente mais amigável tendo em vista a nossa familiaridade com os números decimais.
- Lembrando que com 4 bits podemos representar valores decimais de 0 a $(2^4 - 1)_{10} = (15)_{10}$, os números

$$\underbrace{1010}_{(10)_{10}}, \underbrace{1011}_{(11)_{10}}, \underbrace{1100}_{(12)_{10}}, \underbrace{1101}_{(13)_{10}}, \underbrace{1110}_{(14)_{10}}, \underbrace{1111}_{(15)_{10}}$$

não existem na codificação BCD.

Código BCD

- Embora sejam representados pelos dígitos 0 e 1, esta codificação é diferente da binária.
- A conversão de decimal para BCD é simples. Basta substituir cada dígito decimal pela sua representação binária de 4 bits.
- Obviamente, a conversão de BCD para decimal é feita diretamente para grupos de quatro bits.

Exemplo :

$$\begin{aligned}(137)_{10} &= (10001001)_2 \\ &= (\underbrace{0001}_{1}\ \underbrace{0011}_{3}\ \underbrace{0111}_{7})_{BCD}\end{aligned}$$

Código Gray

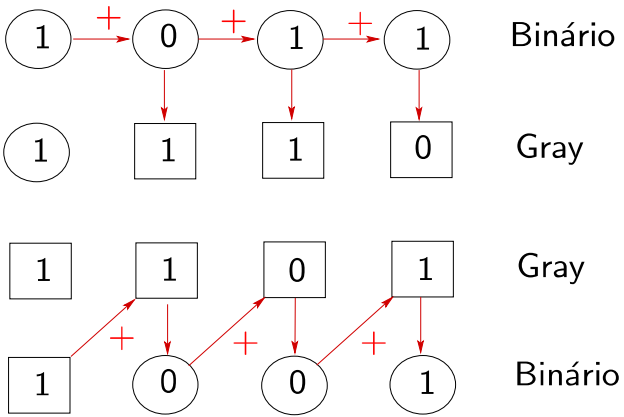
- O código Gray não é posicional e nem aritmético.
- Sua principal característica é que apenas um bit é alterado entre dois números sucessivos na sequência dificultando eventuais erros de codificação.

Decimal	Binário	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

Decimal	Binário	Gray
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Código Gray

- Conversão de binário para código Gray - O MSB permanece. Da esquerda para a direita somam-se os bits adjacentes descartando-se o carry in (vai um).
- Conversão de código Gray para binário - O MSB permanece. Soma-se a cada dígito binário gerado o bit do código Gray da próxima posição adjacente descartando-se o carry (vai um).



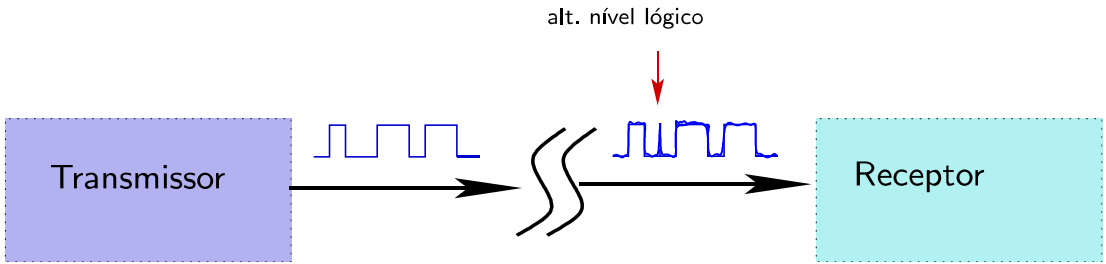
Códigos alfanuméricos

- Os códigos alfanuméricos apresentam não somente números mas também letras do alfabeto e símbolos (comandos, acentos, caracteres especiais tais como %, &, @).
- O código ASCII (American Standard Code for Information Interchange) é o mais usado. Possui 128 caracteres representados por números de 7 bits.
- A tabela seguinte mostra alguns códigos em ASCII.

Nome	Decimal	Binário	Hexadecimal
A	65	1000001	41
a	97	1100001	61
%	37	0100101	25
&	38	0100110	26
:	58	0111010	3A

Detecção de erros - Método da paridade

- Quando a informação é transmitida de um dispositivo para outro podem ocorrer erros durante a transmissão.



- Um dos métodos para a detecção de erros é o método da paridade.

Detecção de erros - Método da paridade

- Neste método, convenciona-se o tipo de paridade, par ou ímpar, a ser adotada e adiciona-se um bit (bit de paridade) ao dado a ser transmitido.

- Para paridade par, por exemplo, se o dado contiver um número ímpar de 1s então o bit de paridade será igual a 1 e, 0 caso contrário. Desta forma, toda a informação transmitida terá sempre um número par de 1s.

$$1\ \underbrace{1000011}_{\text{ímpar}}\ ,\quad 0\ \underbrace{1000001}_{\text{par}}$$

Detecção de erros - Método da paridade

- Para verificar se o receptor recebeu o dado corretamente basta checar se o número de 1s recebidos é par. Caso contrário, ocorreram erros durante a transmissão , e o dado deve ser retransmitido.

- Note que este método é eficiente para a detecção de erros de apenas um bit (ou um número ímpar de bits o que é pouco comum).