Vitor Amadeu Souza

Introdução a

Eletrônica Digital

com

VHDL

© 2023 by Cerne Tecnologia e Treinamento Ltda.

© 2023 by Vitor Amadeu Souza

Nenhuma parte desta publicação poderá ser reproduzida sem autorização prévia e escrita de **Cerne Tecnologia e Treinamento Ltda.** Este livro publica nomes comerciais e marcas registradas de produtos pertencentes a diversas companhias. O editor utiliza as marcas somente para fins editoriais e em benefício dos proprietários das marcas, sem nenhuma intenção de atingir seus direitos.

Julho de 2023

Direitos reservados por:

Cerne Tecnologia e Treinamento Ltda

Produção: *Cerne Tecnologia e Treinamento*

E-mail da Empresa: *cerne@cerne-tec.com.br*

Home Page: *www.cerne-tec.com.br.com.br*

Atendimento ao Consumidor: *sac@cerne-tec.com.br*

Contato com o Autor: *vitor@cerne-tec.com.br*



FEITO NO BRASIL

“Conhece-te a ti mesmo, torna-te consciente de tua ignorância e será sábio.”

Sócrates

Kits Didáticos e Gravadores da Cerne Tecnologia

A Cerne tecnologia têm uma linha completa de aprendizado para os microcontroladores da família PIC, 8051, Holtek, dsPIC, ARM etc. Veja os detalhes de cada um nas figuras abaixo:



Kit Cerne CPLD

CPLD ALTERA
Botões
Leds
Gravação ICSP
Display 7 Segmentos
E muito mais!

Uma linha completa de componentes para o desenvolvimento de seus projetos eletrônicos como displays, microcontroladores, botões, leds, cristais etc. Visite a nossa página na Internet, no endereço **www.cerne-tec.com.br** e conheça melhor nossos serviços e produtos.

Cerne
Conhecimento para o Desenvolvimento

www.cerne-tec.com.br

Sumário

Introdução **13**

Capítulo I - Natureza Analógica e Digital **14**

1. Analógico x Digital 14
2. Aspectos dos circuitos digitais 14
3. Processamento Digital 16
4. Representação binária 17
5. Taxa de comunicação 18
6. Tempo de subida e descida 19
7. Exercícios 21

Capítulo II - Sistemas de Numeração **24**

1. Sistema Decimal 24
2. Sistema Binário 27
   - 2.1 Aglomerado de bits 30
   - 2.2 Conversão de binário para decimal 30
   - 2.3 Conversão de decimal para binário 32
3. Sistema Octal 33
   - 3.1 Conversão Octal para Binário 35
   - 3.2 Conversão Octal para Decimal 36
4. Sistema Hexadecimal 36
   - 4.1 Conversão de hexadecimal para binário 38
   - 4.2 Conversão Hexadecimal para Decimal 38
5. Código BCD 39
6. Código Gray 42
7. Código Alfanumérico 43
8. Comparativo dos sistemas apresentados 44
9. Paridade 45
10. Checksum 47
11. Recursos computacionais 48
12. Representação Sinal Magnitude 49
13. Representação Complemento 2 53
14. Exercícios 55

Capítulo III - Portas Lógicas 73

1. Software LogicSim 73
2. Porta Lógica NOT 73
3. Porta Lógica AND 78
4. Porta Lógica OR 81
5. Porta Lógica NAND 84
6. Porta Lógica NOR 86
7. Porta Lógica XOR 89
8. Porta Lógica XNOR 92
9. Portas lógicas com mais entradas 95
10. Tipos de saída 97
11. Fan-out 98
12. Tipos de encapsulamento 99
13. Circuitos através de expressões booleanas 100
14. Expressões booleanas através de circuitos digitais 102
15. Universalidade da porta NAND 103
16. Universalidade da porta NOR 105
17. Complexidade de Circuitos Digitais 106
18. Família de Circuitos Digitais TTL 107
19. Família de Circuitos Digitais CMOS 108
20. Exercícios 110

Capítulo IV - Álgebra de Boole 131

1. Introdução a Álgebra Booleana 131
2. Propriedades Comutativa, Associativa e Distributiva 131
3. Elemento Neutro 131
4. Teorema para uma variável 132
5. Teorema para duas variáveis 133
6. Teorema de DeMorgan 133
7. Método da Soma de Produtos 134
8. Método de Produto das Somas 134
9. Mapa de Karnaugh 140
10. Condições irrelevantes (Don´t Care) 157
11. Exercícios 160

Capítulo V - Circuitos Combinacionais 177

1.Introdução aos Circuitos Combinacionais 177
2.Decodificador de 2 bits 177

3.Detector de números iguais 182
4.Decodificador de Display de 7 segmentos para 2 dígitos 188
5.Desafios 192
Capítulo VI - Circuitos Sequenciais 193
1.Introdução aos Circuitos Sequenciais 193
2.Flip-flop assíncrono RS com porta NAND 193
3.Flip-flop assíncrono RS com porta OR 199
4.Flip-flop com entrada Preset e Clear 203
5.Flip-flop síncrono RS 206
6.Flip-flop JK 214
7.Flip-flop D 219
8.Flip-flop T 225
9.Latch D transparente 229
10.Contadores 234
11.Exercícios 239
Capítulo VII - Circuitos Lógicos Aritméticos 248
1. Introdução aos Circuitos Aritméticos 248
2. Adição binária de números positivos 248
3. Adição e Subtração Binária em Complemento 2 249
4. Condição de overflow 251
5. Full-adder de 1 bit 252
6. Somador completo de 2 bits 254
7. Somador completo de 4 bits 256
8.Subtrator completo de 2 bits 257
9. Circuito Integrado 74HC283 259
10. Somador de 8 bits com 74HC283 259
11.Exercícios 261
Capítulo VIII - Contadores 263
1. Contador assíncrono de 1 bit 263
2.Contadores assíncrono de n bits 265
3.Contador assíncrono com reset 266
4.O CI 74HC393 268
5.Exercícios 269
Capítulo IX - Registradores 270
1. Registrador de deslocamento 270

2. Circuito SIPO (Serial In – Parallel Out) 270
3. Circuitos PISO (Parallel In – Serial Out) 275
4.Exercícios 276
Capítulo X - Decodificadores e Codificadores 278
1. Decodificadores 278
2. Decodificador 3 para 8 e o CI 74HC138 278
3. Decodificador BCD para Decimal e o CI 74HC42 282
4. Decodificador BCD para 7 Segmentos e o CI CD4511 285
5.Codificadores 287
6. Codificador 8 para 3 e o CI 74HC148 289
7. Exercícios 291
Capítulo XI - Mux, Demux e Comparadores 292
1. Multiplexadores 292
2.O CI 74HC151 295
3.Demultiplexadores 297
4.Comparadores e o CI 74HC85 299
5.Exercícios 303
Capítulo XII - Memórias 305
1. Memória Volátil e não Volátil 305
2. Endereçamento e dados de uma memória 305
3. Conectando a memória a CPU 312
4. Conectando mais memórias a CPU 312
5. Memória RAM 313
5.1 Memória SRAM 315
5.2 Memória DRAM 315
6. Memória ROM 316
7. Memória PROM 317
8. Memória EPROM 317
9. Memória EEPROM 319
10. Memória Flash 321
11. Complexidade e custo de memórias 321
12. Memória FIFO 322
13. Verificação de erros em memória 323

Capítulo XIII - Arquitetura de Computadores 325

1. Unidade Central de Processamento 325
2. Barramento de dados 326

Capítulo I – Introdução ao FPGA 330

1. A importância do FPGA 330
2. Conhecendo o FPGA 330
3. Programando o FPGA 332
4. Características da família Cyclone IV 332

Capítulo II – Ferramenta de desenvolvimento 337

1. Hardware 337
2. Software 338
3. Criando um projeto 339
4. Simulando o projeto 348
5. Gravando o FPGA 368

Capítulo III – Exemplos no Schematic 371

1. Introdução 371
2. Botões e Leds 371
3. Botão Invertido 372
4. Lógica OR 373
5. Lógica AND 374
6. Projeto de Alarme I 376
7. Projeto de Alarme II 377
8. Display de 7 segmentos 378
9. Comparador 379
10. Flip-Flop Tipo D 380
11. MUX I 381
12. MUX II 382
13. DEMUX I 383
14. Decoder I 384
15. Decoder II 385
16. Half-Adder 386
17. Full-Adder 388
18. Multiplicação 389
19. Contador 391
20. Contador com Código Gray 392

21. Somador full adder 74283 394
22. ULA 74382 396
Capítulo IV – VHDL 400

1. Introdução 400
2. Estrutura básica do VHDL 400
3. Comentários 401
4. Identificadores 402
5. Representação numérica em diversas bases 402
6. Números inteiros e reais 403
7. Representação de strings e caracteres 403
8. Tipos de dados 404
9. Constantes 405
10. Variáveis 405
11. Definindo faixas de valores 405
12. Conversão de tipos 406
13. Operador de atribuição 406
14. Operadores aritméticos 406
15. Operadores relacionais 406
16. Operadores lógicos 407
17. Operadores de deslocamento 407
18. Comando de decisão IF 407
19. Comando de decisão IF-ELSE 408
20. O comando de decisão CASE 409
21. O comando de loop FOR 409
22. O comando de loop WHILE 410
Capítulo V – Exemplos no VHDL 412

1. Introdução 412
2. Ligando um LED 413
3. Botão e Led 414
4. Botões e Leds 415
5. Lógica OR com botões 418
6. Lógica AND com botões 419
7. Lógica XOR com botões 420
8. Pisca-pisca 422
9. Contador binário 423
10. Contador de década 425

11. Display de 7 segmentos 426
12. Comparador de 4 bits 427
13. Lógica AND com Nibble 428
14. Lógica OR com Nibble 430
15. Lógica XOR com Nibble 431
16. Lógica NAND com Nibble 432
17. Lógica NOR com Nibble 433
18. Lógica XNOR com Nibble 434
19. Lógica AND com Byte 435
20. Lógica OR com Byte 437
21. Lógica XOR com Byte 438
22. Lógica NAND com Byte 440
23. Lógica NOR com Byte 441
24. Lógica XNOR com Byte 442
25. Somador com Nibble 444
26. Representação complemento 2 448
27. Subtrator com Nibble 450
28. Unidade Lógica Aritmética (ULA) 454
29. Multiplexador 470
30. MUX/DEMUX digital e analógico 471
31. Multiplexador 74HC157N 472
32. Multiplexador 74HC151N 474
33. Projeto de Multiplexador com 4 entradas 476
34. Projeto de Multiplexador com 8 entradas 479
35. Multiplexador 74HC4067 482
36. MUX/DEMUX CD4051 484
37. MUX/DEMUX CD4052 ..... 1 486
38. MUX/DEMUX CD4053 487
39. Projeto de Demultiplexador com 4 saídas 489
40. Codificador Decimal → BCD 492
41. Codificador Decimal → Stibitz ou excesso 3 495
42. Codificador Decimal → Gray 497
43. Codificador BCD → Gray 499
44. Decodificador 74LS139 502
45. Decodificador 74LS138 506
46. Decodificador CD4028 510
47. Decodificador SN7447 514

48. Flip-flop assíncrono RS com porta NAND 517
49. Flip-flop assíncrono RS com porta NOR 521
50. Flip-flop assíncrono RS com PRESET e RESET 525
51. Flip-flop síncrono RS 530
52. Flip-flop JK 536
53. FF tipo D rising edge 544
54. FF tipo D falling edge 547
55. FF tipo T rising edge 550
56. FF tipo T falling edge 552
57. Latch com EN=1 553
58. Latch com EN=0 557
59. Registrador de deslocamento 559
60. Contador 564
61. Temporizador 571
62. Biestável ou Multivibrador biestável 574
63. Monoestável ou Multivibrador monoestável 574
64. Astável ou Multivibrador astável 576
65. Máquina de estados 577
66. Memória RAM 62256 585
67. Operação de leitura (read cycle) 587
68. Operação de escrita (write cycle) 591
Capítulo VI – Exercícios com respostas 595
Referências 630

Introdução

Este livro apresenta de maneira prática e objetiva o universo dos circuitos digitais, passando inicialmente pela natureza do mundo analógico e digital, para em seguida explorar os sistemas digitais com o auxílio de softwares simuladores.

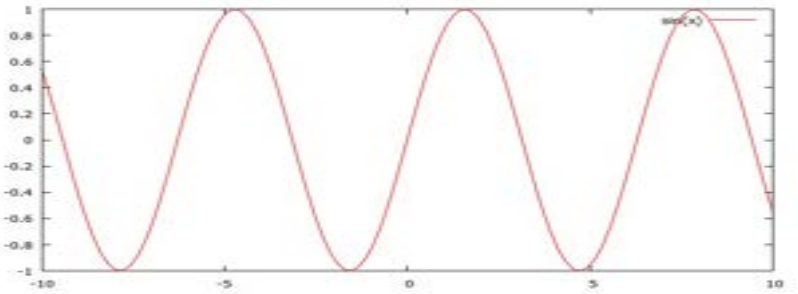
Após esta etapa são exploradas as portas lógicas, álgebra de boole, mapa de Karnaugh, circuitos combinacionais, circuitos sequenciais, circuitos aritméticos, registradores, multiplexadores, demultiplexadores, comparadores, memórias, arquitetura dos computadores dentre outros temas presentes no sumário da obra.

Na parte final do livro é estudado o FPGA e como aplicar o conteúdo até então apresentado de eletrônica digital programado em VHDL.

Capítulo I
Natureza Analógica e Digital

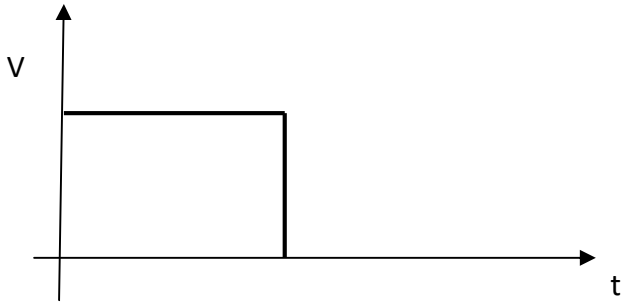
1. Analógico x Digital

Apesar de vivermos na era da evolução da tecnologia digital, cercados de computadores, tablets, notebooks etc, o mundo em si é analógico e não digital. Por exemplo, a temperatura, pressão, luminosidade são alguns exemplos de grandezas analógicas que nos cercam. Desta forma, podemos concluir que grandezas analógicas podem assumir valores contínuos que podem assumir um número infinito de valores em determinado intervalo de tempo, como apresentado na figura a seguir, sendo assim chamadas de variáveis contínuas:



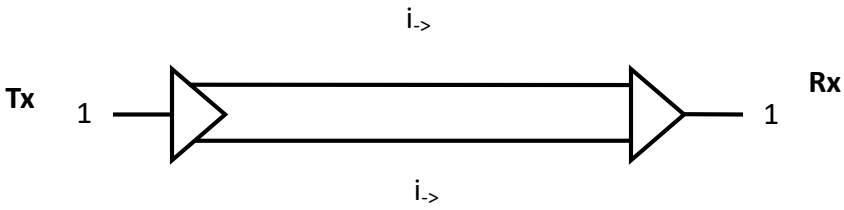
Como exemplos de circuitos analógicos temos os amplificadores de áudio, fontes de tensão, medição de temperatura etc.

Nos circuitos digitais trabalhamos apenas com dois níveis, chamados estes de 0 (false ou falso) e 1 (true ou verdadeiro), onde as variáveis neste caso são discretas, podendo assim obter apenas dois níveis em determinado intervalo de tempo, neste caso 0 ou 1. Estas variáveis são chamadas discretas e na próxima figura temos um exemplo demonstrando isso:

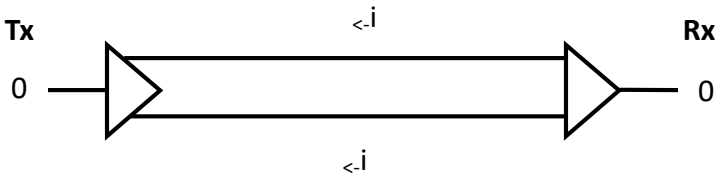


Como exemplos de circuitos digitais temos os computadores, câmeras digitais, os CD/DVDs, televisores digitais etc.

Existem várias formas de representar um sinal digital, onde a ideia básica é que existam dois níveis. Como exemplo, poderíamos ter com som e sem som; com luz e sem luz; com corrente e sem corrente etc. Ou seja, desde que haja dois níveis estáveis podemos representá-lo como sinal digital. Um bom exemplo disso é LVDS (Low Voltage Diferential Signalling – Sinal Diferencial de Baixa Tensão) onde temos em determinado sentido de corrente a representação do bit 1 e em outro sentido da corrente o bit 0. Observe a figura a seguir onde o bit 1 está sendo transmitido. Observe que a corrente flui em um sentido.



Já na figura a seguir, a corrente flui no sentido inverso, transmitindo assim o estado do bit 0.



2. Aspectos dos circuitos digitais

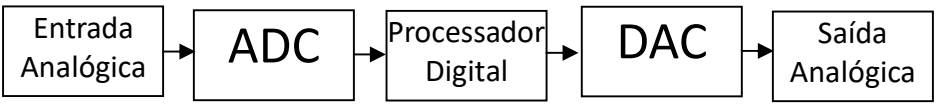
Como principais vantagens dos circuitos digitais em relação aos analógicos, poderíamos citar:

- Projeto mais fácil se comparado ao analógico;
- Facilidade para armazenar informação;
- É programável;
- Circuitos compactos;
- Imunidade a ruído;
- Confiabilidade de funcionamento.

Apesar dos circuitos digitais apresentarem inúmeras vantagens, não podemos esquecer que os mesmos são projetados com circuitos analógicos como transistores, diodos, resistores onde estes estão sujeitos a variações de temperatura, tensão, umidadetc.

3. Processamento Digital

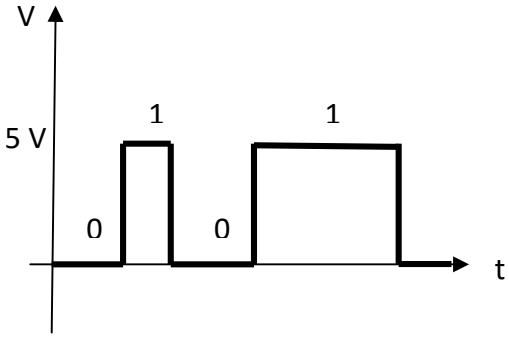
Observamos assim a necessidade de converter uma grandeza analógica para digital, de forma a processarmos a mesma. Isso é feito através de um conversor analógico para digital (ADC – Analog to Digital Converter) que permite realizar este tipo de conversão. Em seguida, já com o sinal digital, um Processador Digital, que pode ser um microcontrolador, FPGA, Computador, DSP (Digital Signal Processor – Processador Digital de Sinais) etc fazem o processamento da informação onde esta poder ser novamente convertida em uma grandeza analógica através de um conversor digital para analógico (DAC – Digital to Analog Converter) como exemplifica o quadro a seguir:



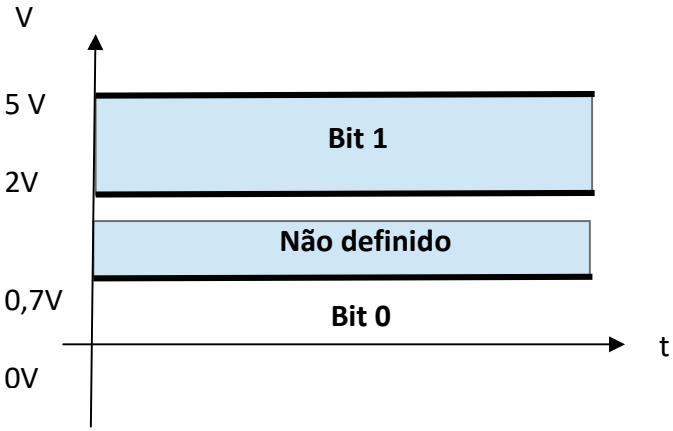
Nos novos projetos eletrônicos, temos cada vez mais a tendência de converter um sinal analógico para a sua forma digital para o processamento digital onde se posterga ao máximo para o final do circuito a nova conversão de digital para analógico.

4. Representação binária

A representação na forma binária nos leva a dois estados possíveis, sendo estes chamados de 1 (verdadeiro ou true) e 0 (falso ou false). Desta forma, há várias formas de representar este estado, por exemplo, através de um nível de tensão, onde o 1 normalmente está associado a tensão de 5V e o 0 a tensão de 0V. Observe o gráfico a seguir:



Pelo fato da representação ser feita em níveis digitais, a uma faixa (range) em que um sinal pode ser interpretado. Como exemplo, na lógica TTL (Lógica Transistor-Transistor) o nível 1 é identificado como uma tensão que está entre 2V e 5V, enquanto que o nível 0 está associada a uma tensão entre 0 e 0,7V. Sendo assim, podemos representar da seguinte forma os níveis binários 0 e 1 na lógica TTL (observe que há outras lógicas com diferentes níveis de tensão):



Note que entre 2V e 5V, o bit é reconhecido como 1 e entre 0V e 0,7V reconhecido como bit 0. Fora desta faixa, entre 0,7V e 2V temos uma faixa indefinida, onde devemos evitar operar de forma a evitar erros de identificação do bit. É importante salientar que há outras lógicas além da TTL, como a CMOS, LVTTL que opera de 0 a 3,3 V dentre outras que funcionam em faixas de tensão diferentes em que antes de se projetar o circuito digital deve se saber em que faixa de tensão o mesmo irá funcionar.

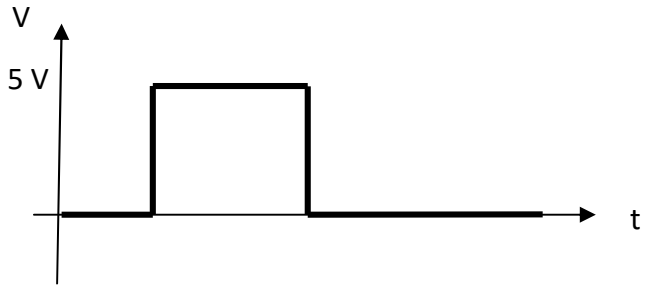
5. Taxa de comunicação

A taxa de comunicação ou baud-rate é a quantidade de bits que podem ser trafegados em uma linha (meio) no intervalo de 1 segundo. Por exemplo, é normal hoje em dia as redes de comunicação local, chamadas LAN terem velocidades na ordem de 10 Mbps (Mega bits por segundo), 100 Mbps, 1 Gbps etc. Outra forma de se comunicar que ainda é muito usada em automação é chamada de RS232, que é uma comunicação do tipo assíncrona em que baud-rates típicos funcionam na velocidade de 1200 bps, 2400 bps, 4800 bps, 9600 bps etc. Essa informação indica o quanto um determinado meio, seja ele por fio ou sem fio (wireless) é capaz de operar de forma segura, garantindo o tráfego correto de informações. Meios de comunicação como a

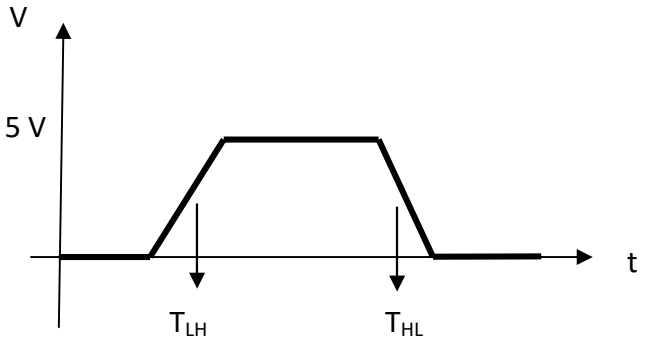
fibra óptica garantem taxas elevadas de comunicação, sendo muito usadas em telecomunicações entre cidades, estados, países ou continentes.

6. Tempo de subida e descida

Observe o gráfico plotado a seguir:



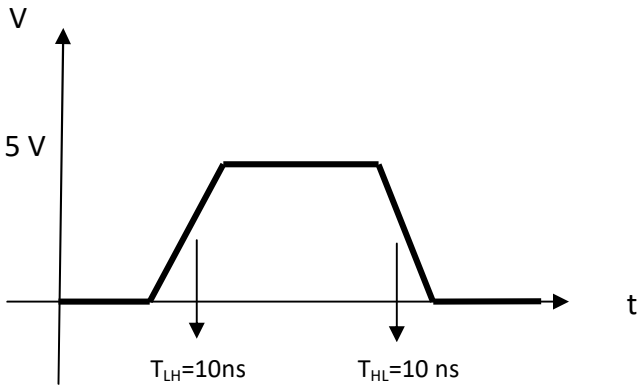
Apesar de parecer que o sinal sai do nível baixo para o alto ou do alto para baixo de forma instantânea, na verdade não é isto o que ocorre, pois normalmente leva-se um tempo para esta transição de nível. Se analisarmos com um osciloscópio com uma escala de divisão da ordem de nanosegundos, veremos algo similar ao apresentado na figura a seguir:



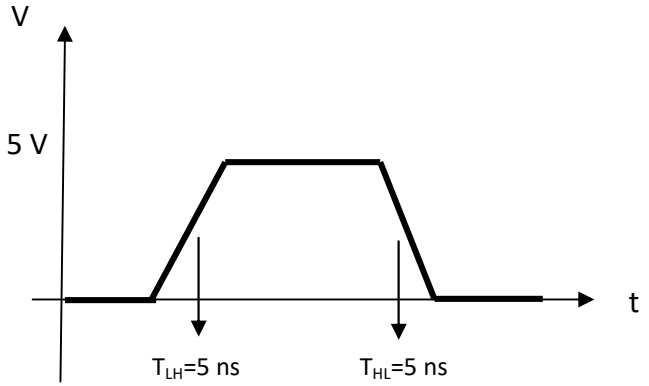
O rise time ou tempo de subida e descida é o tempo que o sinal leva para sair do nível baixo para alto (tempo de subida ou tempo low to high) e de nível alto para baixo (tempo de descida ou tempo high to low). Representamos

este tempo como T_{LH} para o tempo de transição de baixo para alto que é o tempo de subida e T_{HL} para o tempo de transição de alto para baixo que é o tempo de descida.

Quanto menor o tempo de transição, mais rapidamente o circuito digital será capaz de responder. Uma das formas de melhorar o tempo de resposta é diminuir a tensão utilizada para representar o 1, pois o tempo de transição será menor. Como exemplo, observe o gráfico a seguir e note que o tempo que se leva neste gráfico hipotético para a transição T_{LH} e T_{HL}.



Agora se ao invés de usar 5V para representar o bit 1 usássemos 2,5V, o circuito já teria a metade de tempo do rise time do circuito anterior, tornando a resposta do circuito mais rápida, como mostra o gráfico a seguir.



Desta forma, entendemos o motivo dos atuais processadores funcionarem com tensões cada vez menores para representar o bit 1, pois associado a um menor gasto energético, também há a preocupação de deixar o circuito mais rápido, o que é conseguido com um rise time menor.

7. Exercícios

1)Das afirmações abaixo, diga quais são analógicas e quais são digitais.

a)O estado de uma lâmpada= Digital.

b)A temperatura ambiente= Analógico.

c)A voz humana= Analógica.

d)Se o computador está ligado= Digital.

e)A pressão de um pneu= Analógico.

2)Dê outro exemplo de família de circuitos digitais além da TTL e informe sua faixa de operação para os níveis 1 e 0.

Resp: CMOS no qual a faixa do nível alto vai nominalmente de 5V a 18V e o nível baixo de 0 até 3V.

3)Qual a função de um ADC?

Resp: O ADC ou Analog to Digital Converter permite converter uma tensão analógica contínua para uma representação digital de forma discreta.

4)Qual a função de um DAC?

Resp: A função do DAC ou Digital to Analog Converter é inversa do ADC, ou seja, converter uma informação digital para analógica.

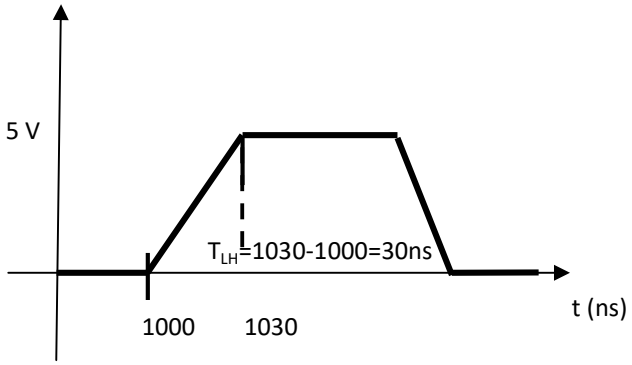
5)Dê exemplos de processadores normalmente utilizados em projetos digitais.

Resp: Os microcontroladores da família PIC, 8051 e AVR são normalmente muito utilizados, pois os mesmos apresentam recursos que possibilitam empregar em uma única pastilha a função do ADC e DAC de forma integrada.

6) Quais outras formas podemos utilizar para representar um estado digital?

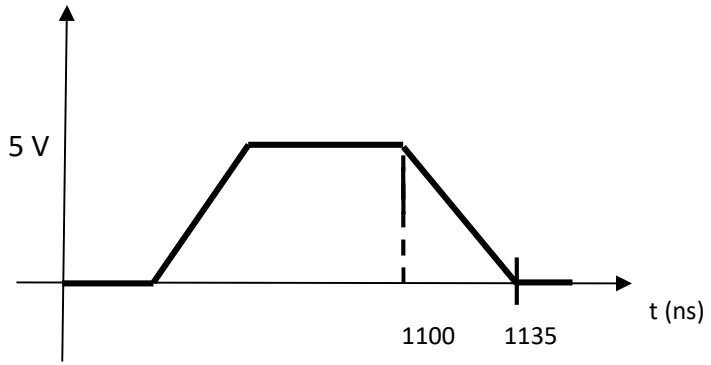
R: Existem várias formas, como o que faz uso do LVDS onde em um sentido de corrente temos a transmissão do bit 1 enquanto em outros estado a transmissão do bit 0.

7) No gráfico a seguir é apresentado o instante de tempo no qual o sinal sai do nível 0 e o instante de tempo onde ele chega no nível 1. Informe qual o tempo de subida deste sinal.



Logo o tempo de subida é de T_{LH}=30ns.

8) No gráfico a seguir é apresentado o instante de tempo no qual o sinal sai do nível 1 e o instante de tempo onde ele chega no nível 0. Informe qual o tempo de descida deste sinal.



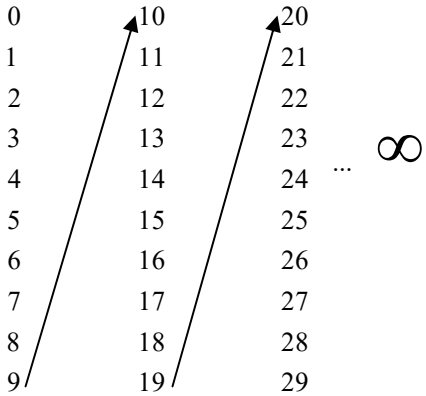
T_{HL}=1135-1100=35ns

Logo o tempo de subida é de T_{HL}=35ns. Note que não necessariamente o tempo de subida e descida em um circuito digital é igual, sendo necessário a consulta do datasheet para observar o tempo exato destes parâmetros.

Capítulo II
Sistemas de Numeração

1. Sistema Decimal

O sistema decimal é baseado no número de dedos presentes em nossas mãos e é usado em nosso cotidiano e representado pelos números 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9. Ao realizarmos uma contagem no dígito das unidades e chegamos ao valor máximo de contagem, no caso 9, o dígito seguinte, no caso a dezena, é incrementada permitindo assim que a contagem seja feita de forma posicional e sequêncial, acompanhe:



Um número decimal pode ser decomposto em unidades, dezenas, centenas etc, como apresentado a seguir:

$1=1x10^{0}$
$12=1x10^{1}+2x10^{0}$
$153=1x10^{2}+5x10^{1}+3x10^{0}$
$1214=1x10^{3}+2x10^{2}+1x10^{1}+4x10^{0}$
$156.53=1x10^{2}+5x10^{1}+6x10^{0}+5x10^{-1}+3x10^{-2}$

Obs: *No sistema americano, que é o que será adotado neste livro, o ponto é*

usado no lugar da vírgula.

O dígito mais a esquerda do número é chamado de MSD (Most Significant Digit – Dígito Mais Significativo) e o mais a direita de LSD (Least Siginificant Digit – Dígito Menos Significativo), observe os exemplos a seguir:

201		
MSD		LSD
2	0	1
10^2	10^1	10^0

1256			
MSD			LSD
1	2	5	6
10^3	10^2	10^1	10^0

10.98			
MSD			LSD
1	0	9	8
10^1	10^0	10^{-1}	10^{-2}

12	
MSD	LSD
1	2
10^1	10^0

Porém, observe que no mundo digital temos apenas dois níveis, neste caso 0 e 1. Desta forma, precisaremos converter o conteúdo de um sistema de numeração para outro de forma a podermos manipular corretamente este valor. Nos próximos tópicos será apresentado como realizar a conversão nas diversas bases usadas como binário, octal, decimal e hexadecimal.

A representação de um sistema é feito colocando-se ao término do número a sua base, como observado abaixo:

$$26_{10}$$

Neste caso temos o 26 na base 10. Normalmente, usaremos este tipo de representação para outras bases onde na base decimal ficará omitido o mesmo.

O número de dígitos diferentes que o sistema será capaz de representar será dado da seguinte forma:

$$10^N$$

Ou seja, se tivermos apenas 1 dígito, poderemos representar 10^1 números diferentes, neste caso de 0 a 9.

Já a contagem máxima no sistema será dada pelo número de dígitos que compõem o mesmo. A forma genérica é dada por:

$$10^N - 1$$

Onde N é o número de dígitos. Digamos que tenhamos 1 dígito, sendo assim a contagem máxima será:

$$10^1 - 1 = 9$$

Que pode ser comprovado pela contagem abaixo onde temos 10 dígitos a começar do 0 até 9:

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

Agora se dispusermos de 2 dígitos, teremos como contagem máxima o seguinte valor:

$$10^2 = 100$$

Que são 100 números diferentes. O valor máximo de contagem será dado por:

$$10^2 - 1 = 99$$

Que pode novamente ser comprovada pela contagem abaixo:

$$0,1,2,3,4,5,6,7,8,9,10,11,12...99$$

2. Sistema Binário

No sistema binário como já é sabido, temos apenas duas representações, sendo estas o 0 e 1. Da mesma forma que o sistema decimal, ao término da contagem máxima o próximo dígito é incrementado. Sendo assim, um exemplo de contagem em binário seria o apresentado a seguir:

0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011...

A representação do número da base binária será feita da seguinte forma:

$$1010_2$$

A quantidade de números binários será dado pela quantidade de bits usados como apresentado a seguir de forma genérica:

$$2^N$$

Se for usado apenas 1 bit, poderemos representar $2^1 = 2$ números diferentes, onde neste caso seria o 0 ou 1.

Da mesma forma que os números decimais, a contagem máxima será dada pelo número de dígitos menos 1, sendo expressa de maneira genérica da seguinte forma:

$$2^N - 1$$

Digamos que tenhamos 1 dígito binário, logo a contagem máxima será de:

$$2^1 - 1 = 1$$

Como pode ser comprovado pelo exemplo a seguir:

$$0,1$$

Da mesma forma se tivermos dois dígitos, poderemos contar até:

$$2^2 - 1 = 3$$

Como pode ser comprovado pelo exemplo a seguir:

$$0,1,10,11$$

Ou seja, quanto mais dígitos tivermos maior será o valor que poderá ser representado em binário.

Da mesma forma que no número decimal, encontraremos também nos dígitos binários o dígito mais significativo (MSB – Most Significant Bit ou Bit Mais Significativo) e o menos significativo (LSB – Least Significant Bit ou Bit Menos Significativo). Por exemplo, o número 1010 possui MSB=1 e LSB=0. Acompanhe a seguir outros exemplos, onde também é apresentado a potência binária de cada dígito.

10	
MSB	LSB
1	0
2^1	2^0

1001			
MSB			LSB
1	0	0	1
2^3	2^2	2^1	2^0

11010001							
MSB							LSB
1	1	0	1	0	0	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

2.1 Aglomerado de bits

Chamamos de aglomerado de bits ao conjunto formado por uma sequência de bits. Se tivermos um conjunto de 4 bits, como o apresentado abaixo, teremos um *nibble*:

1010

No entanto se tivermos o conjunto formado por 8 bits como o caso abaixo, chamaremos este de *byte:*

10101111

Caso tenhamos um conjunto de 16 bits, chamaremos este de *word:*

1010111100001111

A partir do *word,* temos o *dword (double word)* para indicar palavras binárias de 32 bits e *qword (quad word)* para palavras binárias de 64 bits, sendo esta lógica mantida para aglomerações maiores do que estas.

2.2 Conversão de binário para decimal

Na próxima tabela temos um exemplo de contagem em binário e decimal, mostrando a equivalência entre as duas bases.

Binário	Decimal
0	0
1	1

10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Para convertermos um número de binário para decimal, devemos empregar o somatório de cada um dos bits de acordo com a sua potência binária, como expresso de forma genérica a seguir:

$$\sum_{i=0}^{p-1} d_i 2^i$$

Como exemplo, observe a conversão de binário para decimal dos exemplos abaixo:

a) $10_2 = 1\text{x}2^1 + 0\text{x}2^0 = 2 + 0 = 2_{10}$

b) $111_2 = 1x2^2 + 1x2^1 + 1x2^0 = 4 + 2 + 1 = 7_{10}$

c) $1001_2 = 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0 = 8 + 1 = 9_{10}$

d) $10011_2 = 1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 16 + 1 = 17_{10}$

2.3 Conversão de decimal para binário

Podemos também converter um número da forma decimal para binário fazendo uso da técnica das divisões sucessivas, em que um número na forma decimal é dividido por 2 observando sempre o resto da divisão, que poderá ser 0 ou 1. A divisão chega ao fim quando já não é mais possível realizar as divisões sucessivas. Após esta etapa, deveremos formar o número binário do MSB para o LSB onde o bit mais significativo será o último resto da divisão e o LSB o resultado do quociente da primeira divisão. Observe o exemplo abaixo para conversão do número decimal 3 para binário:

$$3 \div 2 = 1 \; Resto \; 1 \longrightarrow LSB$$
$$\downarrow$$
$$MSB$$

Logo 3 em decimal é igual a 11 em binário. Podemos representar da seguinte forma:

$$3_{10} = 11_2$$

Observe agora outro exemplo de conversão, neste caso do número 6 em decimal para binário:

$$6 \div 2 = 3 \; Resto \; 0 \longrightarrow LSB$$

$$3 \div 2 = 1 \; Resto \; 1$$
$$\downarrow$$
$$MSB$$

Logo, podemos comprovar que 6 em decimal é igual a 110 em binário, podendo ser representado da forma abaixo:

$$6_{10} = 110_2$$

Vejamos agora a conversão do número 20 em decimal para binário:

$$20 \div 2 = 10 \; Resto \; 0 \quad \rightarrow LSB$$
$$10 \div 2 = 5 \; Resto \; 0$$
$$5 \div 2 = 2 \; Resto \; 1$$
$$2 \div 2 = 1 \; Resto \; 0$$
$$\downarrow$$
$$MSB$$

Logo, observarmos que 20 em decimal é igual a 10100 em binário, podendo ser representado da forma abaixo:

$$20_{10} = 10100_2$$

3. Sistema Octal

O sistema octal é aquele que faz uso de 8 símbolos para representar o seu conteúdo, onde neste caso a contagem é feita de 0 até 7. Após o valor 7, a contagem reinicializa de 10 em octal onde seu equivalente em decimal seria o 8. Observe a tabela a seguir que apresenta a contagem em octal de 0 até 15 e seu equivalente no sistema binário e decimal.

Octal	Binário	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
10	1000	8
11	1001	9
12	1010	10
13	1011	11
14	1100	12
15	1101	13
16	1110	14
17	1111	15

A representação em octal é muito usada em computação, pois torna a visualização dos dados mais fáceis do que a visualização em binário. Como exemplo, observe a sequência de dígitos binários abaixo:

$$11111111000000001111111100000000_2$$

Este valor convertido em octal ficaria da seguinte forma:

$$37700177400_8$$

Note que temos o mesmo conteúdo apenas convertido para outra base, tornando assim a sua leitura mais fácil do que a binária.

O conceito já apresentado de dígito mais e menos significativo continua válido para a base octal, assim como os pesos de cada dígito.

3.1 Conversão Octal para Binário

A conversão de octal para binário é feita de forma direta, observando o equivalente de cada valor octal em binário, conforme a tabela a seguir:

Octal	Binário
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Sendo importante frisar que cada octal equivale a três bits, onde estes devem ser colocados de forma completa. Como exemplo, observe como ficaria a conversão dos números octais em binário abaixo.

a) $4_8 = 100_2$

b) $14_8 = 001100_2$

c) $123_8 = 001010011_2$

d) $1456_8 = 001100101110_2$

A reciproca também é verdadeira para o caso de conversão de binário para octal, como demonstrado nos exemplos a seguir:

a) $101_2 = 5_8$

b) $110100_2 = 64_8$

c) $011111_2 = 37_8$

d) $001100101110_2 = 1456_8$

3.2 Conversão Octal para Decimal

Esta conversão segue o mesmo padrão adotado para a conversão de binário para decimal com a diferença que a base adotada é octal. Acompanhe os exemplos a seguir.

a) $15_8 = 1x8^1 + 5x8^0 = 8 + 5 = 13$

b) $100_8 = 1x8^2 + 0x8^1 + 0x8^0 = 64 + 0 + 0 = 64_{10}$

c) $2001_8 = 2x8^3 + 0x8^2 + 0x8^1 + 1x8^0 = 1024 + 1 = 1025_{10}$

d) $12345_8 = 1x8^4 + 2x8^3 + 3x8^2 + 4x8^1 + 5x8^0 = 5349_{10}$

4. Sistema Hexadecimal

O sistema hexadecimal é aquele que faz uso de 16 símbolos para representar uma contagem, tendo largo uso para a representação de números que ficariam muito extensos na forma binária. Os símbolos usados são:

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

Observe na próxima tabela a equivalência do sistema hexadecimal com o binário e decimal.

Hexadecimal	Binário	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Verifique que precisamos de um conjunto de 4 bits para representar um caractere em hexadecimal. Esta representação é muito usada em computação, pois torna a visualização dos dados mais fáceis do que a visualização em binário. Como exemplo, observe a sequência de dígitos binários abaixo:

$$11111111000000001111111100000000_2$$

Este valor convertido em hexadecimal ficaria da seguinte forma:

$$FF00FF00_{16}$$

4.1 Conversão de hexadecimal para binário

A conversão de hexadecimal para binário é feita de forma direta, observando o equivalente de cada valor em hexa para binário, conforme a tabela apresentada anteriormente sendo importante frisar que cada número hexa equivale a quatro bits, onde estes devem ser colocados de forma completa. Como exemplo, observe como ficaria a conversão dos números hexadecimais em binário abaixo.

a) $4_{16} = 0100_2$

b) $14_{16} = 00010100_2$

c) $123_{16} = 000100100011_2$

d) $1456_{16} = 0001010001010110_2$

A reciproca também é verdadeira para o caso de conversão de binário para hexadecimal, como demonstrado nos exemplos a seguir:

a) $0101_2 = 5_{16}$

b) $11110100_2 = F4_{16}$

c) $10110011_2 = B3_8$

d) $FF_{16} = 11111111_2$

e) $AB_{16} = 10101011_2$

f) $ABC1_{16} = 1010101111000001_2$

4.2 Conversão Hexadecimal para Decimal

Esta conversão segue o mesmo padrão adotado para a conversão de binário para decimal com a diferença que a base adotada é hexa. Acompanhe

os exemplos a seguir.

a) $15_{16} = 1\text{x}16^1 + 5\text{x}16^0 = 21_{10}$

b) $100_{16} = 1\text{x}16^2 + 0\text{x}16^1 + 0\text{x}16^0 = 256_{10}$

c) $2001_{16} = 2\text{x}16^3 + 0\text{x}16^2 + 0\text{x}16^1 + 1\text{x}16^0 = 8193_{10}$

d) $12345_{16} = 1\text{x}16^4 + 2\text{x}16^3 + 3\text{x}16^2 + 4\text{x}16^1 + 5\text{x}16^0 = 74565_{10}$

e) $AB_{16} = 10\text{x}16^1 + 11\text{x}16^0 = 171_{10}$

f) $D12A_{16} = 13\text{x}16^3 + 1\text{x}16^2 + 2\text{x}16^1 + 10\text{x}16^0 = 53546_{10}$

g) $FI_{16} = 15\text{x}16^1 + 1\text{x}16^0 = 241_{10}$

h) $A9_{16} = 10\text{x}16^1 + 9\text{x}16^0 = 169_{10}$

i)

$$FAABB_{16} = 15\text{x}16^4 + 10\text{x}16^3 + 10\text{x}16^2 + 11\text{x}16^1 + 11\text{x}16^0 = 1026747_{10}$$

5. Código BCD

O código BCD (Binary Coded Decimal – Binário Codificado em Decimal) é uma forma de representar de forma direta os números no formato decimal em binário. Desta forma, o mesmo permite representar do número 0 até 9 em decimal onde os números decimais maiores que 9 não estão implementados, precisando assim de 2 números BCD para realizar tal

representação. Observe a seguir o código BCD dos números decimais de 0 a 9.

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

Os números decimais maiores que 9, como exemplo de 10 a 15 seriam representados da seguinte forma através do BCD.

Decimal	BCD
10	0001 0000
11	0001 0001
12	0001 0010
13	0001 0011
14	0001 0100
15	0001 0101

Esta mesma ideia é válida para outros números. Digamos que desejemos representar o 99 em BCD, neste caso o resultado seria:

1001 1001

Observe que a maior vantagem de usar o BCD está na facilidade de conversão do mesmo já que cada caractere decimal é convertido de maneira direta para o seu representante BCD, porém este acaba por usar mais bits se comparado com o binário puro apresentado anteriormente. Vejamos a seguir alguns exemplos de conversão de decimal para BCD.

a) $20_{10} \quad BCD: 0010 \quad 0000$

b) $53_{10} \quad BCD: 0101 \quad 0011$

c) $100_{10} \quad BCD: 0001 \quad 0000 \quad 0000$

d) $259_{10} \quad BCD: 0010 \quad 0101 \quad 1001$

e) $777_{10} \quad BCD: 0111 \quad 0111 \quad 0111$

f) $1234_{10} \quad BCD: 0001 \quad 0010 \quad 0011 \quad 0100$

g) $5789_{10} \quad BCD: 0101 \quad 0111 \quad 1000 \quad 1001$

h) $10998_{10} \quad BCD: 0001 \quad 0000 \quad 1001 \quad 1001 \quad 1000$

i) $99999_{10} \quad BCD: 1001 \quad 1001 \quad 1001 \quad 1001 \quad 1001$

j) $65_{10} \quad BCD: 0110 \quad 0101$

6. Código Gray

Observe o código binário abaixo:

0001

O incremento, ou seja, o próximo número após este é:

0010

Note que ocorreram duas variações de bits, neste caso os bits que sofreram variações foram os bits 0 e 1. Em alguns sistemas de controle, isto pode ser um empecilho para o funcionamento correto do sistema, pois um ruído externo pode mudar o estado de um bit, ocasionando assim uma falha na medição do sistema. Para resolver isso, existe o código Gray, onde entre uma transição e outra é garantido que apenas um bit irá variar. Acompanhe na próxima tabela um comparativo entre o código Gray, binário e decimal.

Gray	Binário	Decimal
0000	0000	0
0001	0001	1
0011	0010	2
0010	0011	3
0110	0100	4
0111	0101	5
0101	0110	6
0100	0111	7
1100	1000	8
1101	1001	9
1111	1010	10
1110	1011	11
1010	1100	12

1011	1101	13
1001	1110	14
1000	1111	15

Observe que em todos os casos, apenas 1 bit sofreu uma alteração entre uma contagem e outra, trazendo assim diversas aplicações para o código Gray, principalmente aquelas ligadas a uma maior segurança na medida, como sistemas balísticos.

A seguir está apresentado alguns exemplos de conversão de código decimal para o Gray, fazendo uso direto da tabela apresentada.

a) $10_{10} \quad Gray: 0001 \quad 0000$

b) $145_{10} \quad Gray: 0001 \quad 0110 \quad 0111$

c) $887_{10} \quad Gray: 1100 \quad 1100 \quad 0100$

7. Código Alfanumérico

Cada caractere seja ele letras maiúsculas, minúsculas, especiais e numéricos armazenados na memória do computador obedecem um padrão onde normalmente o utilizado é o ASCII (American Standard Code for Information Interchange - Código Padrão Americano para Troca de Informações). Neste padrão, cada dígito possui uma representação em bits que permite saber o que está armazenado na memória do computador. Em seguida é apresentada esta tabela, chamada ASCII padrão.

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Por exemplo, o caractere "2" fica armazenado na memória como 32_{16}. Já o caracter "A" é o 41_{16} e o "a" o 61_{16}. Existe também o ASCII extendido que engloba caracteres maiores que $7F_{16}$ onde neste grupo encontraremos símbolos como o ~ dentre outros.

A seguir alguns exemplos que demonstram os números equivalentes do caractere ASCII com o valor hexa e binário.

a) $Chr: Z \quad Hexa: 5A_{16} \quad Binário: 01011010$

b) $Chr: L \quad Hexa: 4C_{16} \quad Binário: 01001100$

c) $Chr: w \quad Hexa: 77_{16} \quad Binário: 01110111$

8. Comparativo dos sistemas apresentados

A seguir está apresentado uma tabela que apresenta um quadro comparativo entre as diversas bases vistas até agora.

Decimal	Hexa	Octal	Binário	BCD	Gray
0	0	0	0000	0000	0000
1	1	1	0001	0001	0001
2	2	2	0010	0010	0011
3	3	3	0011	0011	0010

4	4	4	0100	0100	0110
5	5	5	0101	0101	0111
6	6	6	0110	0110	0101
7	7	7	0111	0111	0100
8	8	10	1000	1000	1100
9	9	11	1001	1001	1101
10	A	12	1010	0001 0000	1111
11	B	13	1011	0001 0001	1110
12	C	14	1100	0001 0010	1010
13	D	15	1101	0001 0011	1011
14	E	16	1110	0001 0100	1001
15	F	17	1111	0001 0101	1000

9. Paridade

O bit de paridade é um bit adicional, normalmente colocado a esquerda da sequência ASCII, ocupando o lugar do bit MSB que possui a função de conferir se a informação que está guardada na memória ou está sendo transmitida por uma linha serial, por exemplo, está correta. Encontramos dois tipos de paridade, neste caso a par e a ímpar onde caso a paridade par seja utilizada, a soma de todos os bits em nível alto presentes no byte devem resultar em um número par. Caso a opção seja ímpar, a soma de todos os bits devem resultar em um número ímpar. O exemplo a seguir mostra como fica o estado do bit de paridade par para armazenamento do caractere "A" da tabela ASCII (Código 41 em Hexa ou 1000001 em Binário):

$$\overbrace{0}^{Paridade}\ \overbrace{1000001}^{Caractere\,A} = 01000001$$

Caso a paridade ímpar fosse adotada, a informação ficaria da seguinte

forma:

$$\overbrace{1}^{Paridade}\ \overbrace{1000001}^{Caractere\ A} = 11000001$$

Observe que no primeiro caso a soma de todos os bits em 1 resulta em um número par enquanto que no segundo, que faz uso da paridade ímpar a soma de todos os bits resulta em número desta natureza. Desta forma, o computador é capaz de detectar se a informação armazenada está correta ou não. Por exemplo, voltando ao exemplo anterior digamos que a seguinte informação esteja armazenada, porém o modo utilizado é paridade par:

$$\overbrace{1}^{Paridade}\ \overbrace{1000001}^{Caractere\ A} = 11000001$$

Veja que desta forma, o processador é capaz de detectar que a informação é inválida, em virtude principalmente de interferências presentes como ruídos, que podem levar a alteração do estado de um bit, onde neste caso o sistema pode solicitar novamente a informação de modo a recuperar a mesma. Apesar de ser um método simples e eficaz o método da verificação por paridade não é o melhor, pois oferece uma probabilidade alta de erros. Voltemos ao caso do caractere "A", porém supondo que ao invés de um bit errado, hajam dois, neste caso teríamos:

$$\overbrace{0}^{Paridade}\ \overbrace{1010101}^{Caractere\ A} = 01010101$$

Observe que a soma dos bits ainda é um número par e desta forma, o processador irá aceitar a informação como válida. Para sanar este problema, há outros métodos como o CRC, LRC e Checksum onde o último será apresentado no próximo tópico.

10. Checksum

A soma de verificação (checksum) consiste em somar o conteúdo de todos os bytes truncando a parte superior ao byte de forma a verificar se a informação armazenada ou transmitida está correta. Digamos que tenhamos dois bytes de informação, logo com o byte de checksum teremos a seguinte estrutura:

$$Byte_1 \quad Byte_2 \quad Checksum$$

Vamos supor que sejam usados 6 bytes, logo teremos:

$$Byte_1 \quad Byte_2 \quad Byte_3 \quad Byte_4 \quad Byte_5 \quad Byte_6 \quad Checksum$$

Esta estrutura se mantém até o enésimo byte de informação, como representado a seguir:

$$Byte_1 \quad Byte_2 \quad ..Byte_{n-1} \quad Byte_n \quad Checksum$$

Vamos agora supor um exemplo prático, em que o byte 1 tenha o valor 01010000 e o byte 2 o valor 00001010. A soma destes dois bytes, ou seja, o seu checksum será:

$$01010000 + 00001010 = 01011010$$

Logo o checksum é igual a 01011010. Estes bytes ficariam organizados da seguinte forma:

$$\overbrace{01010000}^{Byte\,1} \quad \overbrace{00001010}^{Byte\,2} \quad \overbrace{01011010}^{Checksum}$$

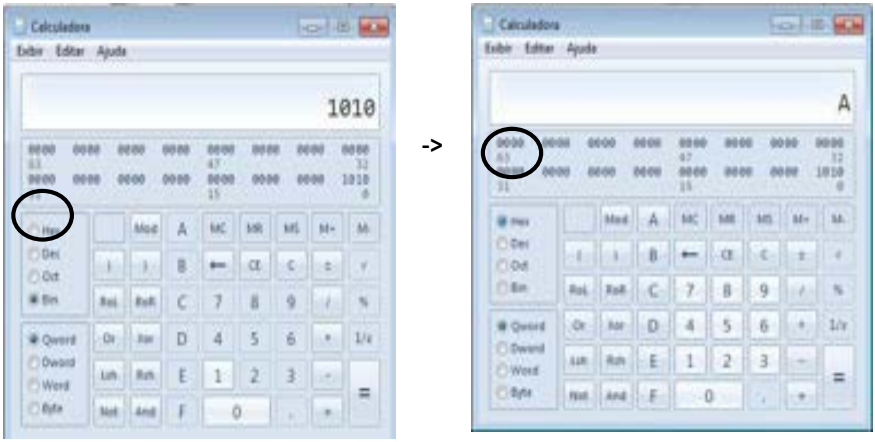
A lógica de funcionamento consiste em somar todos os bytes e conferir se esta soma é igual ao checksum, onde caso sejam os dados estão corretos e caso contrário é sinal que houve algum erro no armazenamento ou recepção dos dados.

11. Recursos computacionais

Os sistemas operacionais modernos como o Windows, Linux e Mac vem equipados com uma calculadora que permite realizar conversões entre base de forma muito rápida. Observe a seguir a calculadora do Windows configurada para o modo Programador:



Observe que do lado esquerdo do programa podemos selecionar uma das bases apresentadas, como a binária e convertê-la para outras bases como a decimal, hexadecimal e octal. Acompanhe a seguir o programa convertendo a sequência 1010 em binário para hexadecimal.



Apesar de ser um ótimo recurso computacional, procure usar as técnicas apresentadas de forma a entendê-las perfeitamente e procure usar estes programas para realizar uma conferência, de forma a saber se está correto o resultado encontrado.

12. Representação Sinal Magnitude

Até agora vimos a representação de números na forma positiva, porém em nosso cotidiano é muito comum também trabalharmos com números negativos. A representação binária na forma de sinal magnitude permite com que façamos isso, já que passamos a contar com um bit de sinal, onde 1 indicará números negativos e 0 números negativos. Esta representação segue a seguinte formatação:

$$N_s = s\,,\, a_{(n-1)}\,,\, a_{(n-2)}\,,\, \ldots\,,\, a_{(1)}\,,\, a_{(0)}$$

Observe a seguir a representação apenas de números positivos de 3 bits.

Binário	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Agora observe este conjunto de bits, porém empregando o sinal magnitude:

Binário	Decimal
011	+3
010	+2
001	+1
000	0
100	0
101	-1
110	-2
111	-3

Note que pelo sinal magnitude, temos duas representações para o 0. O bit 1 e 0 formam o valor binário em si, que correspondem a conversão de binário para decimal enquanto que o bit 2 serve como sinalizador, informando se o número é positivo ou negativo. Analise a seguir alguns números desta