Circuitos Lógicos

Módulo # 1

Sistemas de Numeração e Aritmética Digital

Primeiro computador eletrônico Eniac

Em fevereiro, na Universidade da Pensilvânia :

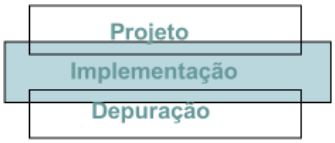
18 mil válvulas e um custo de US$20 milhões – 1946

ENIAC contained 17,468 vacuum tubes, 7,200 crystal diodes, 1,500 relays, 70,000 resistors, 10,000 capacitors and around 5 million hand-soldered joints. It weighed 30 short tons (27 t), was roughly 8.5 feet by 3 feet by 80 feet (2.6 m by 0.9 m by 26 m), took up 680 square feet (63 m²), and consumed 150 kW of power. Input was possible from an IBM card reader, while an IBM card punch was used for output. These cards could be used to produce printed output offline using an IBM accounting machine, probably the IBM 405.

Como é o projeto de hardware hoje

- **Contínua utilização de ferramentas de CAD**
  - . desestímulo aos processos manuais
  - . estímulo às representações abstratas
  - . projeto de hardware parecido com o de software
- **Tecnologia empregada na implementação**
  - . lógica discreta → lógica programável

Algorítmo de projeto



Projeto
- Início: que funções devem ser realizadas ?
- Necessidades: velocidade? tamanho? custo?
- Blocos abstratos ⟶ realizações concretas

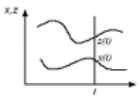
Implementação
- Funções primitivas unidas em blocos complexos
- Ligação dos blocos
- Escolha entre alternativas para otimização

Depuração (Debug)
- Sistema com falhas : projeto, estrutura ou componentes
- Projeto deve facilitar a depuração

Formas de representação

Blocos

Formas de onda



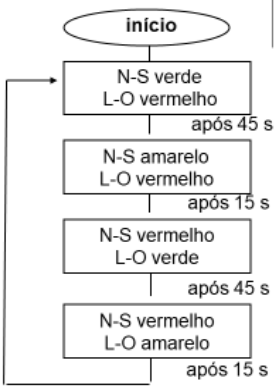
A	B	C	x = ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tabelas verdade

Porta

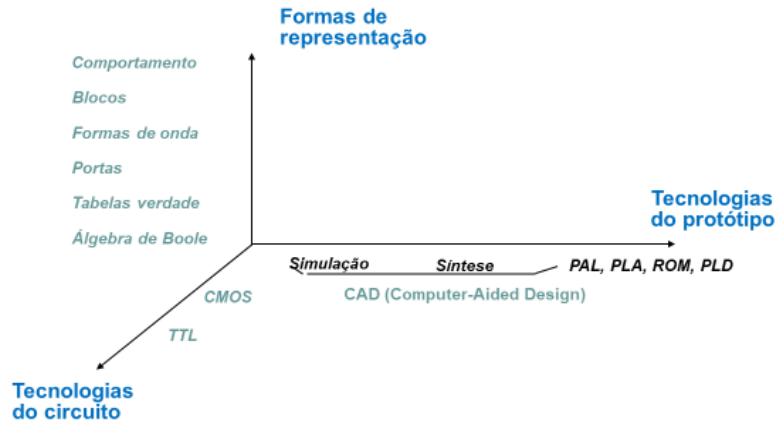
$$X = \overline{A}.B + A.\overline{B}$$

Álgebra de Boole

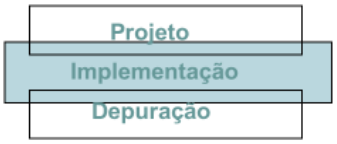


Comportamento

Elementos de projeto



Algorítmo de projeto



Projeto
- Início: que funções devem ser realizadas ?
- Necessidades: velocidade? tamanho? custo?
- Blocos abstratos → realizações concretas

Implementação
- Funções primitivas unidas em blocos complexos
- Ligação dos blocos
- Escolha entre alternativas para otimização

Depuração (Debug)
- Sistema com falhas : projeto, estrutura ou componentes
- Projeto deve facilitar a depuração

Exemplo de Projeto

1. Especificação funcional: o que o sistema deve fazer

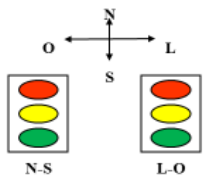
Ex: Controle de sinal de trânsito

. Dois sinais colocados nas direções N e S e dois nas direções L e O

. Os ciclos seguem a seguinte sequência : VERDE-AMARELO-VERMELHO

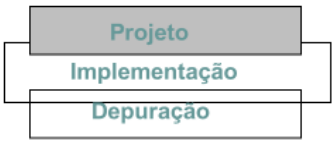
. N-S e L-O não ficam VERDE ou AMARELO simultaneamente

. Fica VERDE por 45 segundos, AMARELO por 15, VERMELHO por 45 segundos.



2. Performance requerida

velocidade: chaveamento em menos de 100 ms
potência: menor que 20 W
área: menor que 20 cm2
custo de fabricação: menor que R$ 20,00



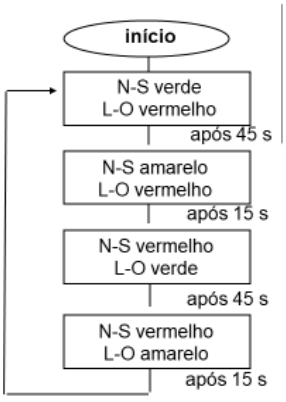
Projetar é saber representar

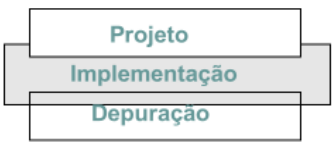
1 - Especificação em linguagem natural: fácil de escrever, mas imprecisa e sujeita a ambiguidades.

2 - Descrição funcional: especificação mais precisa através de diagramas de fluxo e/ou fragmentos de programas

3 - Descrição estrutural: decomposição de componentes complexos

4 - Descrição física: projeto nos blocos mais primitivos (p.e. portas ou transistores)





Projetar é saber representar

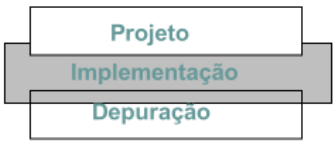
1 - Especificação em linguagem natural: fácil de escrever, mas imprecisa e sujeita a ambiguidades.

2 - Descrição funcional: especificação mais precisa através de diagramas de fluxo e/ou fragmentos de programas

3 - Descrição estrutural: decomposição de componentes complexos

4 - Descrição física: projeto nos blocos mais primitivos (p.e. portas ou transistores)

O algorítmo do projeto - Construção



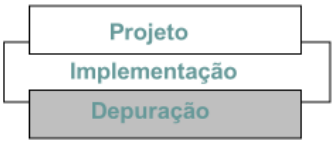
Projeto top-down:
funções complexas substituídas por funções primitivas

Projeto bottom-up:
primitivas são compostas para montagem de funções cada vez mais complexas

Regras de composição:
elétricas: quantos componentes podem ser consecutivos
tempo: velocidade de propagação do sinal

Algorítmo do projeto - Depuração (Debug)

O que pode sair errado ?

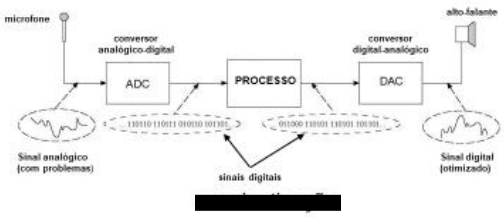


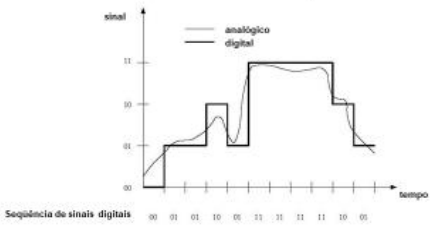
. *Falhas de projeto*
A implementação não atende às especificações
Projeto lógico incorreto
Interpretações falsas ou casos extremos ignorados

. *Falhas de implementação*
Funções modulares corretas, porém sua composição possui erros
Falsas interpretações de comportamento de interface e de temporização
Erros de ligação e elétricos

. *Falhas de componentes*
Componentes fora de especificação ou danificados

sinais analógicos X sinais digitais





sinais analógicos X sinais digitais

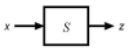
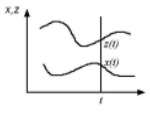
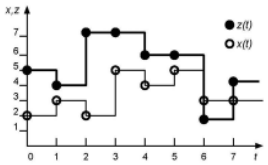


diagrama de blocos



sinais analógicos

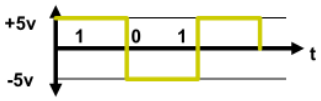


sinais digitais

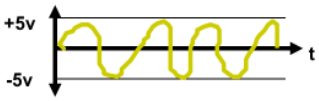
t	0	1	2	3	4	5	6	7
x(t)	2	3	2	5	4	5	3	3
z(t)	5	4	7	7	6	6	2	4

sequências de valores

Projeto de Hardware Digital



forma de onda digital



forma de onda analógica

Vantagens dos Sistemas Digitais sobre os Analógicos

. pequenos erros nas entradas dos sistemas analógicos podem significar grandes erros nas saídas.

. os sistemas digitais são mais precisos e de mais fácil leitura de informações.

. os sistemas digitais são simples para construção de blocos.

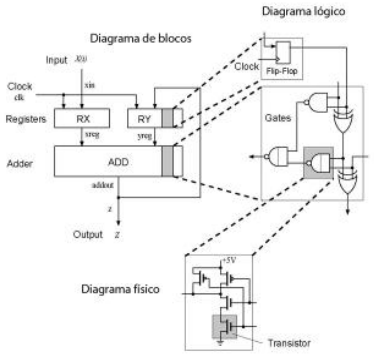
. computadores usam internamente sistemas essencialmente digitais.

. a interface com o meio externo é normalmente analógica.

Níveis de implementação

Exemplo

$$Z(t) = \Sigma_{i=0}^{t} X(i)$$



Projetar é saber representar.

Mas ... o que se quer representar ?

- Caracteres
- Funções
- Observações
- Ações

Projetar é saber representar.

Mas ... o que se quer representar ? Um exemplo ...

Uma fábrica precisa de uma sirene para indicar final de expediente.

Ela deve ser ativada quando ocorrer uma das seguintes condições:

* já passou das 17:00 h e as máquinas estão ligadas ;

* é sexta-feira, a produção foi atingida e todas as máquinas estão desligadas.

SISTEMAS DE NUMERAÇÃO

Sistema de base N

→ tem símbolos b_i , de valores [0,N - 1], ocupando as ordens de peso N^i

$$\ldots \; \underbrace{b_3}_{b_3 \times N3} \; \underbrace{b_2}_{b_2 \times N2} \; \underbrace{b_1}_{b_1 \times N1} \; \underbrace{b_0}_{b_0 \times N0} \; \underbrace{b_{-1}}_{b_{-1} \times N-1} \; \underbrace{b_{-2}}_{b_{-2} \times N-2} \; \ldots$$

Bases utilizadas em sistemas digitais

a) base binária → $N = 2$, $b_i = [\,0\,,\,1\,]$

cada b_i na ordem de peso N^i é chamado de **bit**
ex.: 100110_2 1_2

b) base octal → $N = 8$, $b_i = [\,0,\,7\,]$

ex.: 473_8 110_8

c) base hexadecimal → $N = 16$, $b_i = [\,0,\,15\,]$ V [A, F]

como o intervalo [10, 15] cada valor utiliza 2 algarismos devemos substituí-los por letras; portanto ...
[10, 15] ⇒ [A, F]

ex.: 473_H 110_H $3A1_H$

Conversão de bases em geral

- 10 ⇨ b : divisões sucessivas
- b ⇨ 10 : decomposição

378_{10} ⇨ hexa

$$\frac{378}{16} = 23 + \text{o resto de } 10_{10} = A_{16}$$

$$\frac{23}{16} = 1 + \text{o resto de } 7$$

$$\frac{1}{16} = 0 + \text{o resto de } 1$$

378_{10} ⇨ $17A_H$

$2AF_H$ ⇨ decimal

$$2AF_{16} = 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0$$
$$= 512 + 160 + 15$$
$$= 687_{10}$$

Código BCD

- BCD (*binary-coded-decimal*) é uma maneira muito utilizada de apresentar números decimais em formato binário.
- Cada dígito é convertido em um binário equivalente. BCD *não é* um sistema numérico.
- A principal vantagem do BCD é a comunicação, com a relativa facilidade de conversão para o sistema decimal.

8	7	4	(decimal)
↓	↓	↓	
1000	0111	0100	(BCD)

110100000111001_2 ⇨ BCD

$$\underbrace{0110}_{6}\ \underbrace{1000}_{8}\ \underbrace{0011}_{3}\ \underbrace{1001}_{9}$$

O que os números podem representar ?

- A maioria dos microcomputadores manipulam e armazenam informações e dados binários em grupos de 8 bits. Oito bits equivale a 1 byte.
- Uma palavra é um grupo de bits que representa uma determinada unidade de informação.
- O tamanho da palavra pode ser definido como o número de bits na palavra binária em que um sistema digital opera. O tamanho da palavra de um PC é de 8 bytes (64 bits).
- O código alfanumérico ASCII (American Standard Code for Information Interchange) representa todos os caracteres e as funções encontrados em um teclado de computador: 26 letras minúsculas e 26 maiúsculas, 10 dígitos, 7 sinais de pontuação, de 20 a 40 outros caracteres.

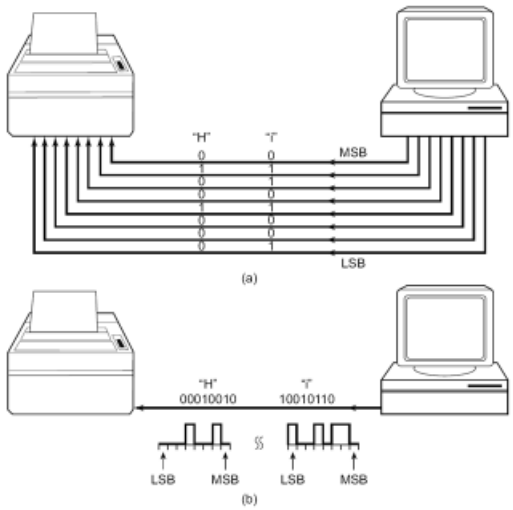
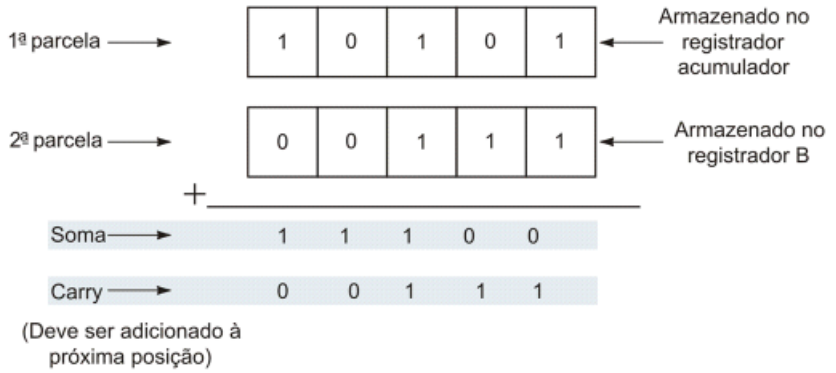


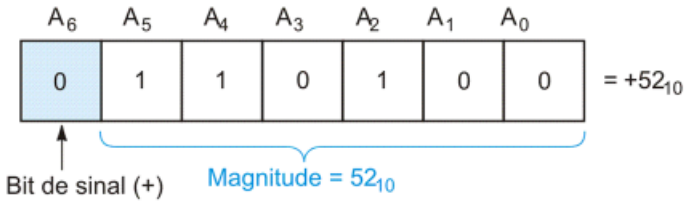
FIGURA 1.15 (a) A transmissão paralela usa uma linha de conexão por bit, e todos os bits são transmitidos simultaneamente; (b) a transmissão serial usa apenas uma linha de sinal, na qual os bits são transmitidos serialmente (um de cada vez).

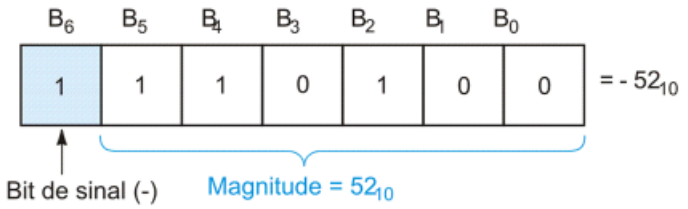
Aritmética - processo típico de uma adição binária.



Representação de números com sinal

1) forma sinal-magnitude.





Representação de números com sinal

2) forma complemento de 1.

Bit de sinal (+) Forma binária direta

Bit de sinal (-) Complemento de 2 1

Representação de números com sinal

3) forma complemento de 2.

Bit de sinal (+) Forma binária direta

Bit de sinal (-) Complemento de 2

Leitura indicada

Maini, A.K. “Digital Electronics – Principles and Integrated Circuits”

a) Sec. 1.1 – 1.15 , pgs. 1 – 10

b) Sec. 2.1 – 2.7 , pgs. 47 – 69

1 Number Systems and Codes

LEARNING OBJECTIVES

After completing this chapter, you will learn the following:

- Difference between analogue and digital ways of representing data.
- Basic concepts underlying the decimal, binary, octal, and hexadecimal number systems.
- Conversion of a given number in one number system to its equivalent in another number system.
- Representation of positive and negative numbers.
- Data representation using floating-point notation.
- Difference between weighted and unweighted binary codes.
- BCD code along with BCD-to-binary and binary-to-BCD conversion.
- Excess-3 code with excess-3-to-decimal and decimal-to-excess-3 conversion.
- Gray code with Gray-to-binary and binary-to-Gray conversion.
- Alphanumeric codes including ASCII, EBCDIC and seven-segment display codes.
- Error detection and correction codes.
- Unicode.
- Hamming code.

2 Digital Arithmetic

LEARNING OBJECTIVES

After completing this chapter, you will learn the following:

- Basic rules of binary addition and subtraction.
- Two's complement method of binary addition and subtraction.
- Addition and subtraction of BCD numbers.
- Binary multiplication and division.
- Addition, subtraction, multiplication, and division of binary floating-point numbers.