
DEPEL - Departamento
de Engenharia Elétrica

# - Arquitetura de sistemas digitais-

# Cap 1 - Introdução

Prof. Alan Petrônio - www.ufsj.edu.br/alan - 2012

- Conteúdos/propósitos deste capítulo:

1.1 - Relembrando conceitos

1.2 - Arquitetura de computadores

1.3 - Sistemas embarcados

1.4 - Arquitetura de microcontroladores

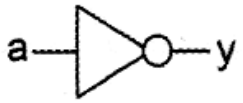
1.5 - Processadores digitais de sinais

1.6 - Arquitetura de dispositivos lógicos programáveis

# 1.1 – Relembrando conceitos

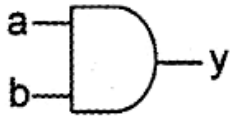
- Aritmética binária
- Portas lógicas

Inversor



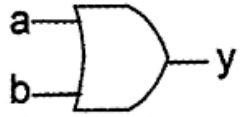

| a | y |
|---|---|
| 0 | 1 |
| 1 | 0 |

AND




| a b | y |
|---|---|
| 0 0 | 0 |
| 0 1 | 0 |
| 1 0 | 0 |
| 1 1 | 1 |

OR




| a b | y |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 1 |

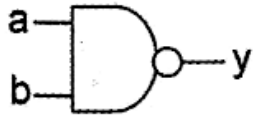
XOR


| a b | y |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |

Buffer

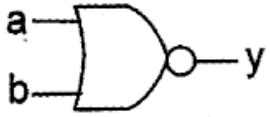

| a | y |
|---|---|
| 0 | 0 |
| 1 | 1 |

NAND




| a b | y |
|---|---|
| 0 0 | 1 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |

NOR

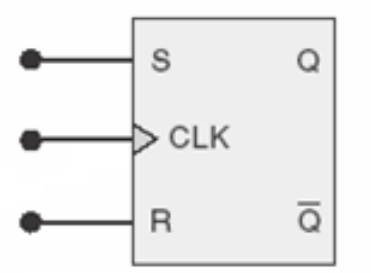



| a b | y |
|---|---|
| 0 0 | 1 |
| 0 1 | 0 |
| 1 0 | 0 |
| 1 1 | 0 |

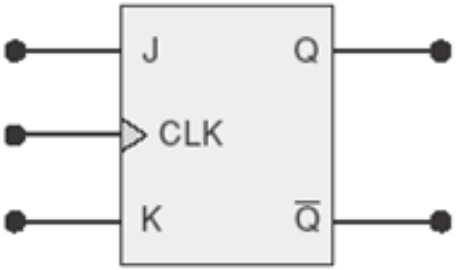
XNOR


| a b | y |
|---|---|
| 0 0 | 1 |
| 0 1 | 0 |
| 1 0 | 0 |
| 1 1 | 1 |

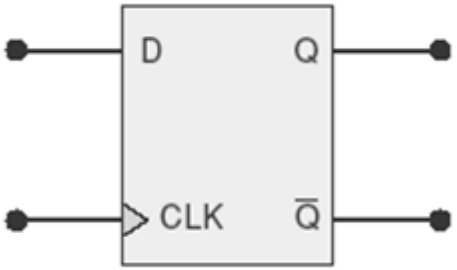
- Os Flip flops (FF)




| Entradas | | | Saída |
|---|---|---|---|
| S | R | CLK | Q |
| 0 | 0 | ↑ | $Q_0$ (Não muda) |
| 1 | 0 | ↑ | 1 |
| 0 | 1 | ↑ | 0 |
| 1 | 1 | ↑ | Ambíguo |

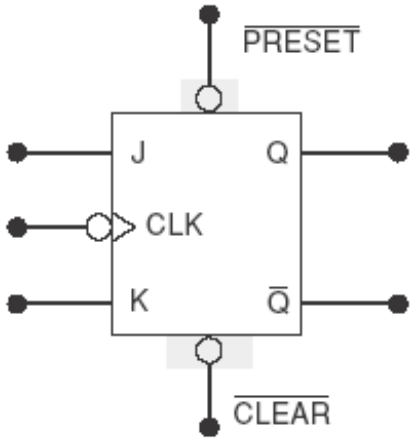



| J | K | CLK | Q |
|---|---|---|---|
| 0 | 0 | ↑ | $Q_0$ (não muda) |
| 1 | 0 | ↑ | 1 |
| 0 | 1 | ↑ | 0 |
| 1 | 1 | ↑ | $\overline{Q_0}$ (comuta) |



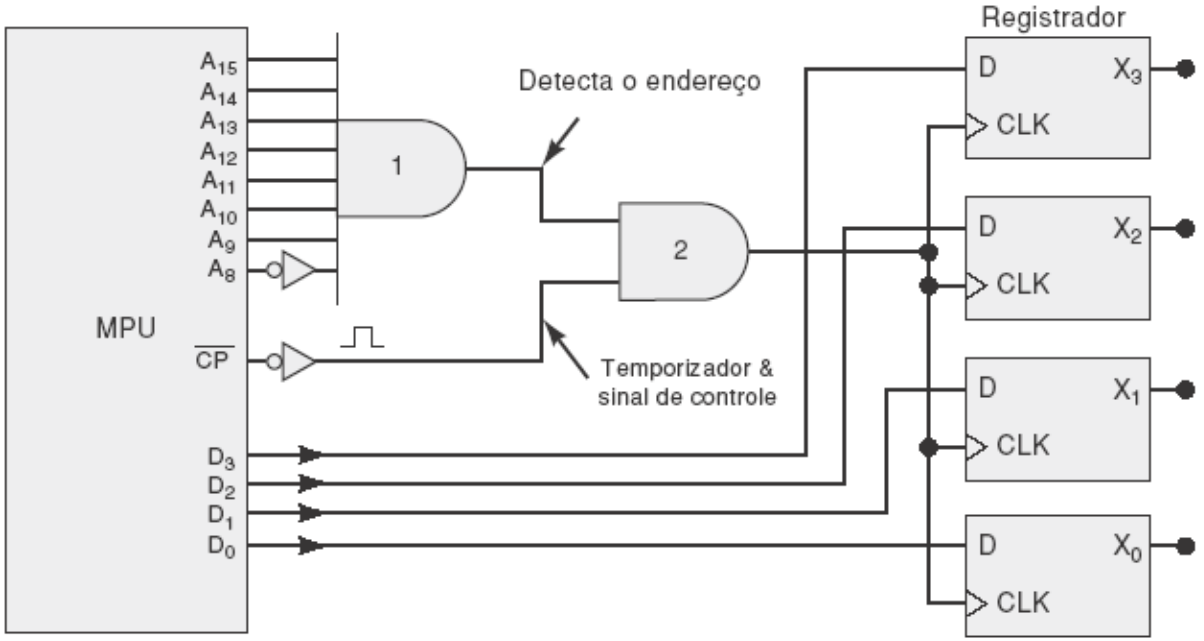

| D | CLK | Q |
|---|---|---|
| 0 | ↑ | 0 |
| 1 | ↑ | 1 |

- Sinais de controle síncronos e assíncronos

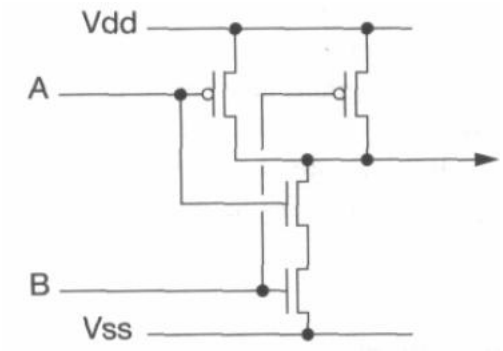


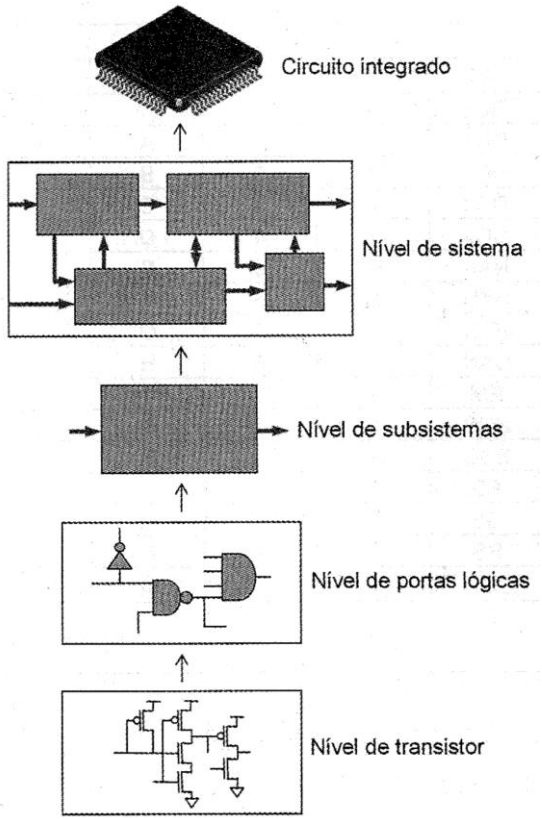

| J | K | CLK | $\overline{PRE}$ | $\overline{CLR}$ | Q |
|---|---|---|---|---|---|
| 0 | 0 | ↓ | 1 | 1 | Q (não muda) |
| 0 | 1 | ↓ | 1 | 1 | 0 (reset síncrono) |
| 1 | 0 | ↓ | 1 | 1 | 1 (set síncrono) |
| 1 | 1 | ↓ | 1 | 1 | $\overline{Q}$ ( comutação síncrona) |
| x | x | x | 1 | 1 | Q (não muda) |
| x | x | x | 1 | 0 | 0 (clear assíncrono) |
| x | x | x | 0 | 1 | 1 (preset assíncrono) |
| x | x | x | 0 | 0 | (Inválido) |

- Exemplo:



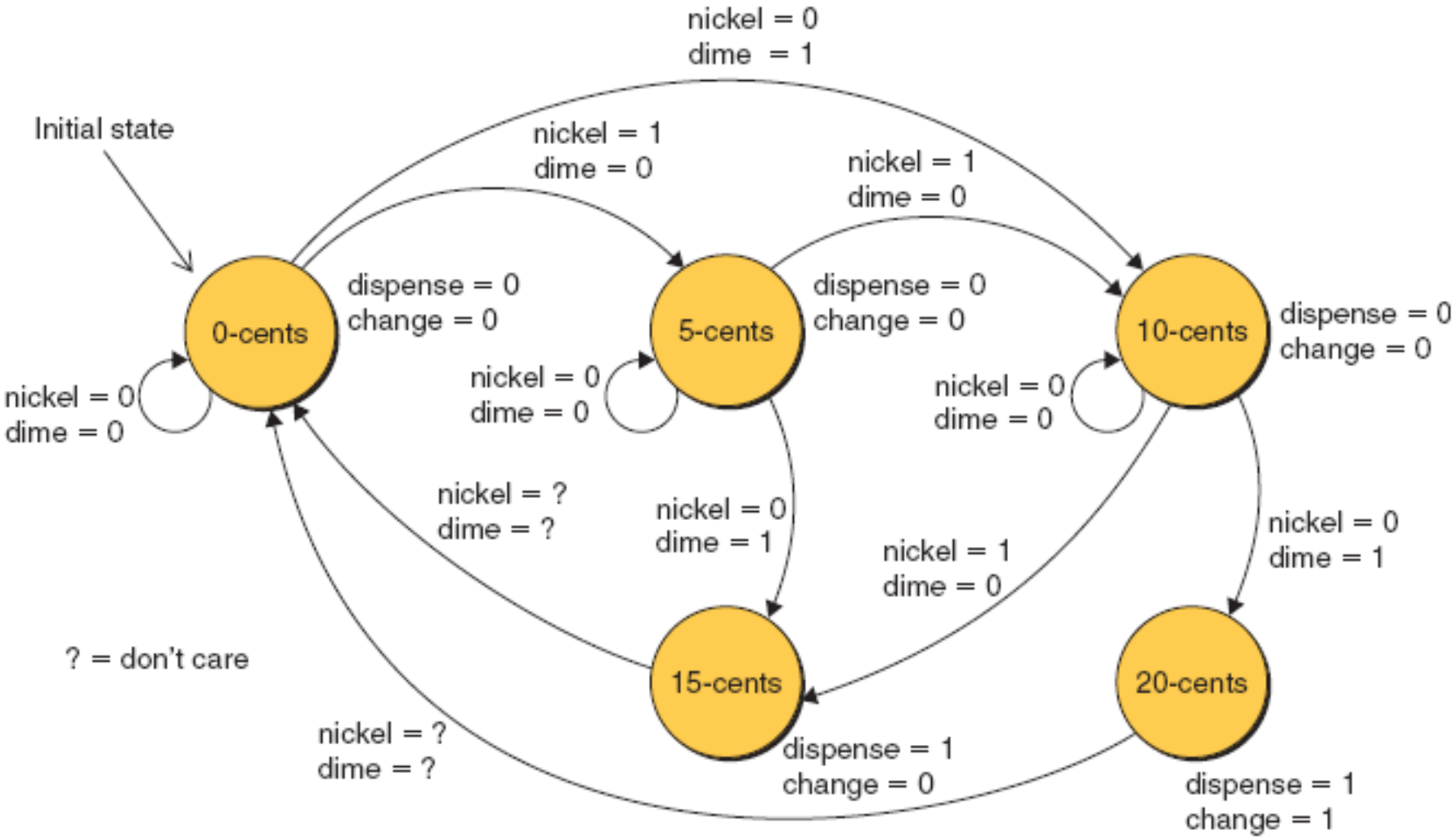
- Níveis de abstração de um sistema digital



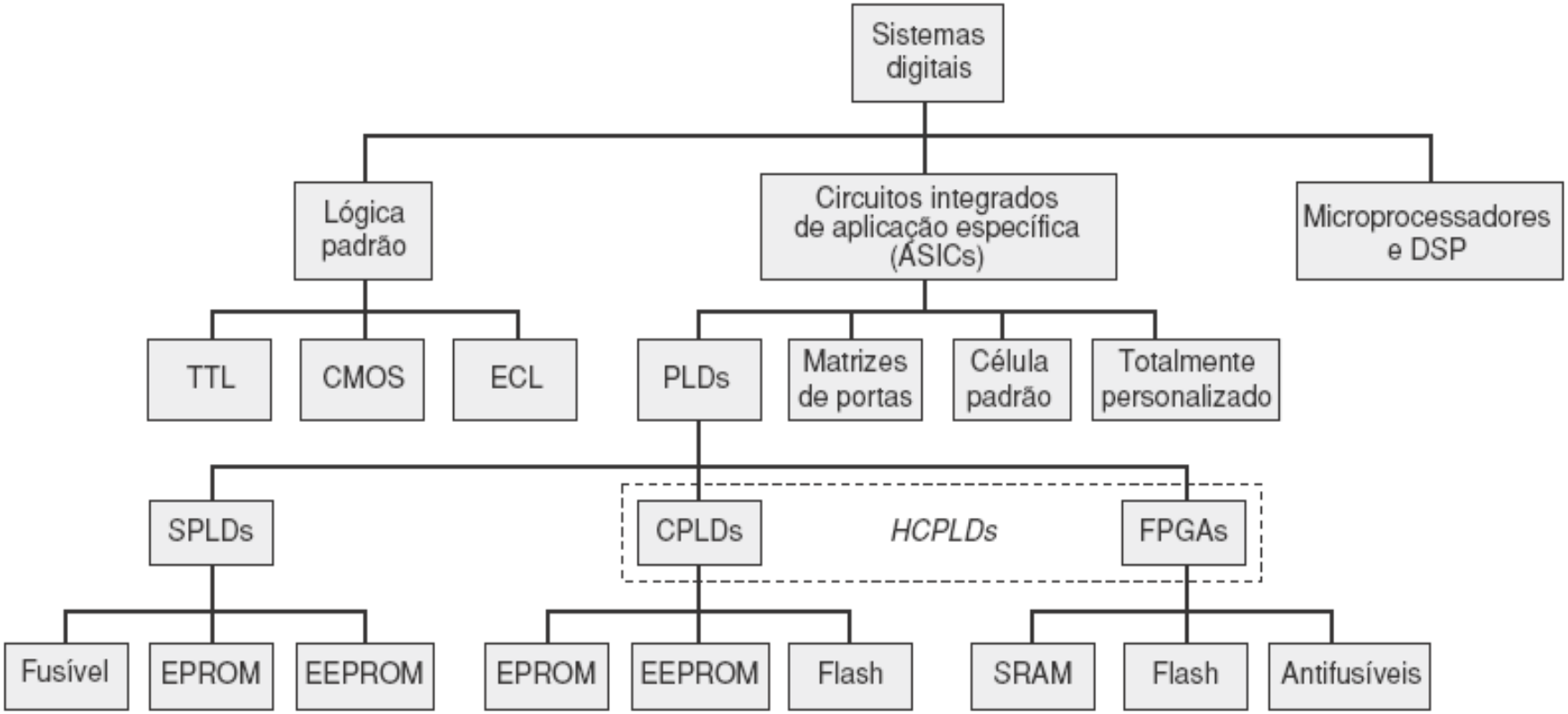




- "Classificação" dos CI´s quanto à "programabilidade"
  - CI não programável
  - CI programável (instruções/software)
  - CI com hardware programável (linguagens HDL)

- Máquinas de estados



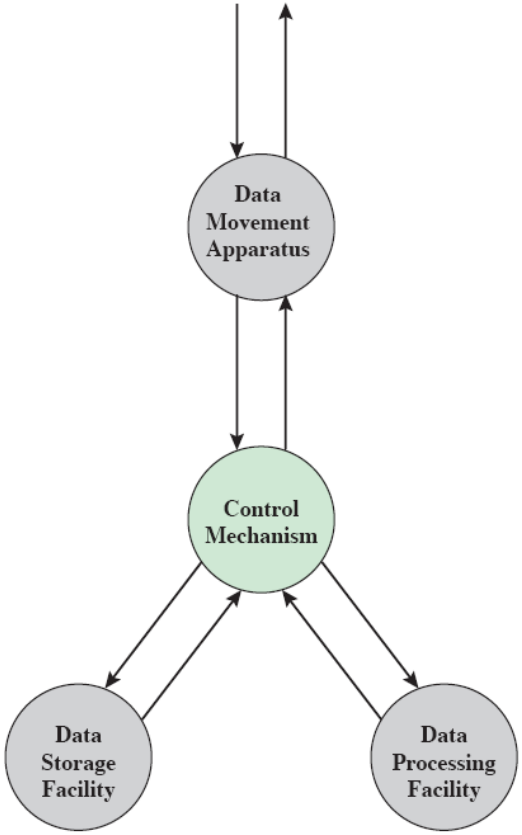
- Tecnologias de sistemas digitais

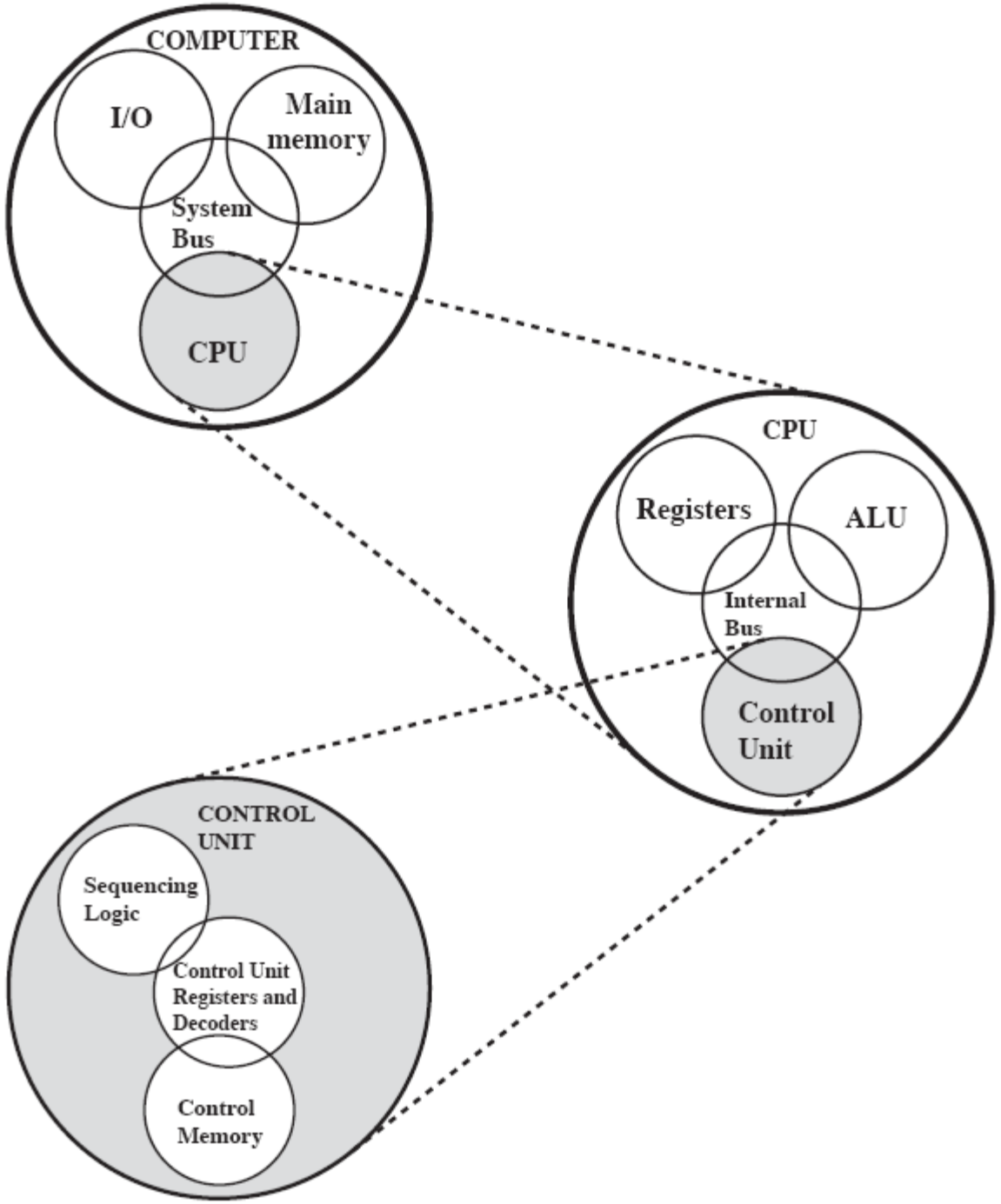


# 1.2 Arquitetura de computadores

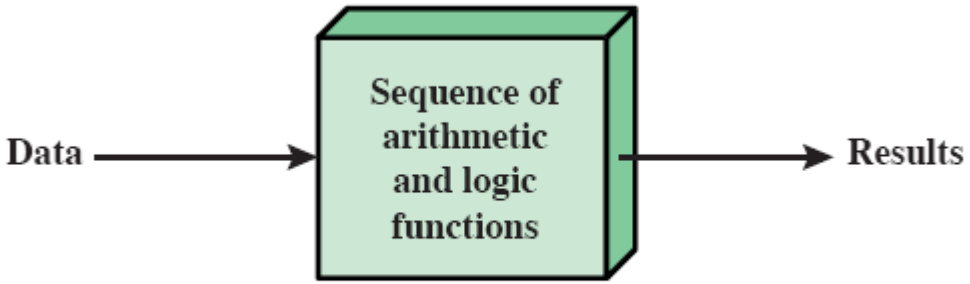
- Arquitetura
  - atributos que possuem impacto sobre a execução lógica de instruções
- Organização
  - unidades operacionais e suas interconexões => realizam especificações arquiteturais
- Mesma arquitetura mas diferente organização !

- Funcionabilidade de um sistema de computador
  - processamento de dados;
  - armazenamento de dados;
  - movimentação de dados;
  - controle.

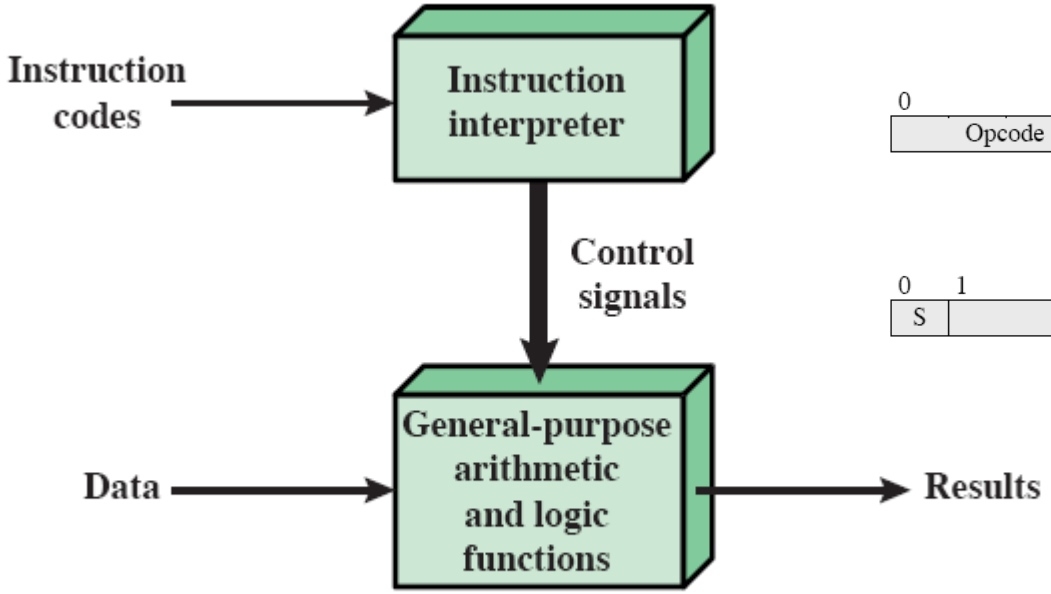



COMPUTER
I/O
Main memory
System Bus
CPU
CPU
Registers
ALU
Internal Bus
Control Unit
CONTROL UNIT
Sequencing Logic
Control Unit Registers and Decoders
Control Memory

- Hardware x software

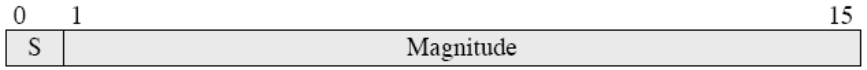



(a) Programming in hardware



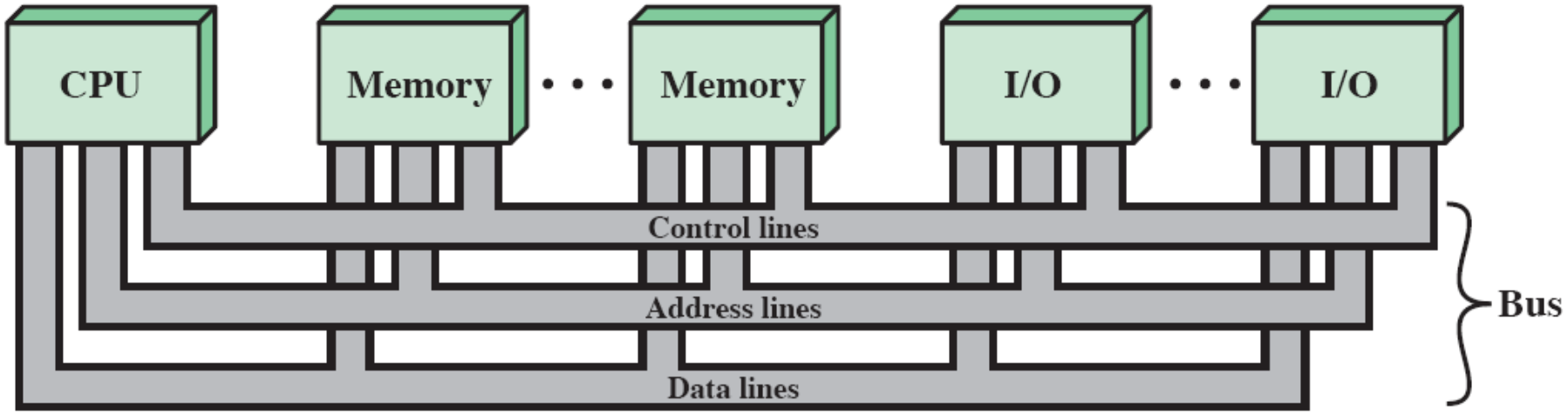

(b) Programming in software


(a) Instruction format



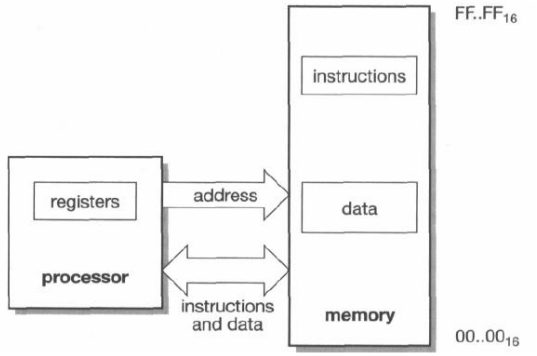

(b) Integer format

- Protocolos/arbitração



# 1.3 - Arquitetura de microcontroladores

- Processador: CI que executa instruções em memória

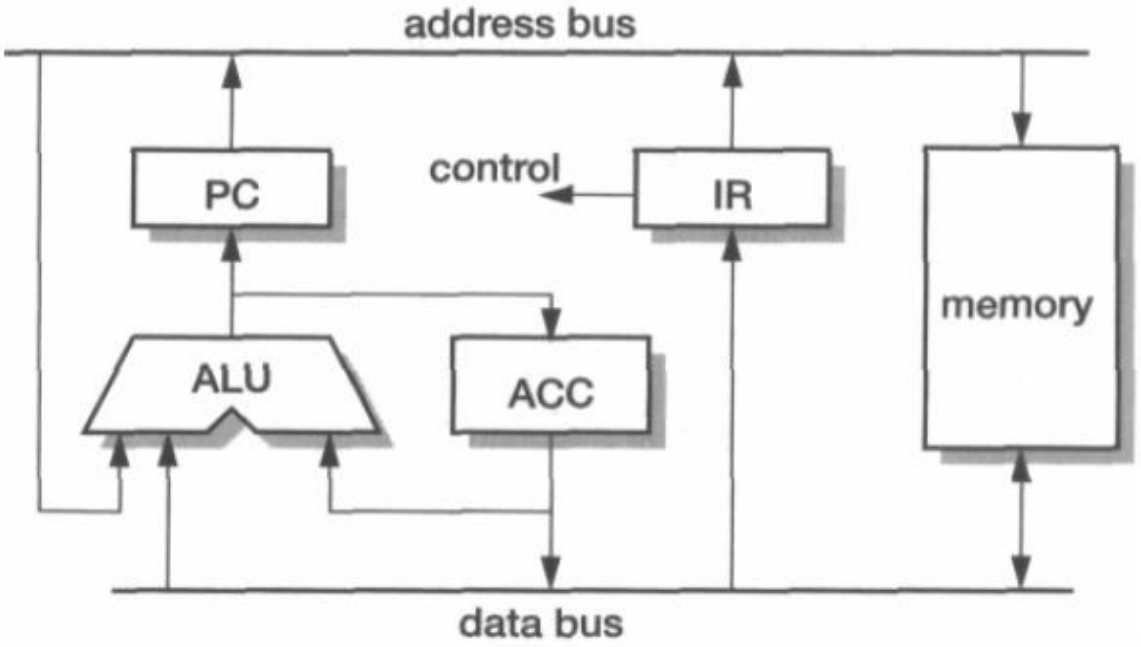



- Princípio do “programa armazenado em memória”
  - Instrução e dados na mesma memória
  - Flexibilidade e universalidade
- Avanços
  - Eletrônica: transistores efeito-campo e tecnologia VLSI
  - Arquitetura/organização microprocessadores
    - Pipeline; cache; busca antecipada, RISC/CISC, etc ...

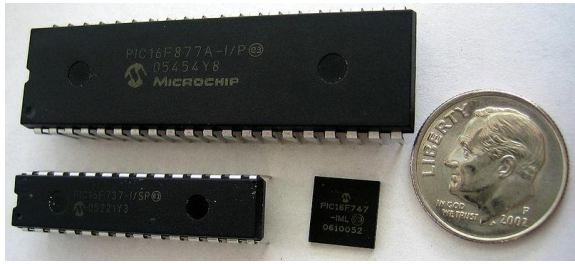
- A hierarquia em diferentes níveis de abstrações ...
  - Transistores
  - Portas lógicas, flip-flops
  - Somadores simples, multiplexadores, decoders ...
  - Somadores completos, registradores de deslocamento, barramentos, multiplicadores ...
  - ULA, bancos de memória/registradores ...
  - Cache, gerenciador de programa e unidade de controle
  - Processador
  - Periféricos (conversores AD/DA, memórias 'externas', drivers, USB, etc ...) => ***MICROCONTROLADORES (MCU)***
  - Sistemas integrados em CI´s
- "Instruction set"

- Exemplo:

| Instruction | Opcode | Effect |
|---|---|---|
| LDA S | 0000 | ACC := $mem_{16}$[S] |
| STO S | 0001 | $mem_{16}$[S] := ACC |
| ADD S | 0010 | ACC := ACC + $mem_{16}$[S] |
| SUB S | 0011 | ACC := ACC – $mem_{16}$[S] |
| JMP S | 0100 | PC := S |
| JGE S | 0101 | if ACC ≥ 0 PC := S |
| JNE S | 0110 | if ACC ≠ 0 PC := S |
| STP | 0111 | stop |



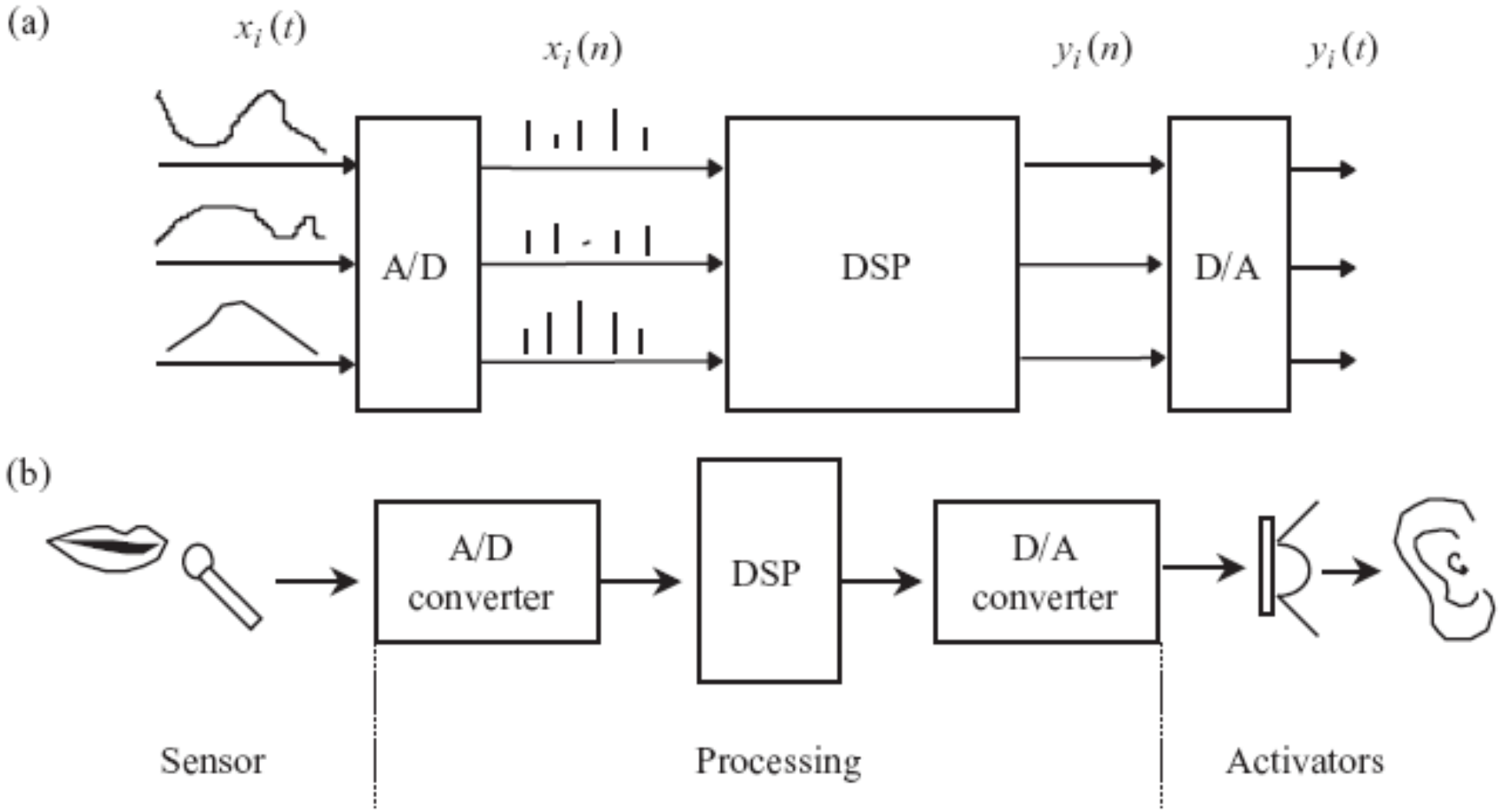
- Diversos modelos microcontroladores
  - PIC (12,16,18), Z80, 8051, ARMs ...
  - ARM
    - 10 MHz a 1.5 GHz
    - 8 a 64 BITS
    - Cache “on-chip” (até 64KB)
    - 12 a >200 pinos
    - Sistemas operacionais (Andoird , Integrity, Linux, Neutrino, Tornado, Windows CE e dereivações)
    - Até 3.000 MIPS (milhões de instruções por segundo)
      - PIC: 80 MIPS a 80MHz (máximo)




# 1.4 - Processadores digitais de sinais

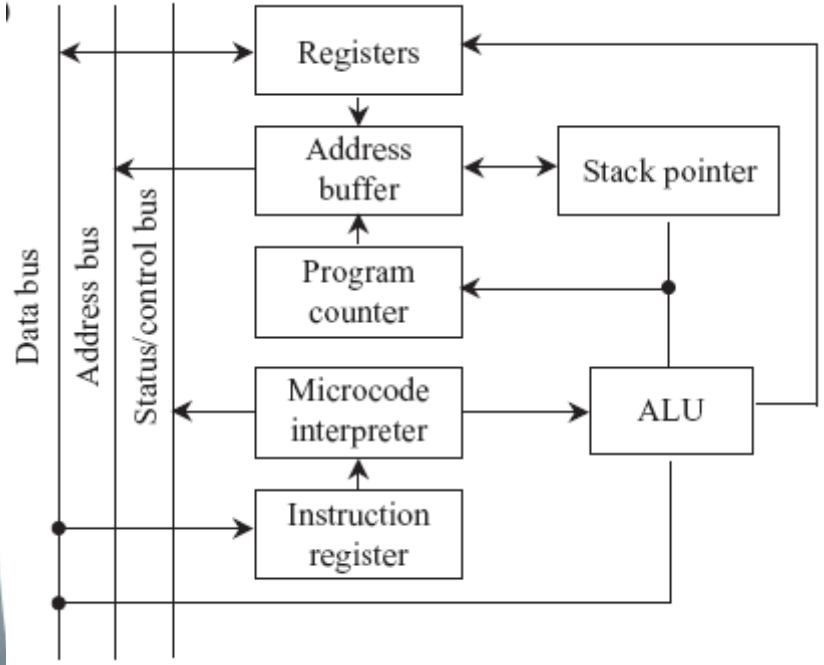
- DSP = digital signal processor
  - Entradas e saídas discretas: y[n] = *L*{ x[n] }
- Década 70
  - filtros e FFT
- Século XXI
  - telecomunicações (mobile) metade aplicações DSP
- Capacidade de:
  - Operar com milhões de amostrar por segundo
  - Grande largura de memória
  - Grande demanda computacional
  - Requerimentos em tempo real que extrapolam MCU

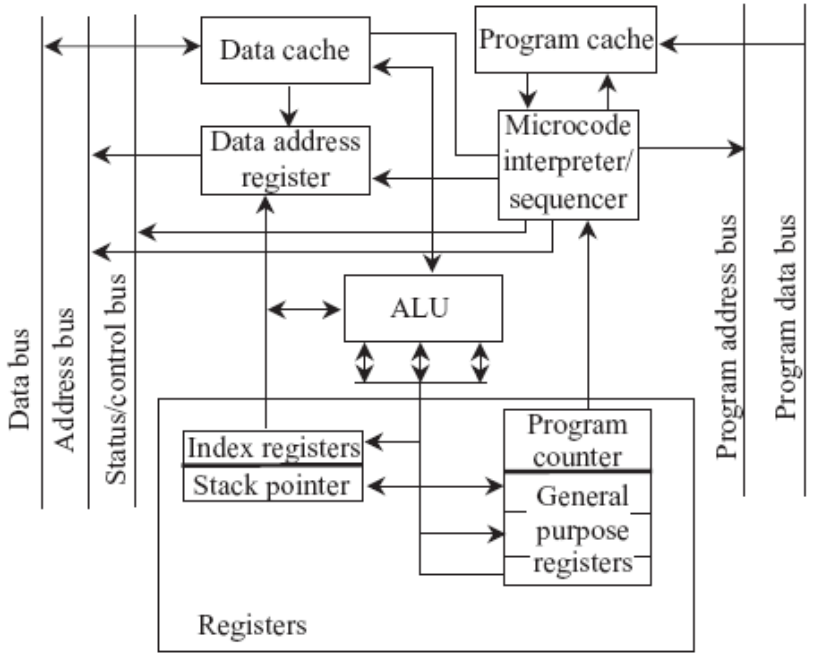
- Exemplo


(a)
$x_i(t)$
$x_i(n)$
$y_i(n)$
$y_i(t)$
A/D
DSP
D/A
(b)
A/D converter
DSP
D/A converter
Sensor
Processing
Activators

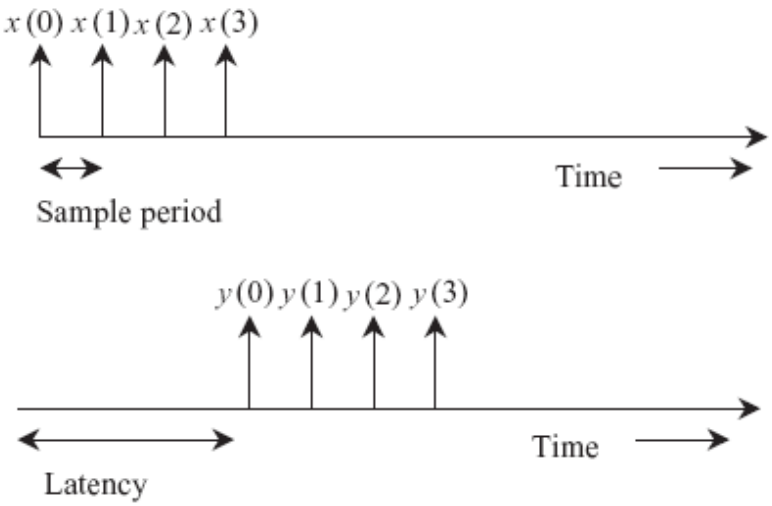
- Otimizado para :
  - convoluções de resposta ao impulso (soma produtos);
  - resposta ao impulso recursiva (filtragem)
  - FFT;
  - outras operações/algoritmos típicos de PDS;
  - interface otimizada para portas externas (tempo real).

- RISC x CISC
  - Operações simples (+, -, shift): um ciclo
  - Operações complexas (x, /): vários ciclos
  - RISC: ± 1 ciclo (instruções mais “rápidas”)
  - CISC: vários ciclos com múltiplas instruções (micro-código ‘hard coded’)
- DSP x microcontroladores



- “Tempo real”
  - Latência do processador
  - Período de amostragem



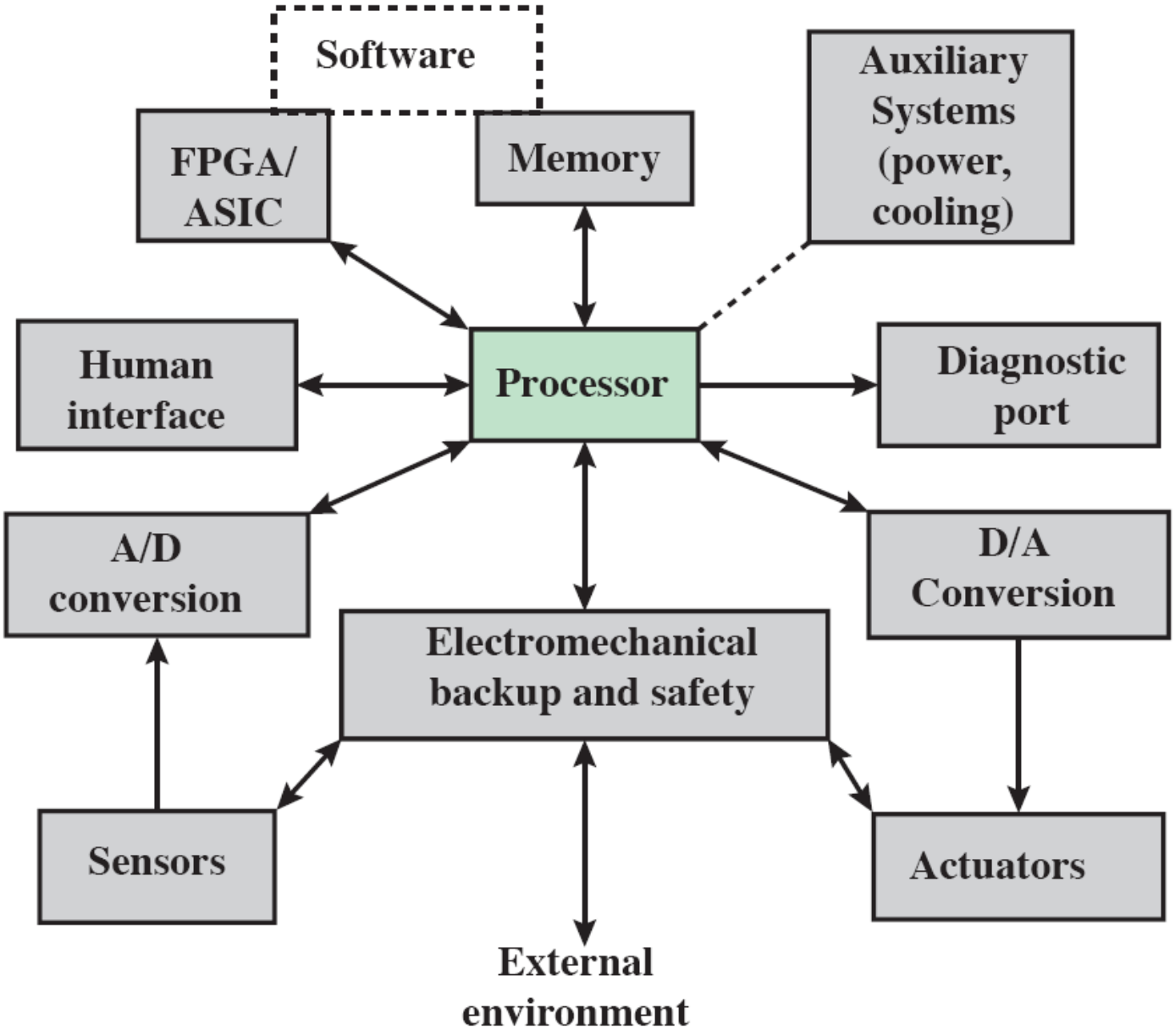

- Integração em CI de MCU + DSP

# 1.5 – Sistemas embarcados

- "Sistema computacional aplicado"
  - Avanços na tecnologia;
  - diminuição do custo;
  - variados níveis de componentes (software/hardware)

- Falsas verdades:
  - Sistemas embarcados são mais "limitados" em funcionabilidade de hardware/software
  - Tarefas dedicadas;
  - Sistema de computação com maior *qualidade*, *confiança* e *segurança*
  - *alguns dispositivos tais (e.g., PDA) não são realmente sistemas embarcados*

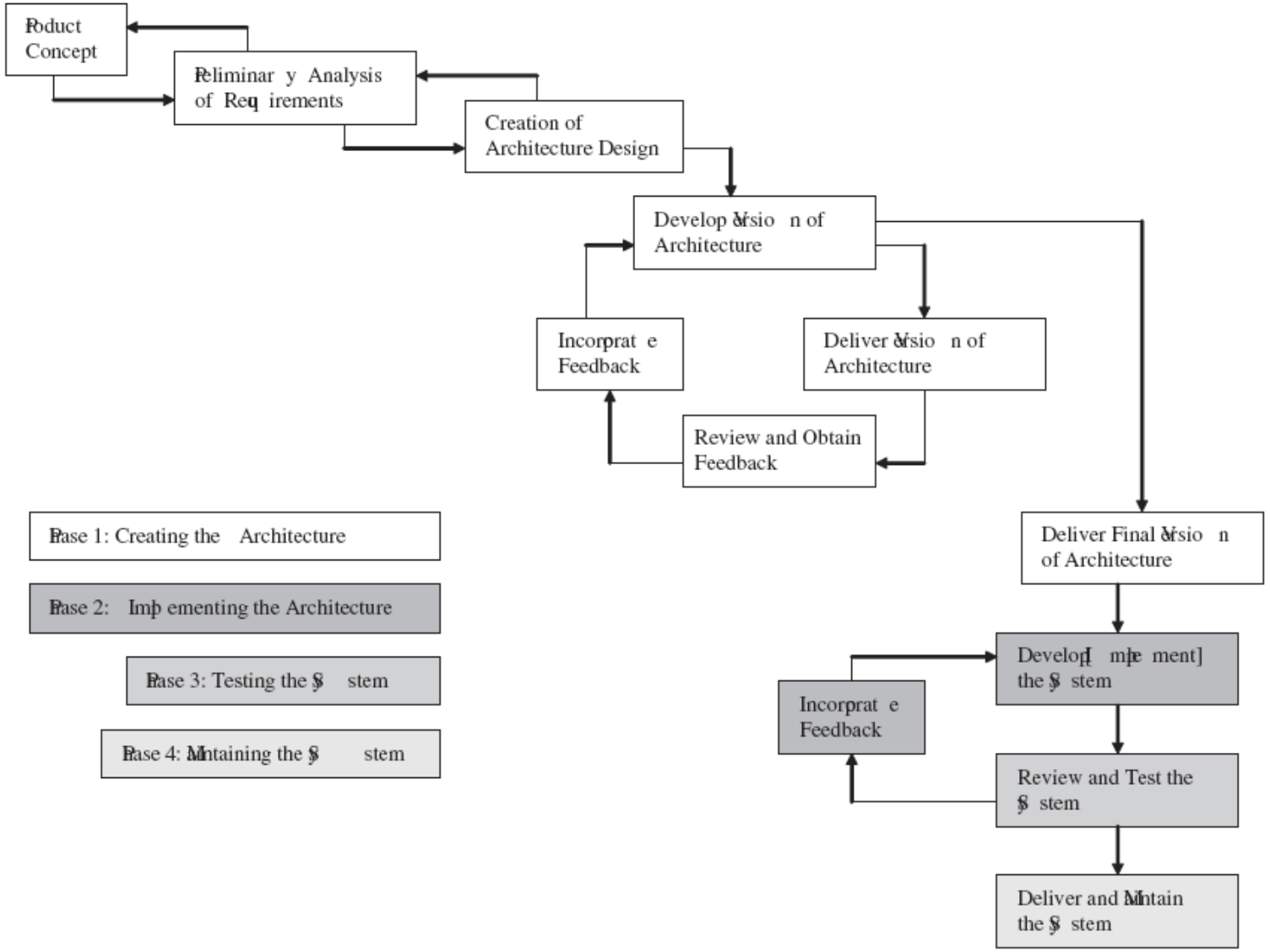
| Market | Embedded Device |
|---|---|
| Automotive | Ignition System |
| | Engine Control |
| | Brake System (i.e., Antilock Braking System) |
| Consumer Electronics | Digital and Analog Televisions |
| | Set-Top Boxes (DVDs, VCRs, Cable Boxes, etc.) |
| | Personal Data Assistants (PDAs) |
| | Kitchen Appliances (Refrigerators, Toasters, Microwave Ovens) |
| | Automobiles |
| | Toys/Games |
| | Telephones/Cell Phones/Pagers |
| | Cameras |
| | Global Positioning Systems (GPS) |
| Industrial Control | Robotics and Control Systems (Manufacturing) |
| Medical | Infusion Pumps |
| | Dialysis Machines |
| | Prosthetic Devices |
| | Cardiac Monitors |
| Networking | Routers |
| | Hubs |
| | Gateways |
| Office Automation | Fax Machine |
| | Photocopier |
| | Printers |
| | Monitors |
| | Scanners |


Software
FPGA/
ASIC
Memory
Auxiliary
Systems
(power,
cooling)
Human
interface
Processor
Diagnostic
port
A/D
conversion
D/A
Conversion
Electromechanical
backup and safety
Sensors
Actuators
External
environment

- Projetos da ARQUITETURA de um sistema embarcado:
  - Definir e entender o projeto
  - Limitar custos
  - Requerimentos de segurança/confiança
  - Limitações estruturais (bateria, memória,etc)
  - Interfacear módulos e suas relações

- Ciclos de projeto:
  - Magaiver: chiclete + clips + braço
  - Code-and-fix: definição de requerimentos sem processos antes do desenvolvimento
  - Waterfall: desenvolvimento em etapas onde o resultado de uma etapa subsidia as demais
  - Espiral: desenvolvimento em etapas + feedback

- Projeto de um sistema embarcado e seu ciclo de vida

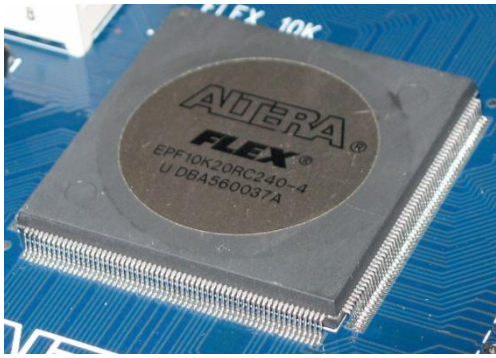


# 1.6 - Dispositivos lógicos programáveis

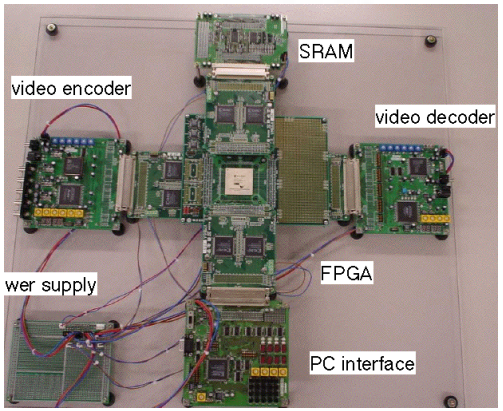
- Motivação:
  - "Problemas" de sistemas digitais microcoprocessados
    - Sequência programa de instruções
    - Necessidade de respostas mais rápidas
    - Necessidade de arquitetura dedicada
  - Integração dos circuitos discretos

- *Dispositivos de lógica programável (PLD)*
  - *Estrutura interna não é fixa*

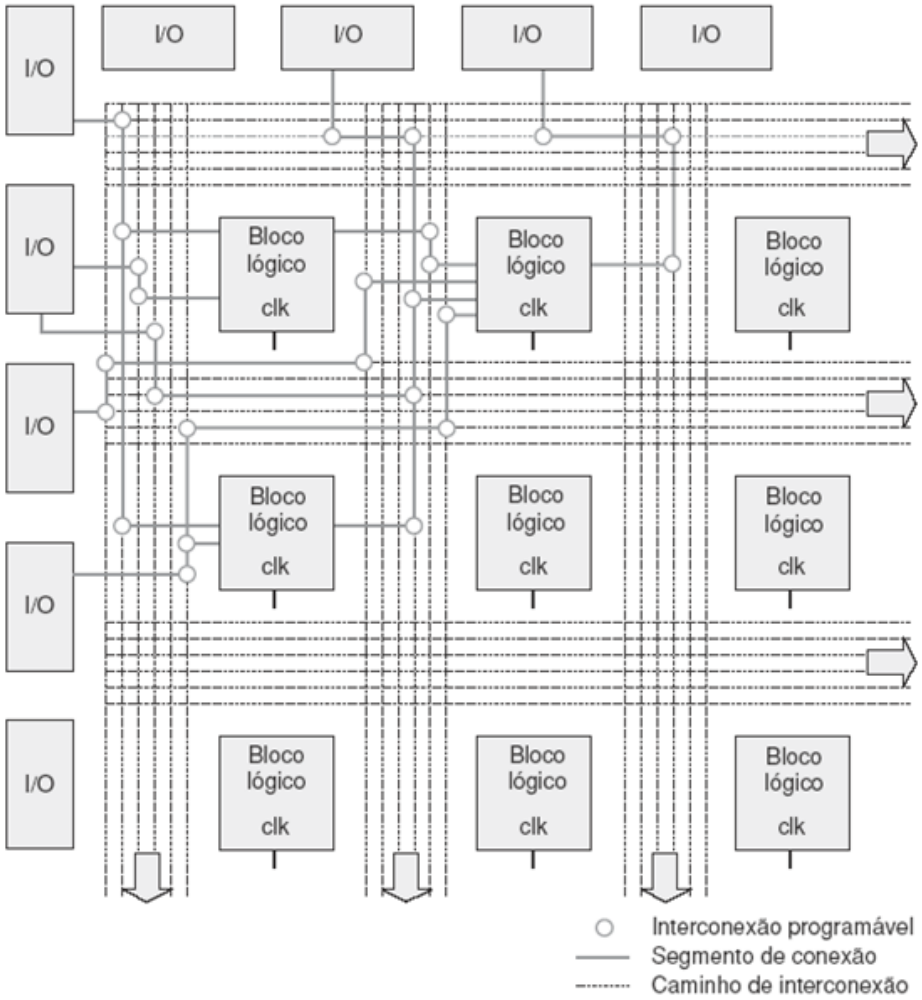
- Vantagens:
  - Flexibilidade;
  - Mesma funcionabilidade em um único CI (SoC)
  - Menor espaço
  - Menor consumo energia
  - Maior confiabilidade
  - “Menor” complexidade de desenvolvimento
  - Menor custo de fabricação (geralmente)
  - Resolução de problemas mais complexos
  - Otimização de circuitos
  - Interface com outras tecnologias e “reutilização” de “código”(IP - Intellectual Property)



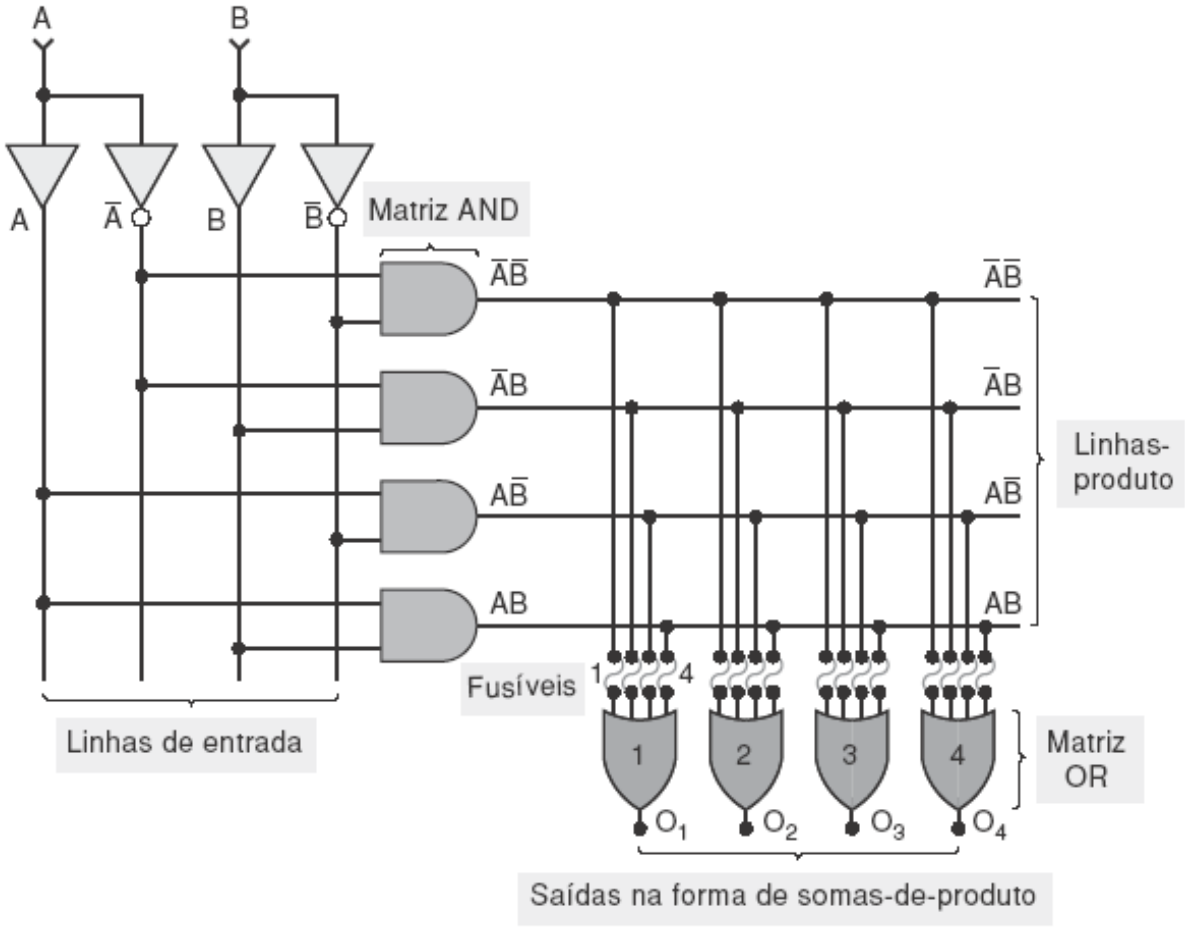
- Aplicações
  - Circuitos lógicos diversos ou sistemas digita
  - Interface de computadores
  - Microcontroladores (DSP, ARM e afins)
  - Sistemas embarcados
  - Processamento de imagem
  - Processamento de sinais
    - DSP x FPGA
  - Clustering
  - Sistemas de controle
    - CLP
  - Outras aplicações: militares, dispositivos médicos



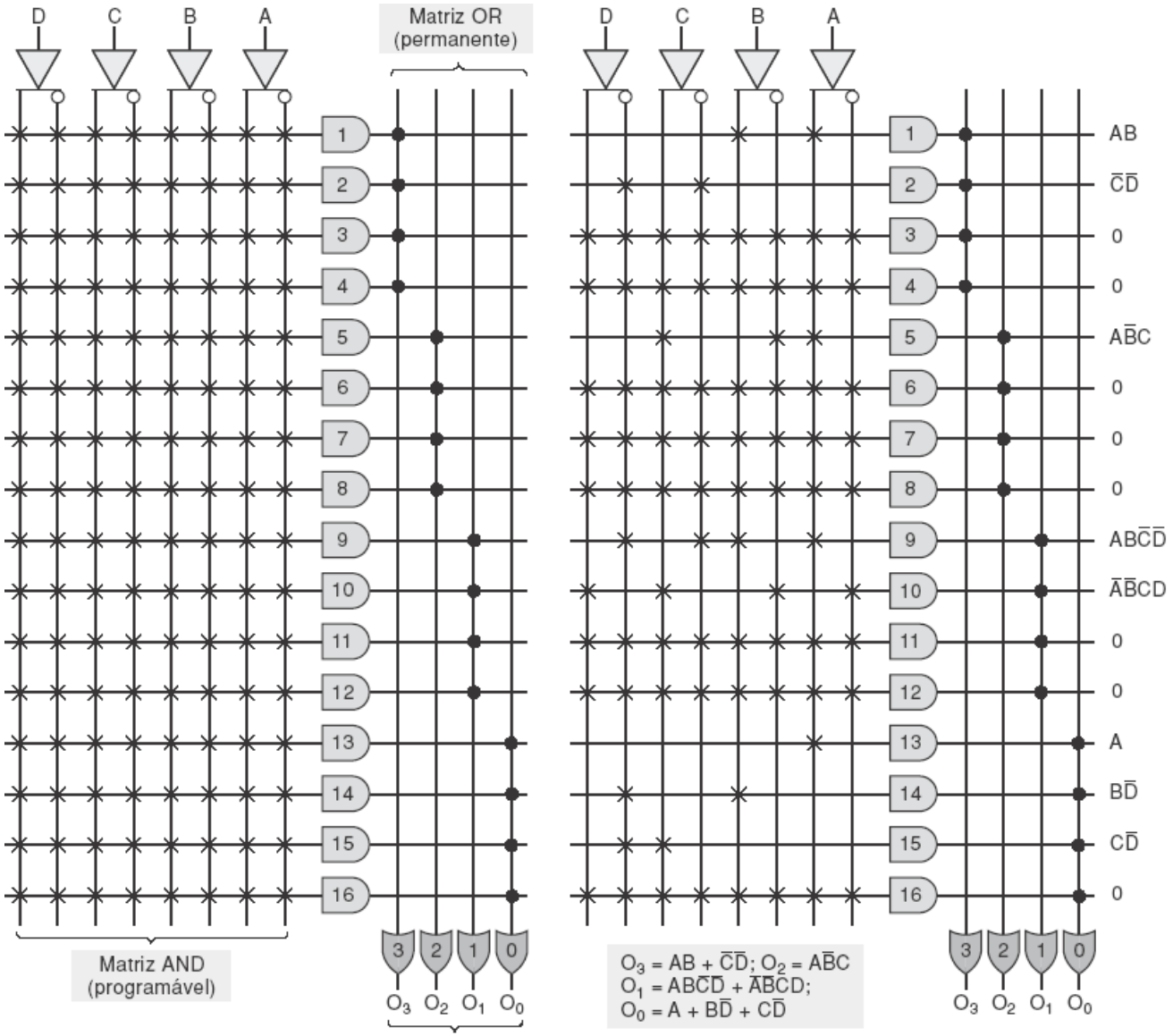

- **Bloco lógicos /Macrocélulas**
  - Qualquer circuito lógico
  - Cerca 5 variáveis de entrada
- Roteamento de sinal lógico
  - Atrasos de sinal e Roteamento
- FF da macrocélula pode ser configurado (D, JK, T, SR)
- Pinos I/O associados com registradores e latchs
- Informações de programação:
  - cada bloco lógico
  - I/O
  - Interconexão blocos
- Geralmente ISP (‘*in-system programmable’*)
- Arquitetura diversificada
  - FPGA x CPLD



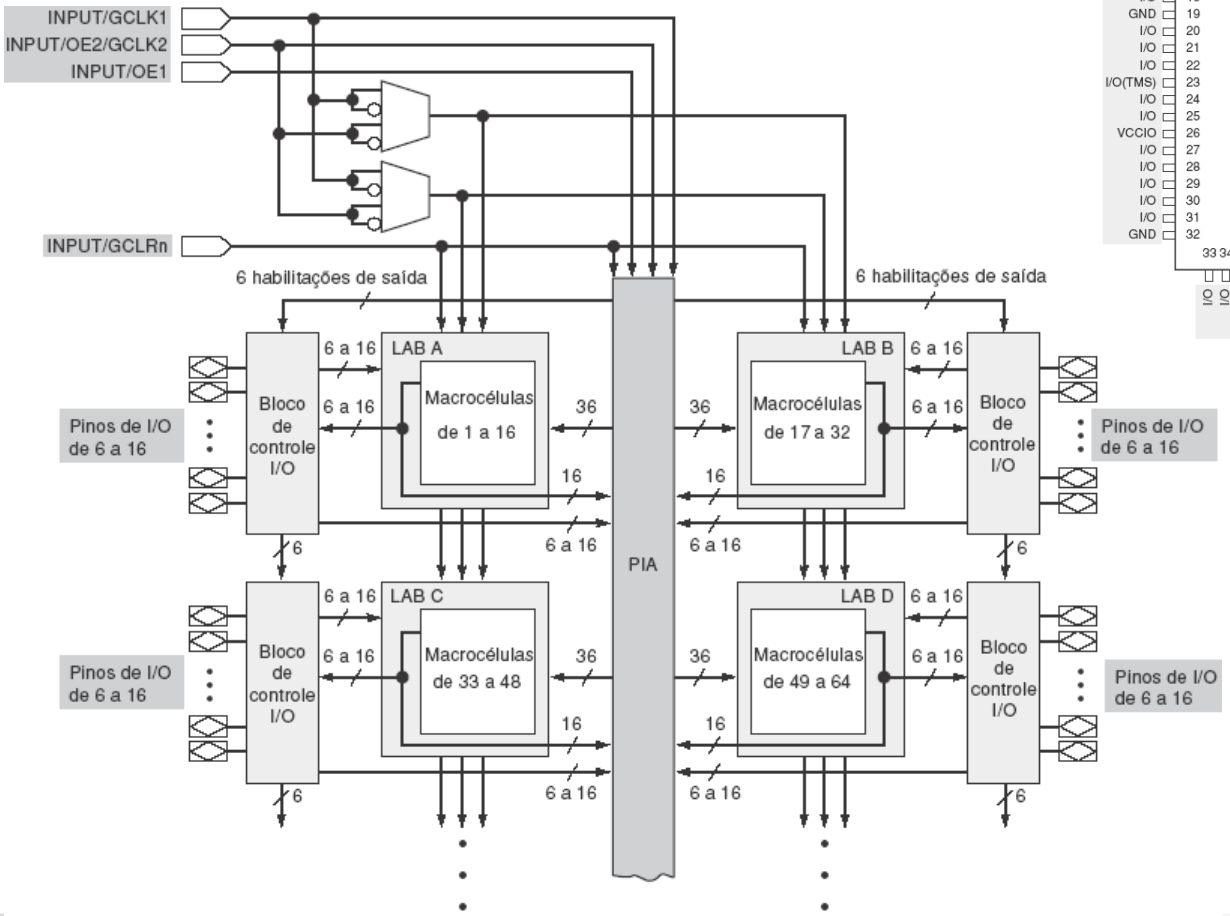
- Conexões por fusíveis
  - **Programação de conexões !!!!**
- Compilador lógico
  - Equações lógicas -> mapa de fusíveis
- Saída programada para ser qualquer função de A e B
- Implementação de função lógica na forma de soma de produtos

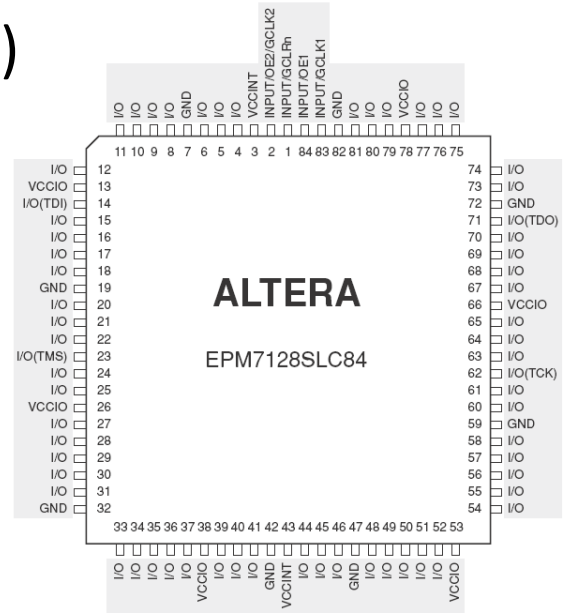


- Arquitetura PAL

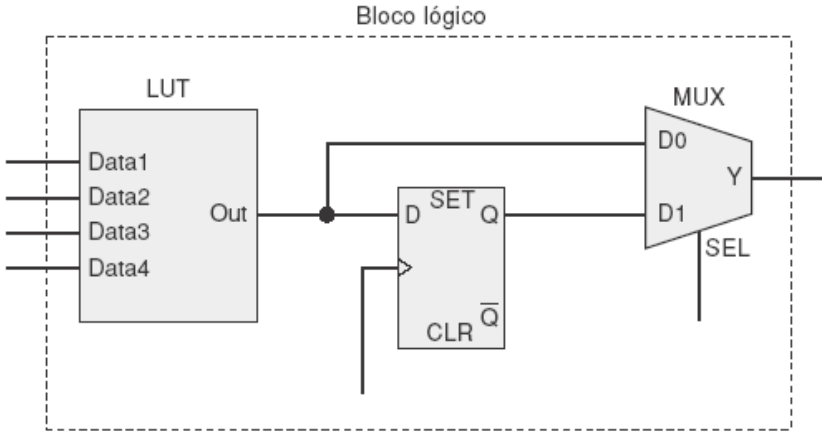


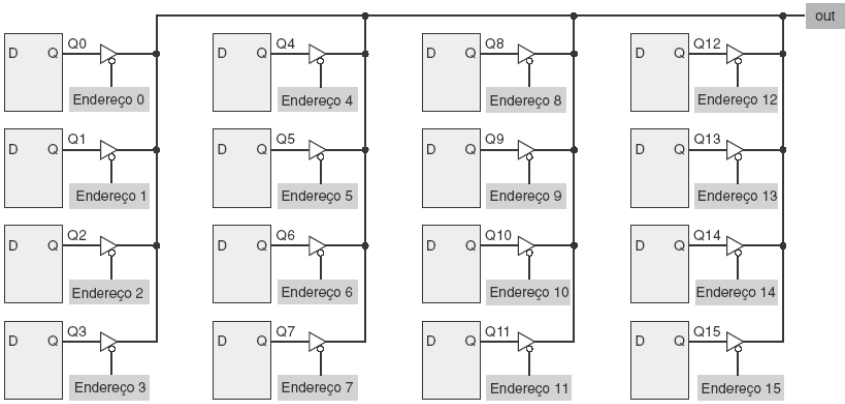
- CPLD EPM7128
  - 16 macrocélulas Podem compartilhar recursos lógicos
  - Arranjo de interconexão programável (PIA)
  - Barramento global
  - Roteamento de sinais lógicos



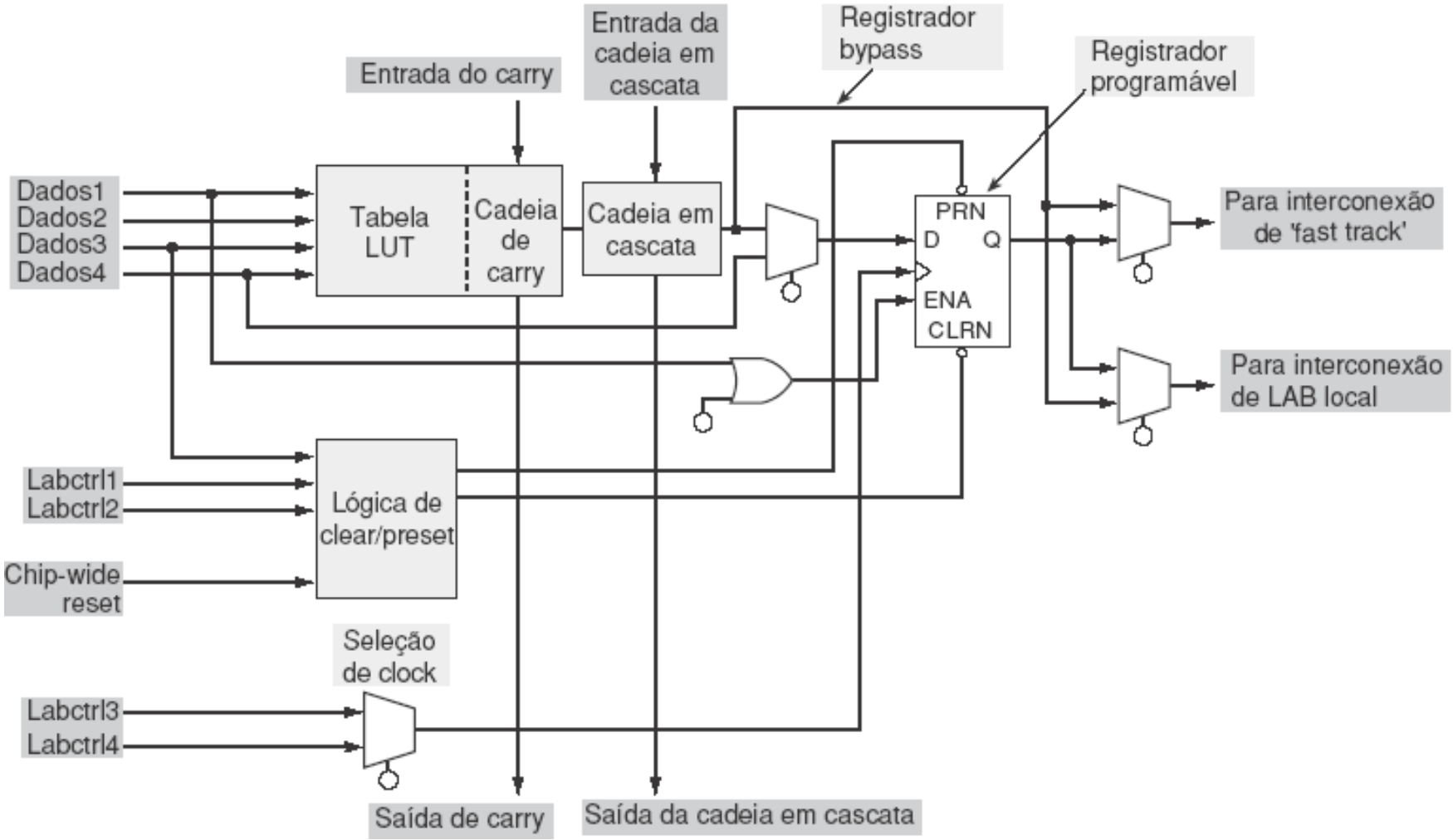



- Tabela LUT

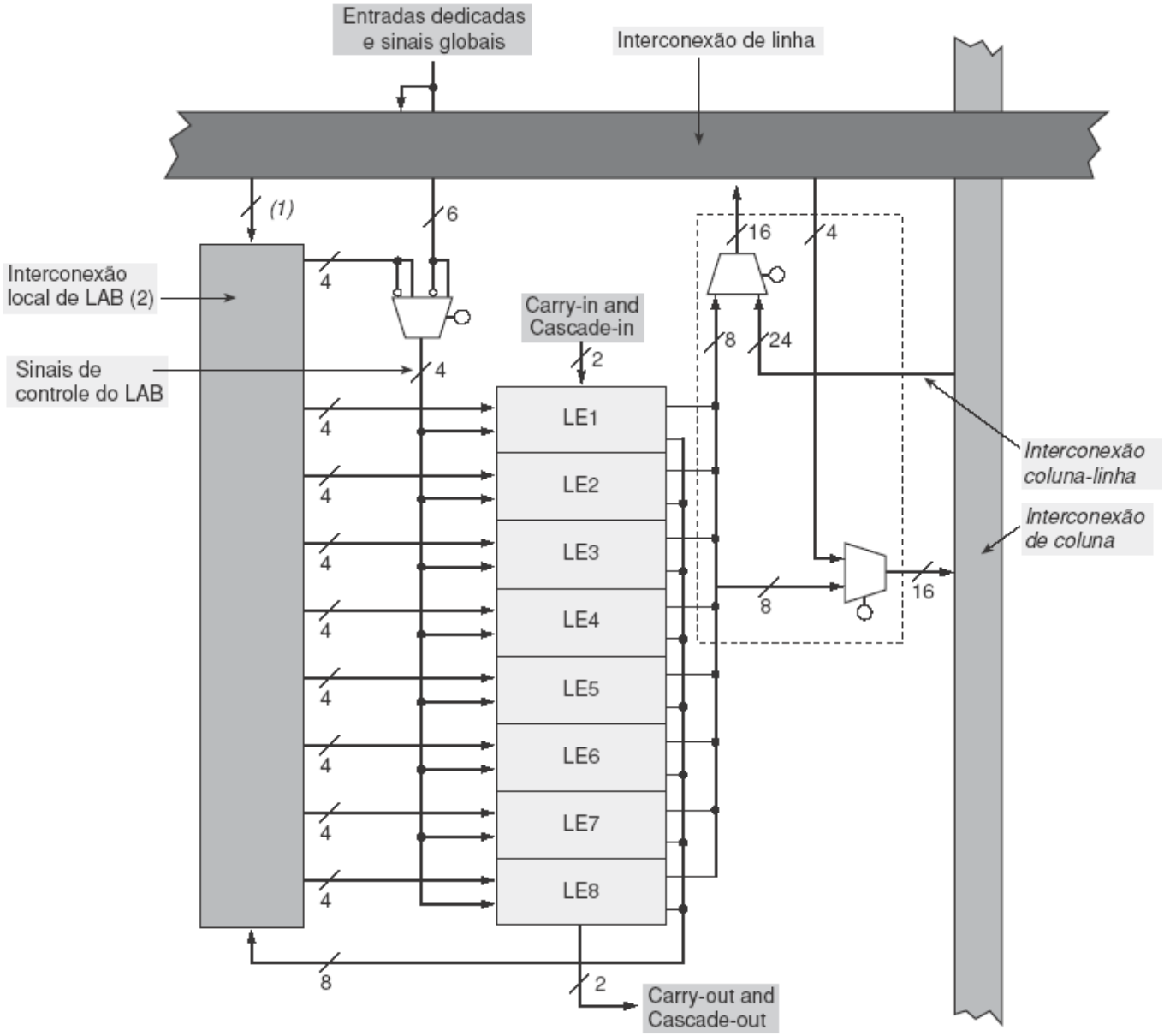

Bloco lógico
LUT
Data1
Data2
Data3
Data4
Out
SET
D
Q
CLR
Q
MUX
D0
D1
Y
SEL


out
Q0
Endereço 0
Q1
Endereço 1
Q2
Endereço 2
Q3
Endereço 3
Q4
Endereço 4
Q5
Endereço 5
Q6
Endereço 6
Q7
Endereço 7
Q8
Endereço 8
Q9
Endereço 9
Q10
Endereço 10
Q11
Endereço 11
Q12
Endereço 12
Q13
Endereço 13
Q14
Endereço 14
Q15
Endereço 15

- Elemento lógico



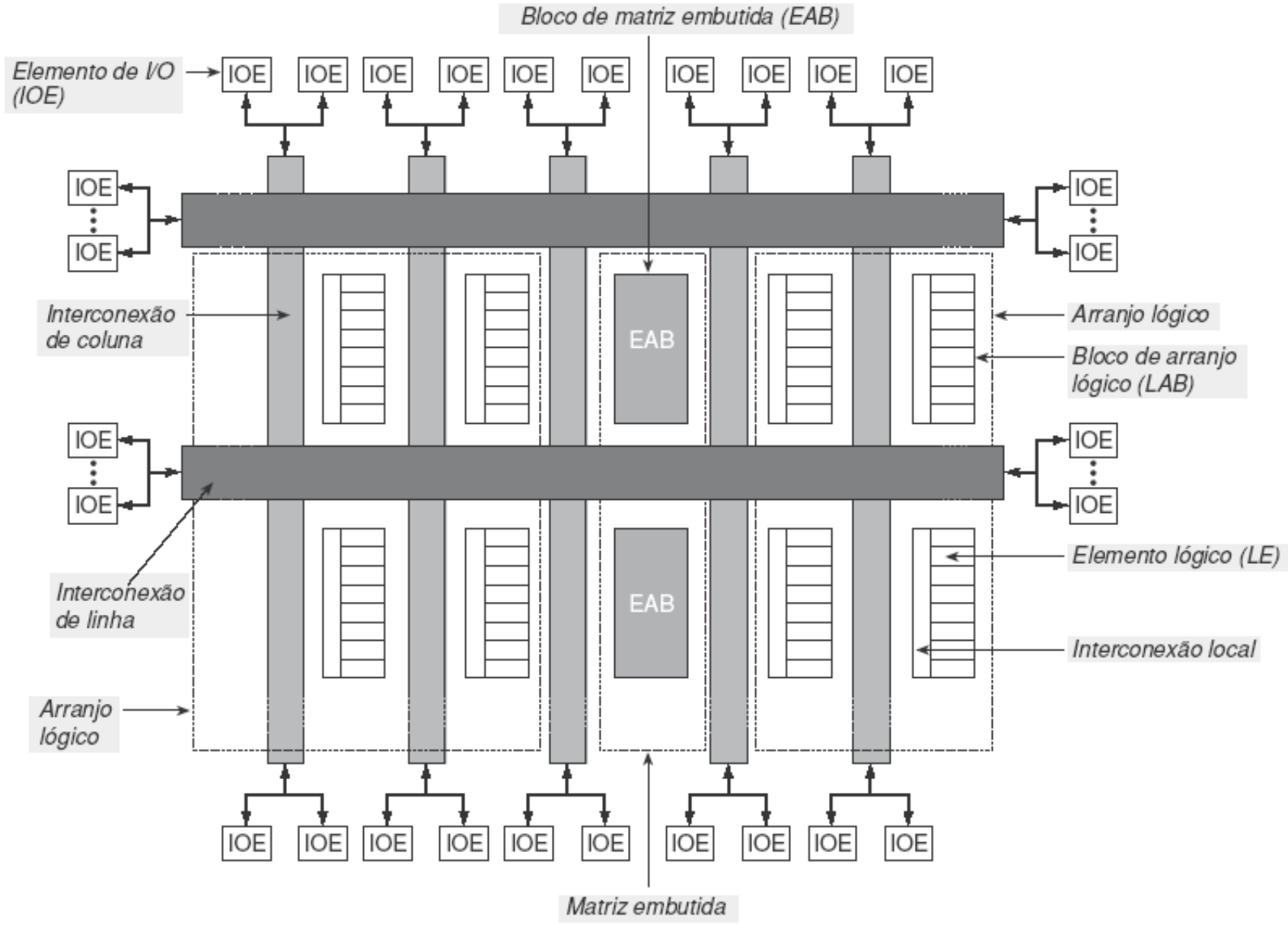
- Bloco de arranjo lógico





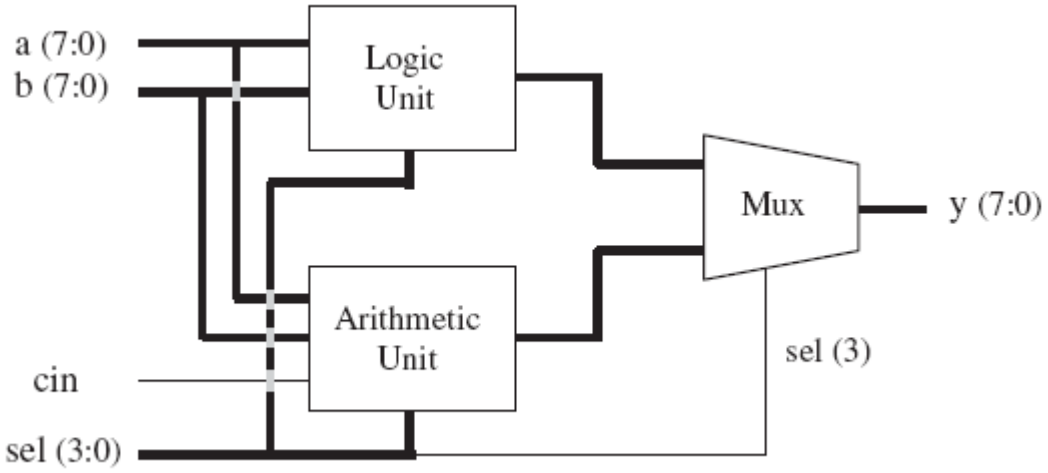

UFSJ

- Esquema geral



- Forte apelo para:
  - “Inovação” tecnológica.
  - Reaproveitamento e maior produtividade
- Verilog x VHDL
- Fabricantes:
  - Actel
  - **Altera**
  - Atmel
  - Cypress
  - **Xilinx**

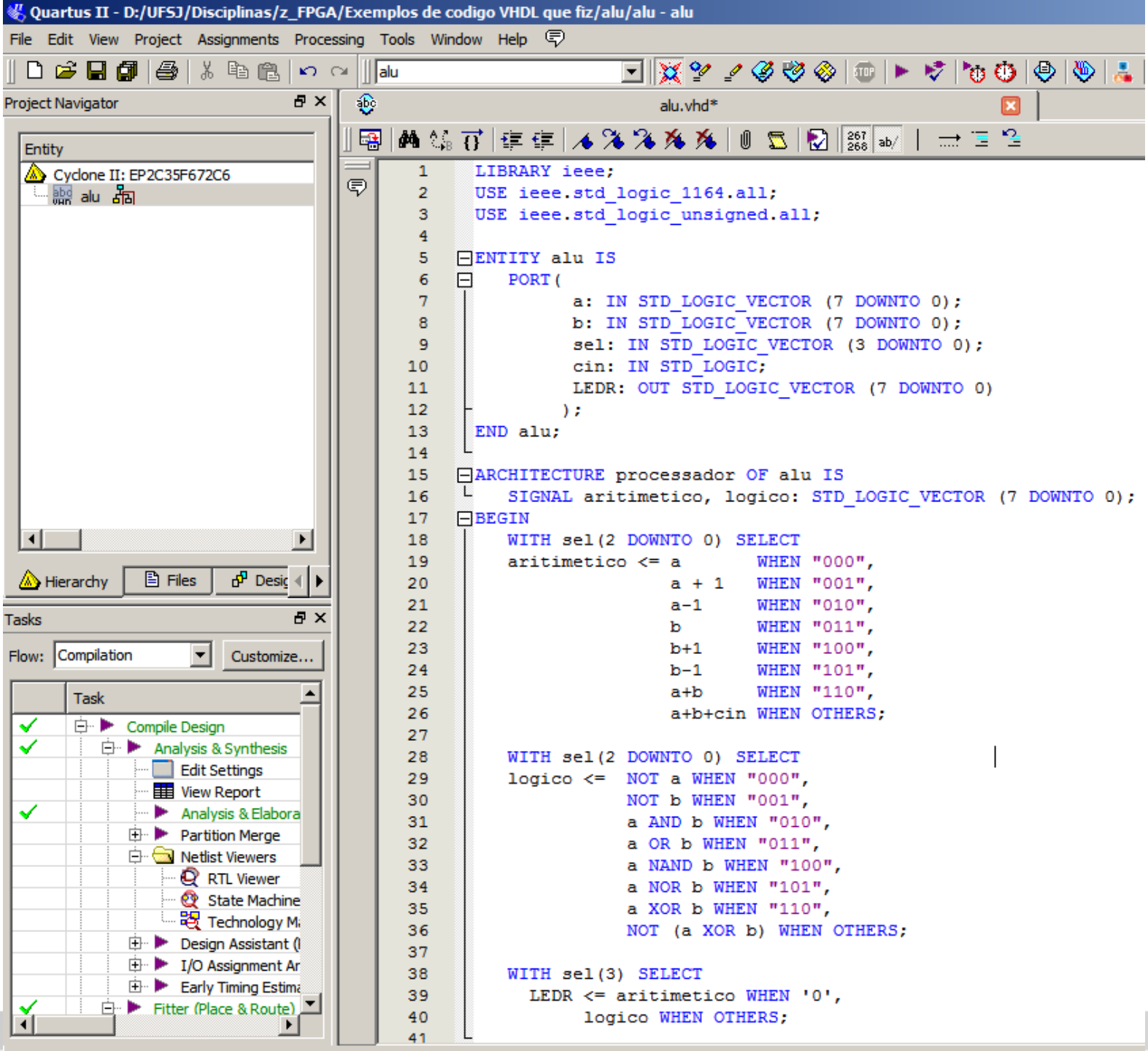
- Exemplo:




| sel | Operation | Function | Unit |
|---|---|---|---|
| 0000 | y <= a | Transfer a | Arithmetic |
| 0001 | y <= a+1 | Increment a | |
| 0010 | y <= a-1 | Decrement a | |
| 0011 | y <= b | Transfer b | |
| 0100 | y <= b+1 | Increment b | |
| 0101 | y <= b-1 | Decrement b | |
| 0110 | y <= a+b | Add a and b | |
| 0111 | y <= a+b+cin | Add a and b with carry | |
| 1000 | y <= NOT a | Complement a | Logic |
| 1001 | y <= NOT b | Complement b | |
| 1010 | y <= a AND b | AND | |
| 1011 | y <= a OR b | OR | |
| 1100 | y <= a NAND b | NAND | |
| 1101 | y <= a NOR b | NOR | |
| 1110 | y <= a XOR b | XOR | |
| 1111 | y <= a XNOR b | XNOR | |

```
ENTITY alu IS
        PORT(
            a: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
                                        b: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
                                        sel: IN STD_LOGIC_VECTOR (3 DOWNTO 0);
                                        cin: IN STD_LOGIC;
                                        LEDR: OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
            );  END alu;
ARCHITECTURE processador OF alu IS
        SIGNAL aritimetico, logico: STD_LOGIC_VECTOR (7 DOWNTO 0);
BEGIN
  WITH sel(2 DOWNTO 0) SELECT
        aritimetico <= a       WHEN "000",
                       a + 1   WHEN "001",
                       a-1     WHEN "010",
                        b      WHEN "011",
                       b+1     WHEN "100",
                       b-1     WHEN "101",
                      a+b      WHEN "110",
                    a+b+cin WHEN OTHERS;
  WITH sel(2 DOWNTO 0) SELECT
        logico <=  NOT a WHEN "000",
                     NOT b WHEN "001",
                   a AND b WHEN "010",
                     a OR b WHEN "011",
                a NAND b WHEN "100",
                   a NOR b WHEN "101",
                   a XOR b WHEN "110",
           NOT (a XOR b) WHEN OTHERS;

  WITH sel(3) SELECT
    LEDR <= aritimetico WHEN '0',
        logico WHEN OTHERS;
END processador;
```


Quartus II - D:/UFSJ/Disciplinas/z_FPGA/Exemplos de codigo VHDL que fiz/alu/alu - alu
File Edit View Project Assignments Processing Tools Window Help
alu
Project Navigator
Entity
Cyclone II: EP2C35F672C6
alu
Hierarchy
Files
Tasks
Flow: Compilation
Customize...
Task
Compile Design
Analysis & Synthesis
Edit Settings
View Report
Analysis & Elabora
Partition Merge
Netlist Viewers
RTL Viewer
State Machine
Technology M
Design Assistant (
I/O Assignment Ar
Early Timing Estima
Fitter (Place & Route)
alu.vhd*
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3 USE ieee.std_logic_unsigned.all;
4
5 ENTITY alu IS
6 PORT(
7 a: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
8 b: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
9 sel: IN STD_LOGIC_VECTOR (3 DOWNTO 0);
10 cin: IN STD_LOGIC;
11 LEDR: OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
12 );
13 END alu;
14
15 ARCHITECTURE processador OF alu IS
16 SIGNAL aritimetico, logico: STD_LOGIC_VECTOR (7 DOWNTO 0);
17 BEGIN
18 WITH sel(2 DOWNTO 0) SELECT
19 aritimetico <= a WHEN "000",
20 a + 1 WHEN "001",
21 a-1 WHEN "010",
22 b WHEN "011",
23 b+1 WHEN "100",
24 b-1 WHEN "101",
25 a+b WHEN "110",
26 a+b+cin WHEN OTHERS;
27
28 WITH sel(2 DOWNTO 0) SELECT
29 logico <= NOT a WHEN "000",
30 NOT b WHEN "001",
31 a AND b WHEN "010",
32 a OR b WHEN "011",
33 a NAND b WHEN "100",
34 a NOR b WHEN "101",
35 a XOR b WHEN "110",
36 NOT (a XOR b) WHEN OTHERS;
37
38 WITH sel(3) SELECT
39 LEDR <= aritimetico WHEN '0',
40 logico WHEN OTHERS;
41

Quartus II - D:/UFSJ/Disciplinas/z_FPGA/Exemplos de codigo VHDL que fiz/alu/alu - alu

File Edit View Project Assignments Processing Tools Window Help

alu

Project Navigator

Entity

Cyclone II: EP2C35F6

alu

Hierarchy

Tasks

Flow: C Customize...

Task

Compile

Ana

Fitt

alu.vhd*

```
1   LIBRARY ieee;
2   USE ieee.std_lo
3   USE ieee.std_lo
4
5   ENTITY alu IS
6      PORT(
7            a: IN
8            b: IN
9            sel: I
10           cin: I
11           LEDR: (
12        );
13  END alu;
14
15  ARCHITECTURE pr
16     SIGNAL aritir
17  BEGIN
18     WITH sel(2 D
19     aritimetico
20
21
22
23
24
25
26
27
28     WITH sel(2 D
29     logico <=  N
30                N
31                a
32                a
33                a
34                a
35                a
36                N
37
38     WITH sel(3)
39       LEDR <= ar
40            logic
41
```

RTL Viewer - D:/UFSJ/Disciplinas/z_FPGA/Exemplos de codigo VHDL que fiz/alu/alu - alu

File Edit View Tools Window Help

Page Title: alu Page: 1 of 2

Mux15 Mux14 Mux13 Mux12 Mux7

Type Message

System Processing Extra Info Info Warning Critical

Message: Location: Locate

0% 00:00:00

Iniciar Disco Local (D:) Quartus II - D:/UFSJ/Dis... RTL Viewer - D:/UFSJ/... Google - Google Chrome Cap1-introducao arquite... 15:00