# omas Embarcados - SE

MSc. Fabio H. Pimentel

## guagem VHDL

## te 1

# os de Sistemas Embarcados

## m VHDL

**Speed ASIC Description Language**

**m para descrição de hardware**

**truturada para a descrição de circuitos digitais**

**o eletrônico é descrito por sentenças**

**a ao circuito ser simulado e sintetizado, isto é,**

**nado em portas lógicas**

# os de Sistemas Embarcados

## m VHDL - Histórico

Speed ASIC Description Language

borada pelo Departamento de Defesa (EUA)
poca da crise do ciclo de vida dos projetos eletrônicos

S: linguagem básica concluída pelas empresas
Intermetrics, IBM e Texas Instruments

itos transmitidos ao IEEE

licação do padrão IEEE VHDL 87

visões foram realizadas ao longo desses anos.

EE STD 1076-2008

# os de Sistemas Embarcados

## m VHDL

h Speed ASIC Description Language

s

sinônimo de RAPIDEZ e PRODUTIVIDADE.

código pode ser usado em diversas tecnologias.

dade e longevidade para um projeto.

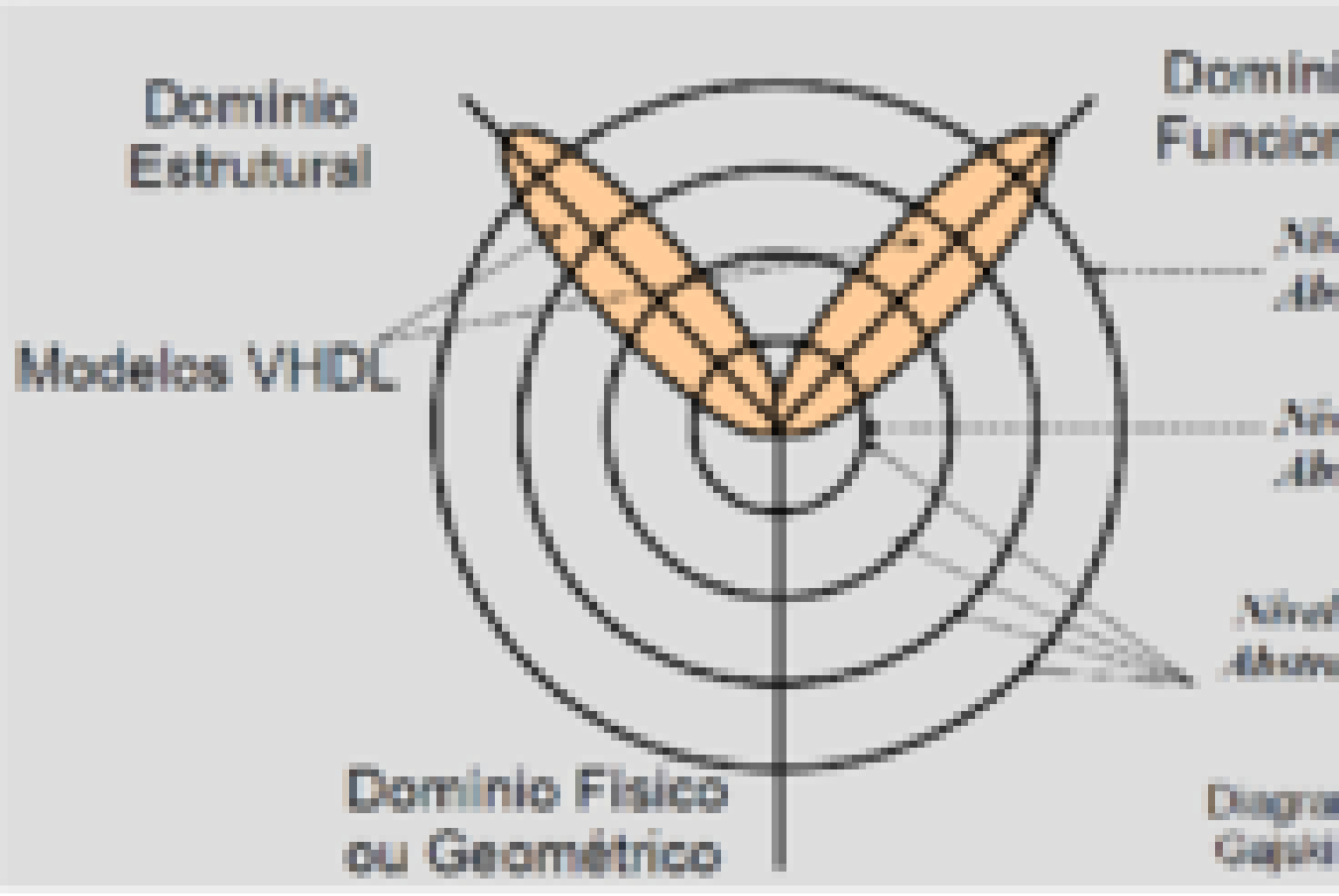
níveis de teste no código = confiabilidade nos resultados.

# os de Sistemas Embarcados

## m VHDL

**Speed ASIC Description Language**

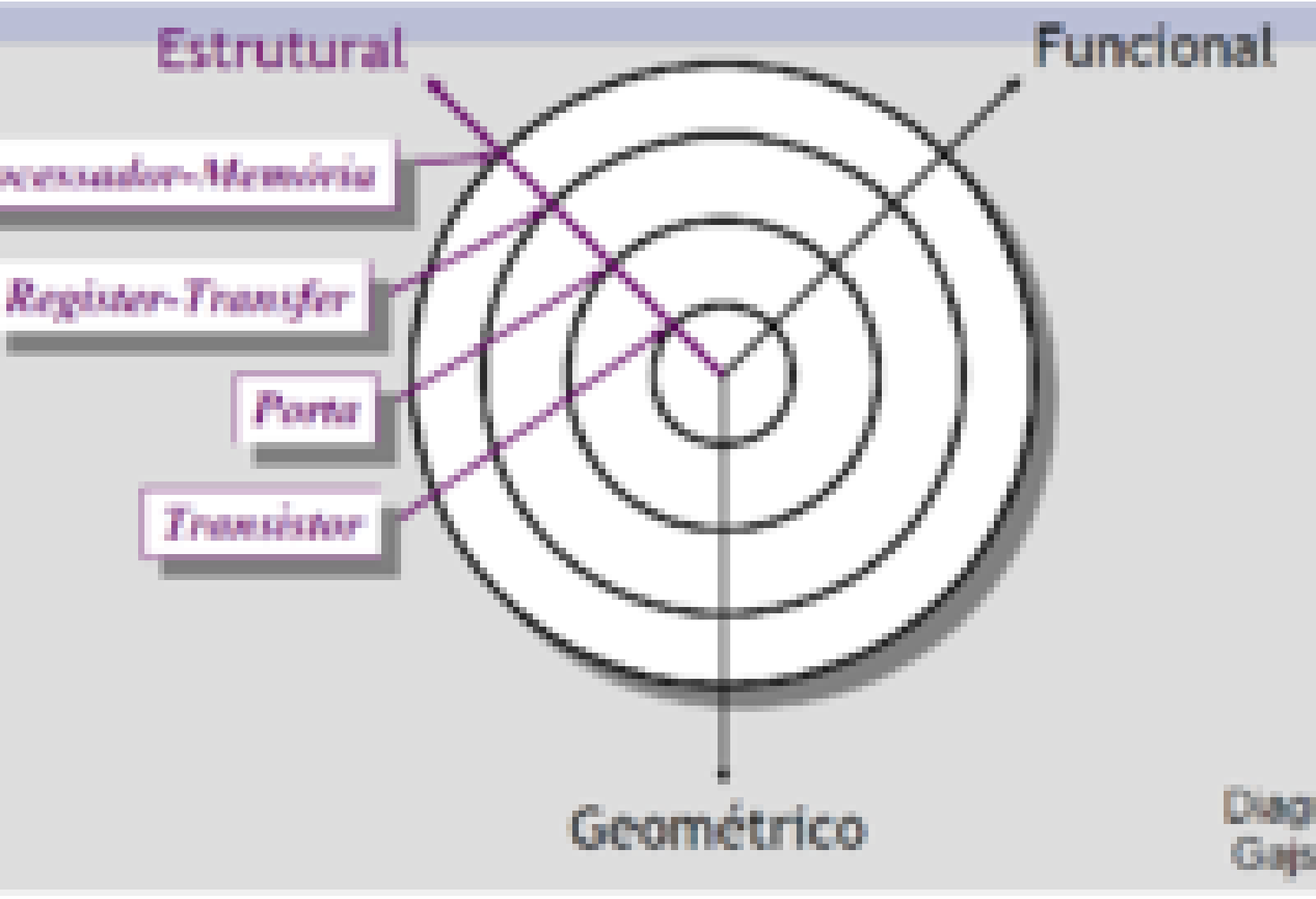
**escrição do circuito funcionalmente e estruturalmente.**



# os de Sistemas Embarcados

## m VHDL

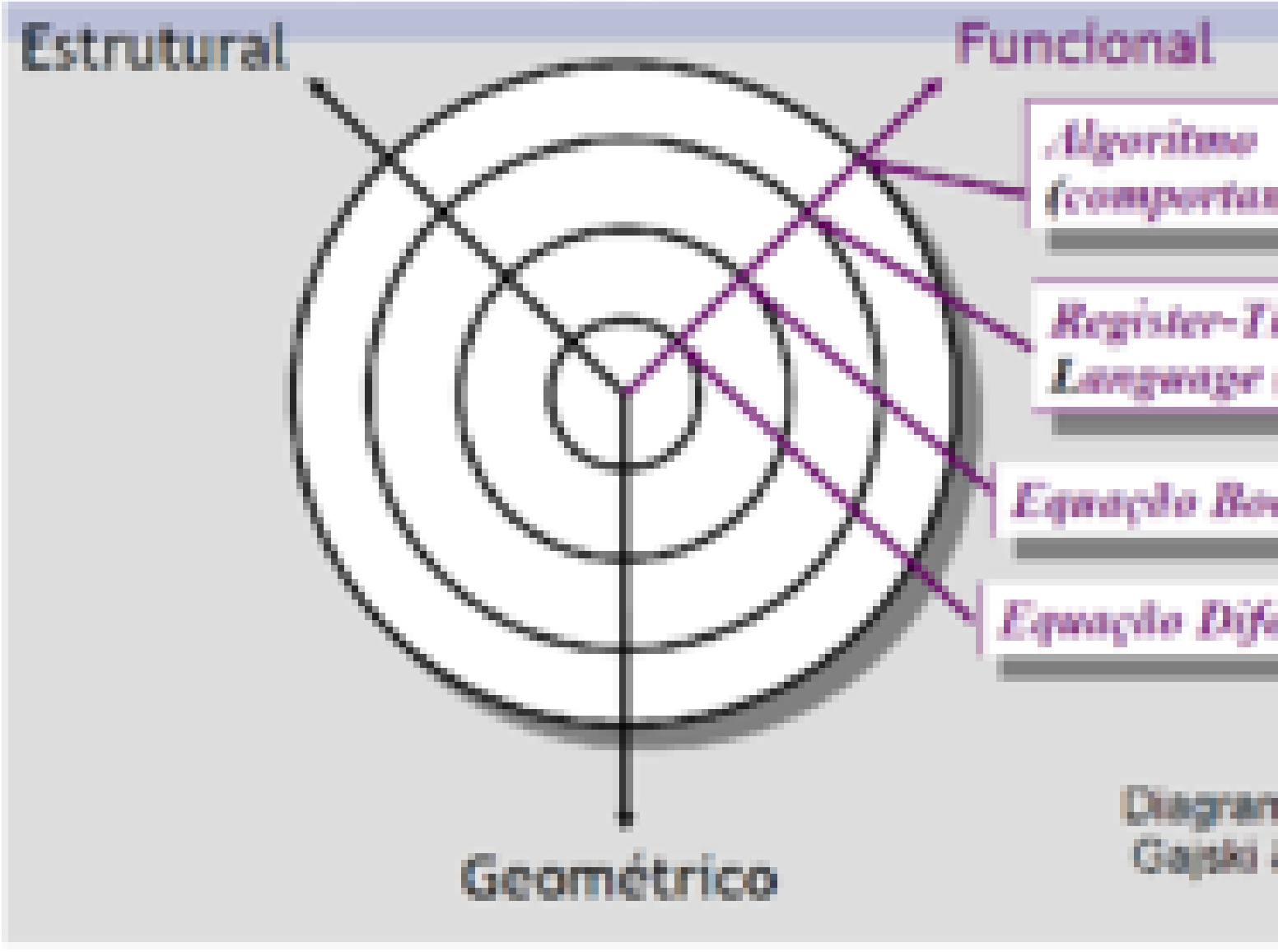
**Speed ASIC Description Language**



# os de Sistemas Embarcados

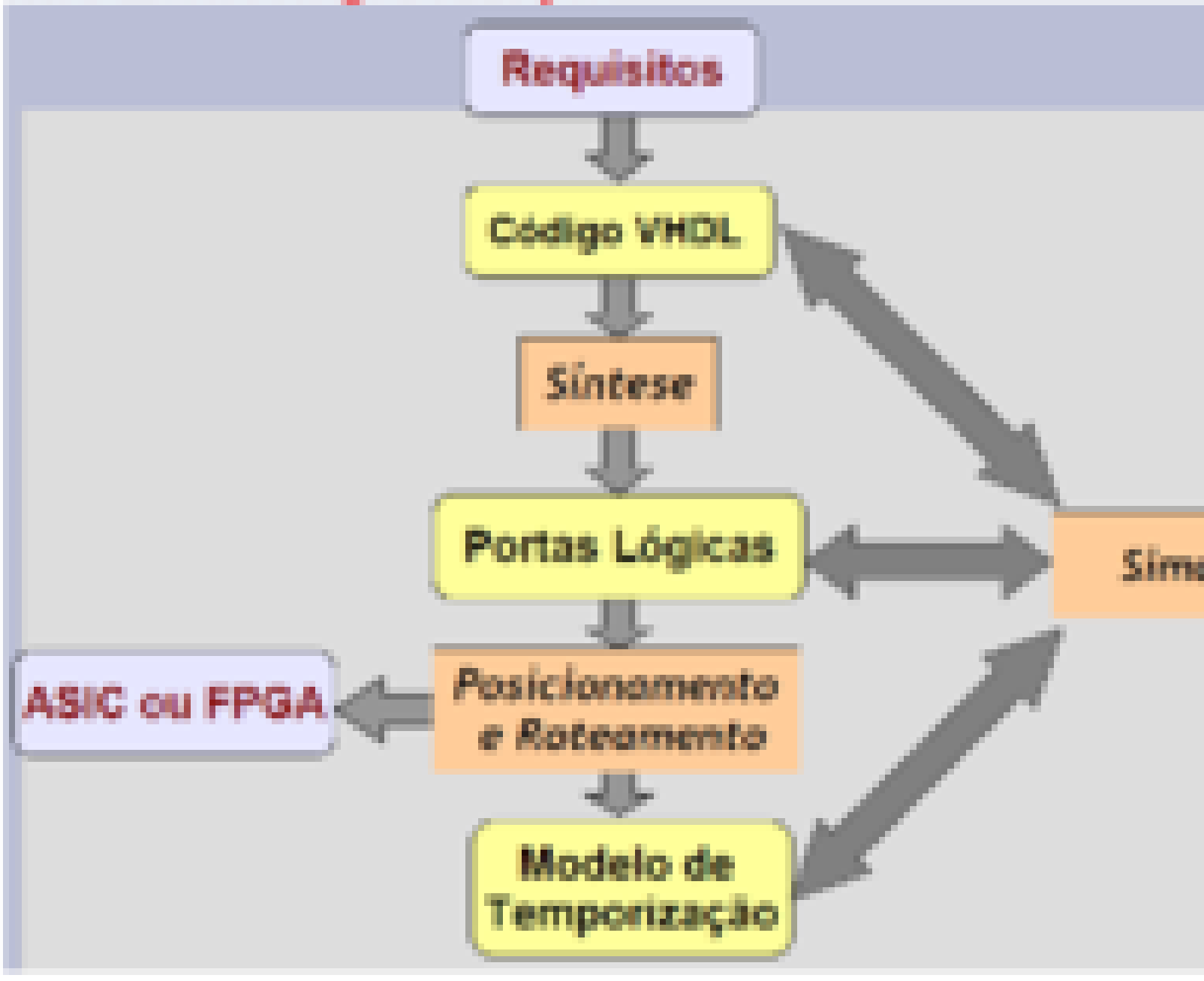
## m VHDL

Speed ASIC Description Language



# os de Sistemas Embarcados

## m VHDL - Metodologia de Projeto



# os de Sistemas Embarcados

## m VHDL – Exemplo Processo

**blema:**

**Projetar um meio somador de um bit com vai_um e habilita.**

**ecificações:**

- **Passa o resultado apenas se habilita for igual a '1'.**
- **Resultado é zero se habilita for igual a '**
- **Resultado recebe x + y**
- **Vai_um recebe o vai_um, se houver, de**

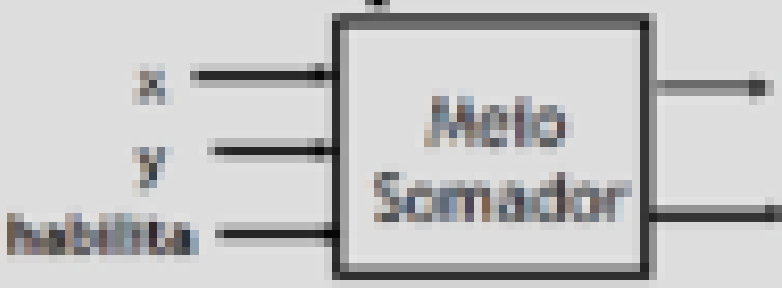
# os de Sistemas Embarcados

## m VHDL – Exemplo Projeto Comportamental

**iciando com um algoritmo, uma descriçã**
**alto nível do somador é criada:**

```
bilita = 1 THEN
   resultado = x XOR y
   vai_um = x AND y

   vai_um = 0
   resultado = 0
```

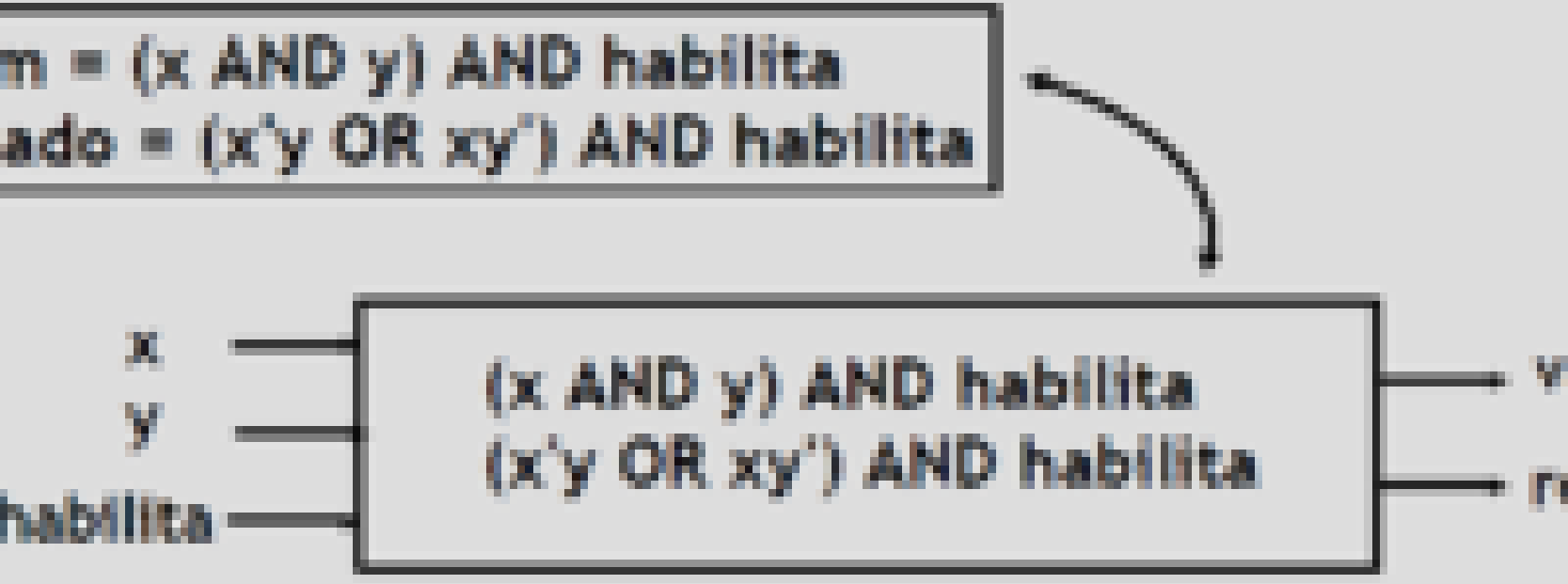



**O modelo pode ser agora simulado ness**
**nível de descrição para verificar o corre**
**entendimento do problema.**

# os de Sistemas Embarcados

## m VHDL – Exemplo Projeto Fluxo de Dados

**n a descrição de alto nível confirmada**
**ações lógicas descrevendo o fluxo de**
**os são então criadas.**

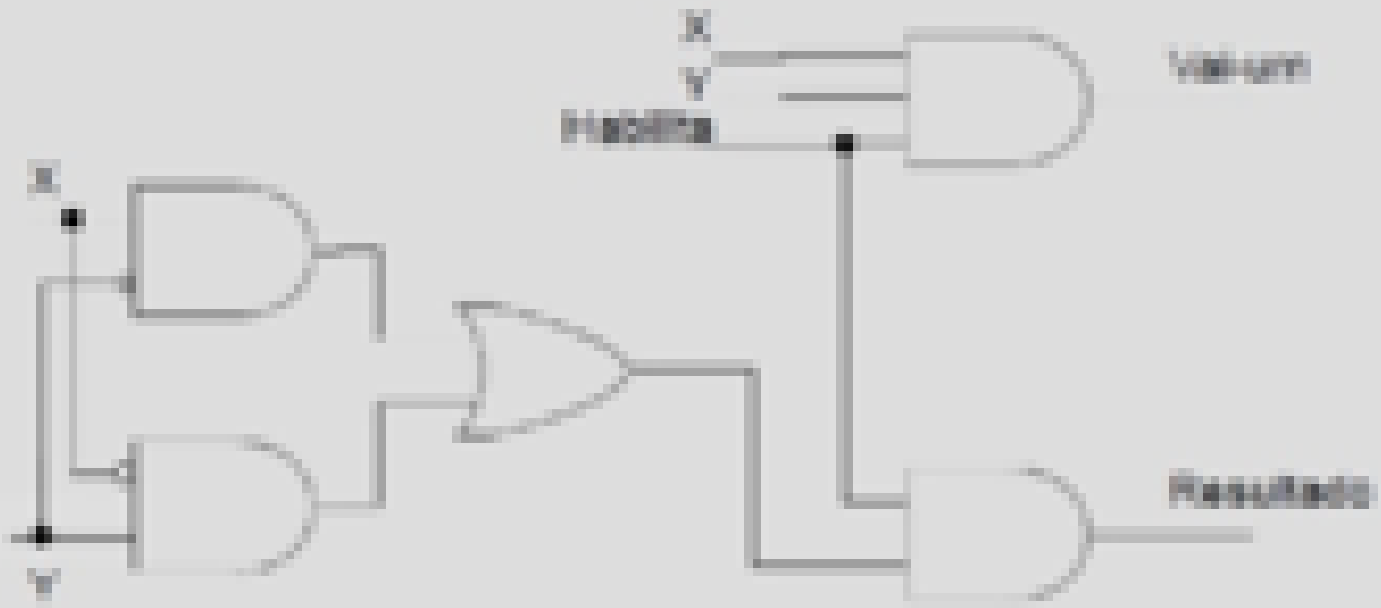



**vamente, o modelo pode ser simulado**
**ste nível para confirmar as equações lógi**

# os de Sistemas Embarcados

## m VHDL – Exemplo Projeto Lógico

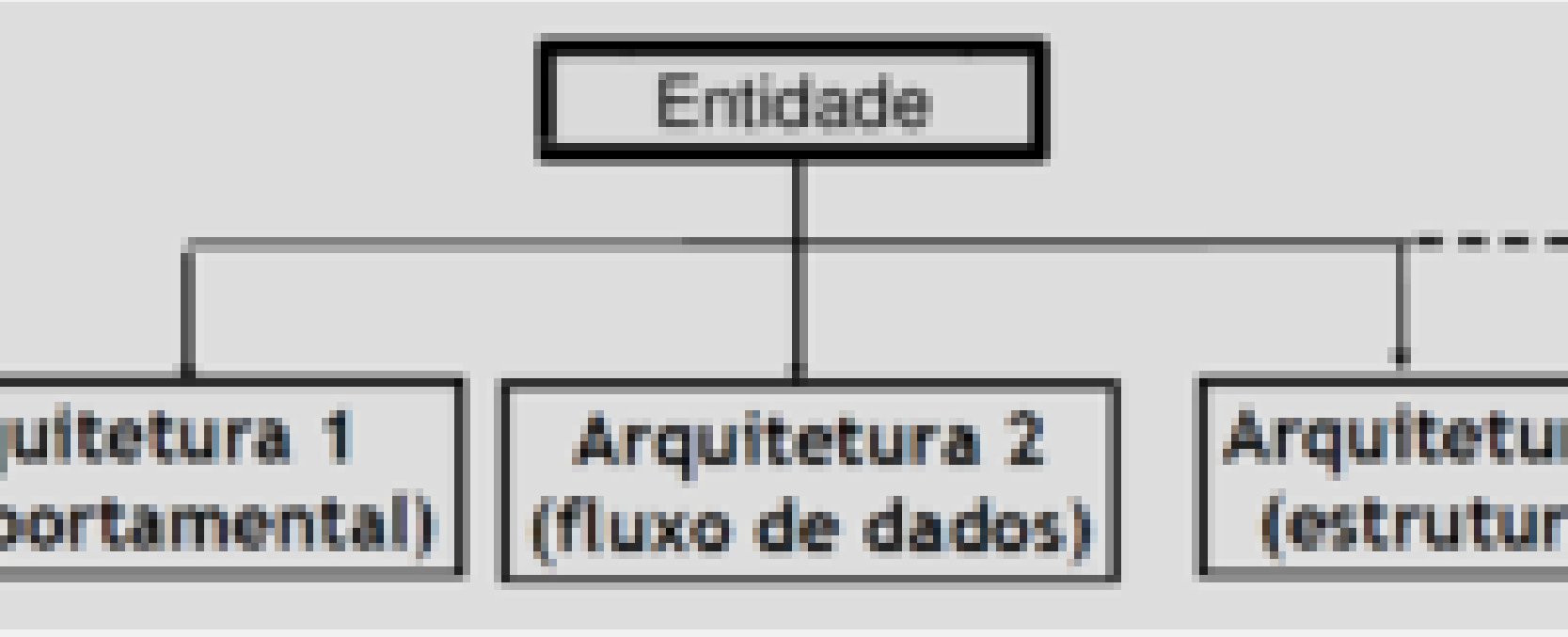
* Finalmente, uma descrição estrutu
é criada no nível de portas.




* Essas portas podem ser obtidas d
uma biblioteca de componentes.

# os de Sistemas Embarcados

## m VHDL – Processo de Projeto VHDL



# os de Sistemas Embarcados

## m VHDL – Declaração de Entidade

a declaração de entidade (ENTITY) descr
terface do componente.

a cláusula PORT indica as portas de entra
da.

a entidade pode ser pensada como um sím
a um componente.

# os de Sistemas Embarcados

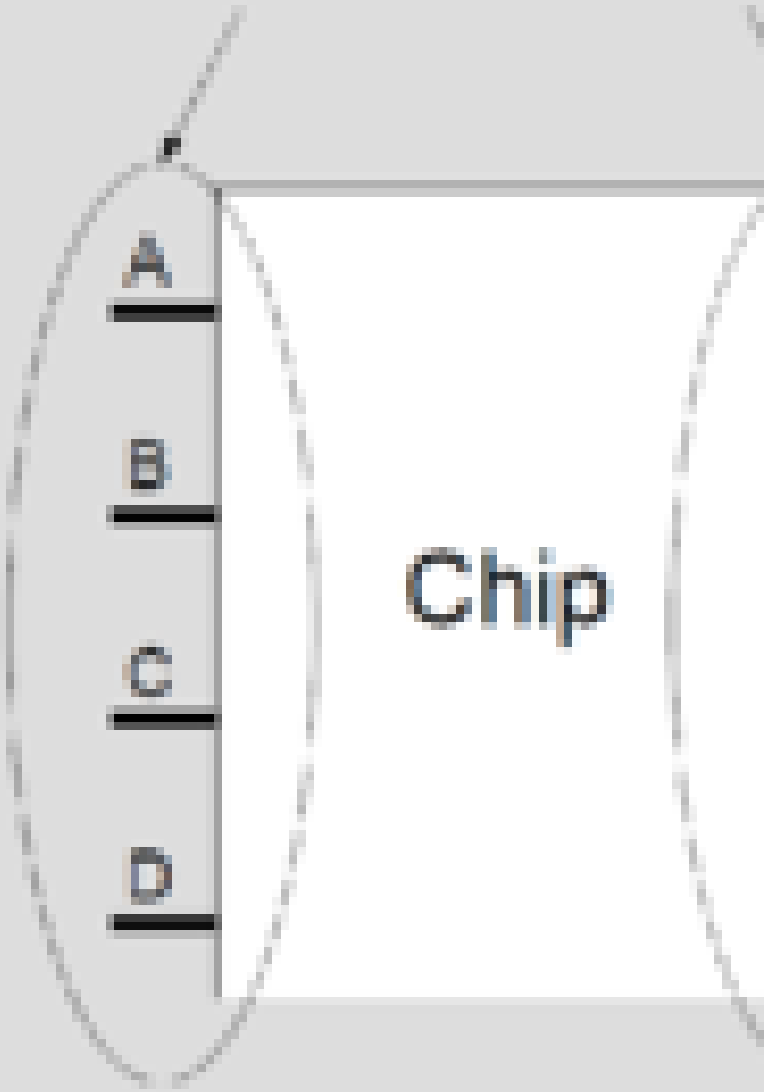
## m VHDL – Entidade

efine entradas e saídas

xemplo:

```
test is
  A,B,C,D: in std_logic;
  E: out std_logic);
est;
```

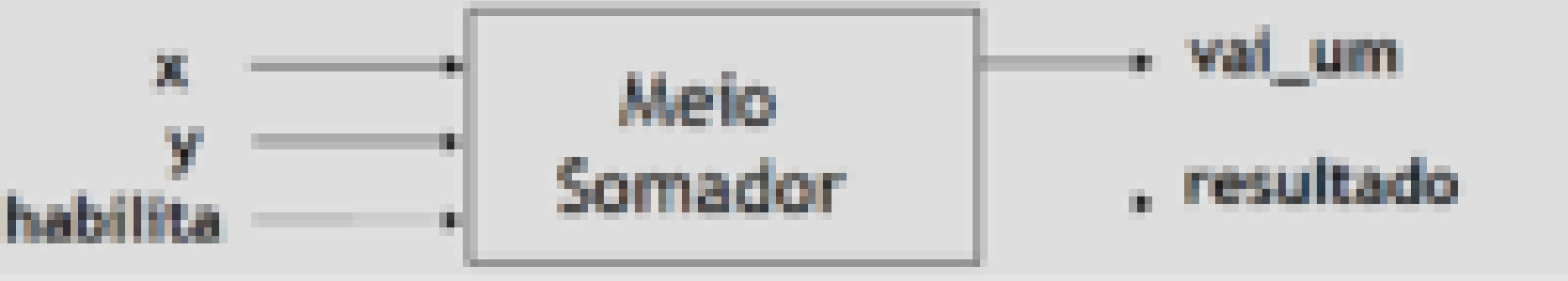
Entradas e Saídas



# os de Sistemas Embarcados

## m VHDL – Declaração de Entidade

```
ENTITY meio_somador IS
    PORT (x, y, habilita: IN bit;
          vai_um, resultado: OUT bit);
END meio_somador;
```



# os de Sistemas Embarcados

## m VHDL – Declaração de Porta

**a declaração de porta (PORT) estabelece**
**erface entre o componente e o mundo ex**

**três partes na declaração PORT**
- **Nome**
- **Modo**
- **Tipos de Dados**

```
NTITY test   IS
        PORT (<nome> : <modo>  <tipos_dados
ND test;
```

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta - Nome

nas letras, dígitos e sublinhados podem s
dos;
rimeiro caractere deve ser uma letra;
timo caractere não pode ser um sublinha
são permitidos dois sublinhados consecu

| Nomes Legais | Nomes Ilegais |
|---|---|
| rs_clk | _rs_clk |
| ab08B | sinal#1 |
| A_1023 | A__1023 |
| | rs_clk_ |

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta - Nome

ão é sensível à "Caixa Alta ou Baixa"
nputa, INPUTA e InputA se referem à mesma
variável.
omentários
--' marca um comentário até o final da linha
Se você deseja comentar múltiplas linhas, um
precisa ser colocado no início de cada linha.
s sentenças são terminadas por ';'
ribuição de valores aos sinais: '<='
ribuição de valores às variáveis: ':='

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta - Modo

odo da porta de interface descreve o sentid
de dados tomando como referência o
ponente.
inco tipos de fluxo de dados são:
os dados entram nesta porta e podem ape
lidos (é o padrão).
T: os dados saem por essa porta e podem
erem escritos.
FFER: similar a Out, mas permite realiment
erna.
OUT: o fluxo de dados pode ser em qualqu
tido, com qualquer número de fontes perm
rramento)
KAGE: o sentido do fluxo de dados é decon

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Tipos de Dados

pos de dados que passam através de uma
n ser especificados para completar a interf

ados podem ser de diferentes tipos, depen
acote e bibliotecas utilizados.

ns tipos de dados definidos no padrão IEE

, Bit_vector
olean
teger
d_ulogic, std_logic

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Tipos de Dados

- **bit values: '0', '1'**
- **boolean** values: TRUE, FALSE
- **integer** values: -(231) to +(231 – 1)
- **std_logic** values: 'U','X','1','0','Z','W','H','L','-'
  - 'U' = uninitialized
  - 'X' = unknown
  - 'W' = weak 'X'
  - 'Z' = floating
  - 'H'/'L' = weak '1'/'0'
  - '-' = don't care

- **std_logic_vector** (n downto 0);
- **std_logic_vector** (0 upto n);

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Tipos de Dados - Arquitetura

clarações do tipo Architecture descrever
eração do componente.

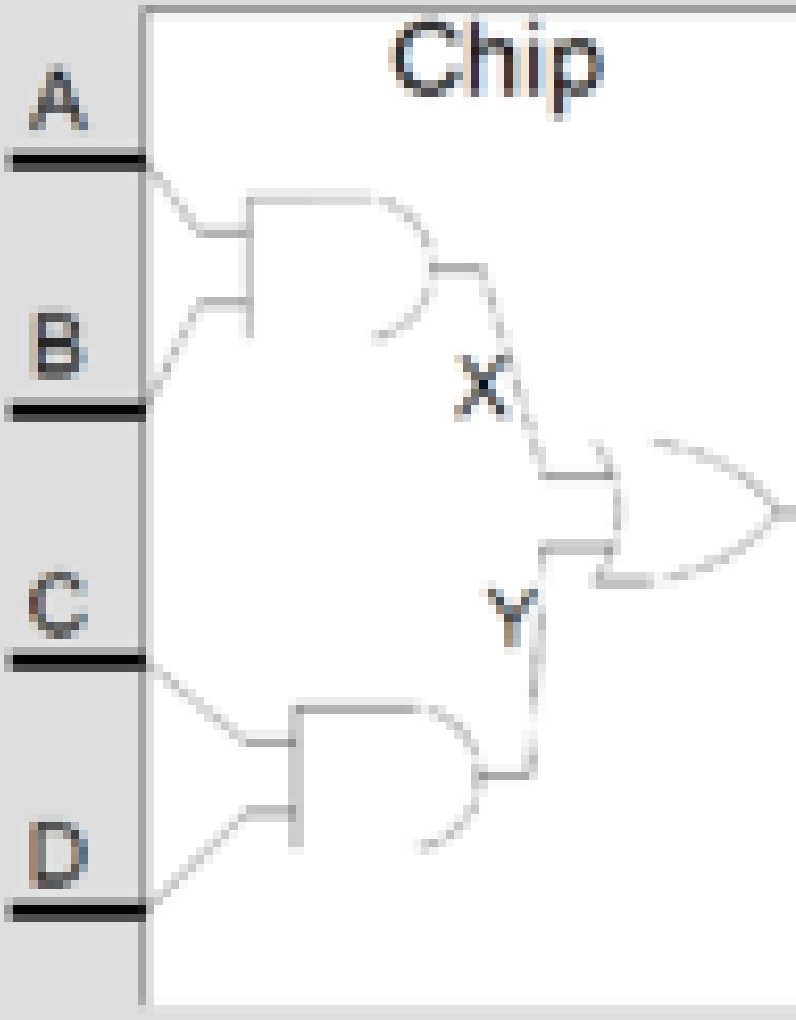
litas arquiteturas podem existir para uma
sma entidade, mas apenas pode haver u
las ativa por vez.

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Tipos de Dados - Arquitetura

**fine a funcionalidade**
**circuito**

= A AND B;
= C AND D;
= X OR Y;



# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Tipos de Dados – Arquitetura

```
ARCHITECTURE behavior1 OF meio_somador IS
BEGIN
        PROCESS (habilita, x, y)
        BEGIN
            IF (habilita = '1') THEN
                resultado <= x XOR y;
                vai_um <= x AND y;
            ELSE
                vai_um <= '0';
                resultado <= '0';
            END IF;
        END PROCESS;
END behavior1;
```

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Tipos de Dados – Arquitetura

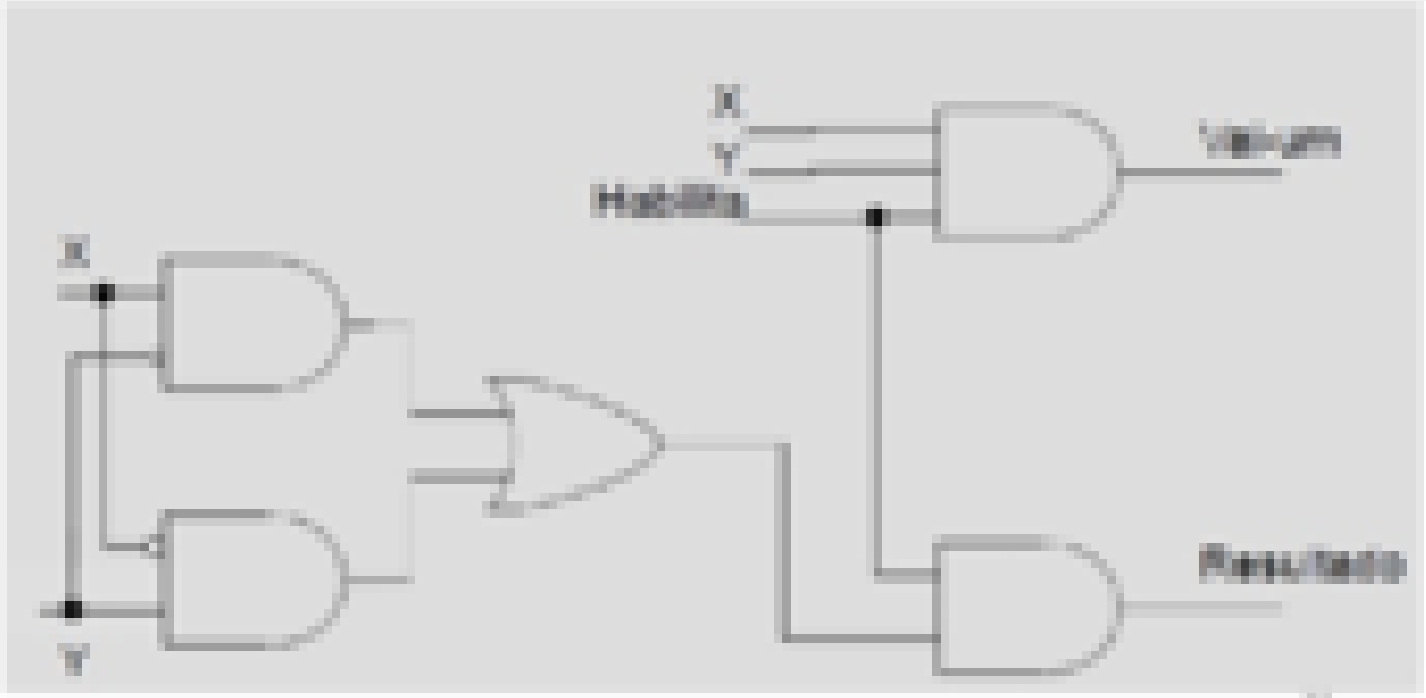
```
ARCHITECTURE data_flow OF meio_somador I
BEGIN
        vai_um <= (x AND y) AND habilita;
        resultado <= (x XOR y) AND habilita;
END data_flow;
```

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Tipos de Dados – Arquitetura

**fazer a arquitetura estrutural, nós preci**
**eiro definir as portas a serem utilizadas.**

**xemplo a seguir, nós precisamos definir**
**as NOT, AND, e OR.**



# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Tipos de Dados – Arquitetura

```
ENTITY not_1 IS
        PORT (a: IN bit; output: OUT bit);
END not_1;

ARCHITECTURE data_flow OF not_1 IS
BEGIN
        output <= NOT(a);
END data_flow;
```

```
ENTITY and_2 IS
        PORT (a,b: IN bit; output: OUT bit);
END and_2;

ARCHITECTURE data_flow OF and_2 IS
BEGIN
        output <= a AND b;
END data_flow;
```

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Tipos de Dados – Arquitetura

```
ENTITY or_2 IS
        PORT (a,b: IN bit; output: OUT bit);
END or_2;

ARCHITECTURE data_flow OF or_2 IS
BEGIN
        output <= a OR b;
END data_flow;
```

```
ENTITY and_3 IS
        PORT (a,b,c: IN bit; output: OUT bit);
END and_3;

ARCHITECTURE data_flow OF  and_3 IS
BEGIN
        output <= a AND b AND c;
END data_flow;
```

# os de Sistemas Embarcados

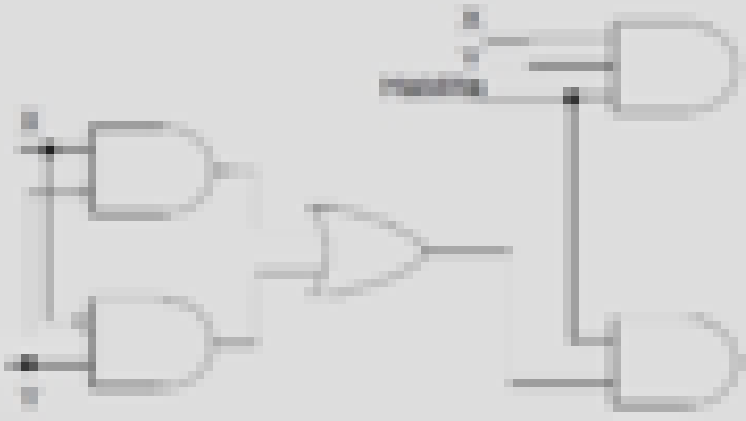
## m VHDL – Declaração de Porta – Tipos de Dados – Arquitetura

```
TECTURE structural OF meio_somador IS
PONENT and2 PORT(a,b: IN bit; output: OUT bit); END COMPONENT;
PONENT and3 PORT(a,b,c: IN bit; output: OUT bit); END COMPONEN
PONENT or2 PORT(a,b: IN bit; output: OUT bit); END COMPONENT;
PONENT not1 PORT(a: IN bit; output: OUT bit); END COMPONENT;

ALL: and2 USE ENTITY work.and_2  (data_flow);
ALL: and3 USE ENTITY work.and_3  (data_flow);
ALL: or2 USE ENTITY work.or_2    (data_flow);
ALL: not1 USE ENTITY work.not _1 (data_flow);

AL v,w,z,nx ,ny: BIT;


ot1 PORT MAP (x,nx);
ot1 PORT MAP (y,ny);
nd2 PORT MAP (nx,y,v);
nd2 PORT MAP (x,ny,w);
r2 PORT MAP (v,w,z);
nd2 PORT MAP (habilita,z ,resultado);
nd3 PORT MAP (x,y,habilita,vai_um);
ructural;
```

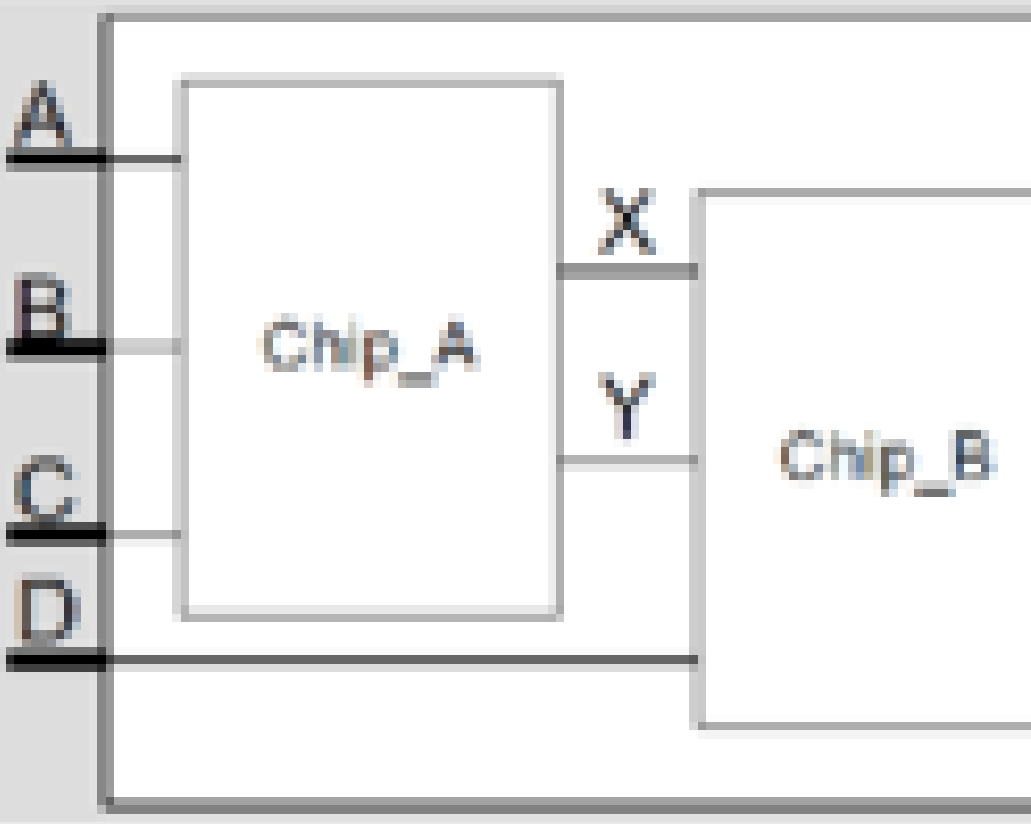


# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Port Map

```
Chip_A
p (A,B,C,X,Y);

Chip_B
p (X,Y,D,E);
```



# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Port Map Exemplo

```
RY IEEE;
EEE.STD_LOGIC_1164.ALL;

TY TEST IS
 (A,B,C,D  :  IN STD_LOGIC;
      E    : OUT STD_LOGIC);
TEST;

ITECTURE BEHAVIOR OF TEST IS

AL X,Y : STD_LOGIC;

PONENT Chip_A
 (L,M,N  :  IN STD_LOGIC;
      O,P  : OUT STD_LOGIC);
COMPONENT;
```

```
COMPONENT Chip_B
PORT (Q,R,S :  IN STD_L
           T  : OUT STD_L
END COMPONENT;

BEGIN

Chip1 : Chip_A
PORT MAP (A,B,C,X,Y);

Chip2 : Chip_B
PORT MAP (X,Y,D,E);

END BEHAVIOR;
```

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Port Map Exemplo AND

```
entity and2 is
    port ( a, b : in bit;  y : out bit );
end entity and2;

architecture basic of and2 is
begin
    and2_behavior : process is
    begin
        y <= a and b after 2 ns;
        wait on a, b;
    end process and2_behavior;
end architecture basic;
```

# os de Sistemas Embarcados

## m VHDL – Declaração de Porta – Port Map Exemplo Somador d

```
1   ENTITY add1 IS
2   PORT(
3           cin     :IN BIT;
4           a       :IN BIT;
5           b       :IN BIT;
6           s       :OUT BIT;
7           cout    :OUT BIT);
8   END add1;
9
10  ARCHITECTURE a OF add1 IS
11  BEGIN
12
13        s       <= a XOR b XOR cin;
14        cout    <= (a AND b) OR (a AND cin) OR (b AND cin);
15  END a;
```

Continua...