Fundamentos de Sistemas Digitais

Introdução à Linguagem VHDL

prof. Dr. Alexandre M. Amory

Referências

- Sugiro estudarem nesta ordem de preferência:
- Vahid, Cap 9 – Ignorar verilog e SystemC
  - Bem didático! um pouco incompleto !
- Free Range VHDL
  - Até sec 4.3, mais completo
- D'AMORE, Roberto. VHDL: Descrição e Síntese de Circuitos Digitais. Rio de Janeiro: LTC, 2005. 259 p.
- PEDRONI, Volnei A. Eletrônica Digital Moderna e VHDL. Elsevier Ltda. Editora, Rio de Janeiro, RJ: 2010. 619 p.
  - Cap 19
- Floyd. A Parte de VHDL está espalhada no livro, o que pode ser um pouco confuso.

ATENÇÃO

- Não se aprende alguma linguagem :
  - Escutando alguém explicar
  - Somente lendo livros

- Aprende praticando
  - Exercício EXTRA-CLASSE é FUNDAMENTAL !
  - Façam exercícios sugeridos e procurem outros nos livros indicados

Foco deste PPT

- Ênfase na representação de expressões Booleanas e Portas lógicas em VHDL

Níveis de abstração típicos:

1o : máscara de layout (baixa abstração)

2o: diagrama de transistores

3o: portas lógicas

4o: blocos / funções lógicos

5o: nível de registradores

Qual é o problema do esquemático ?!?!?

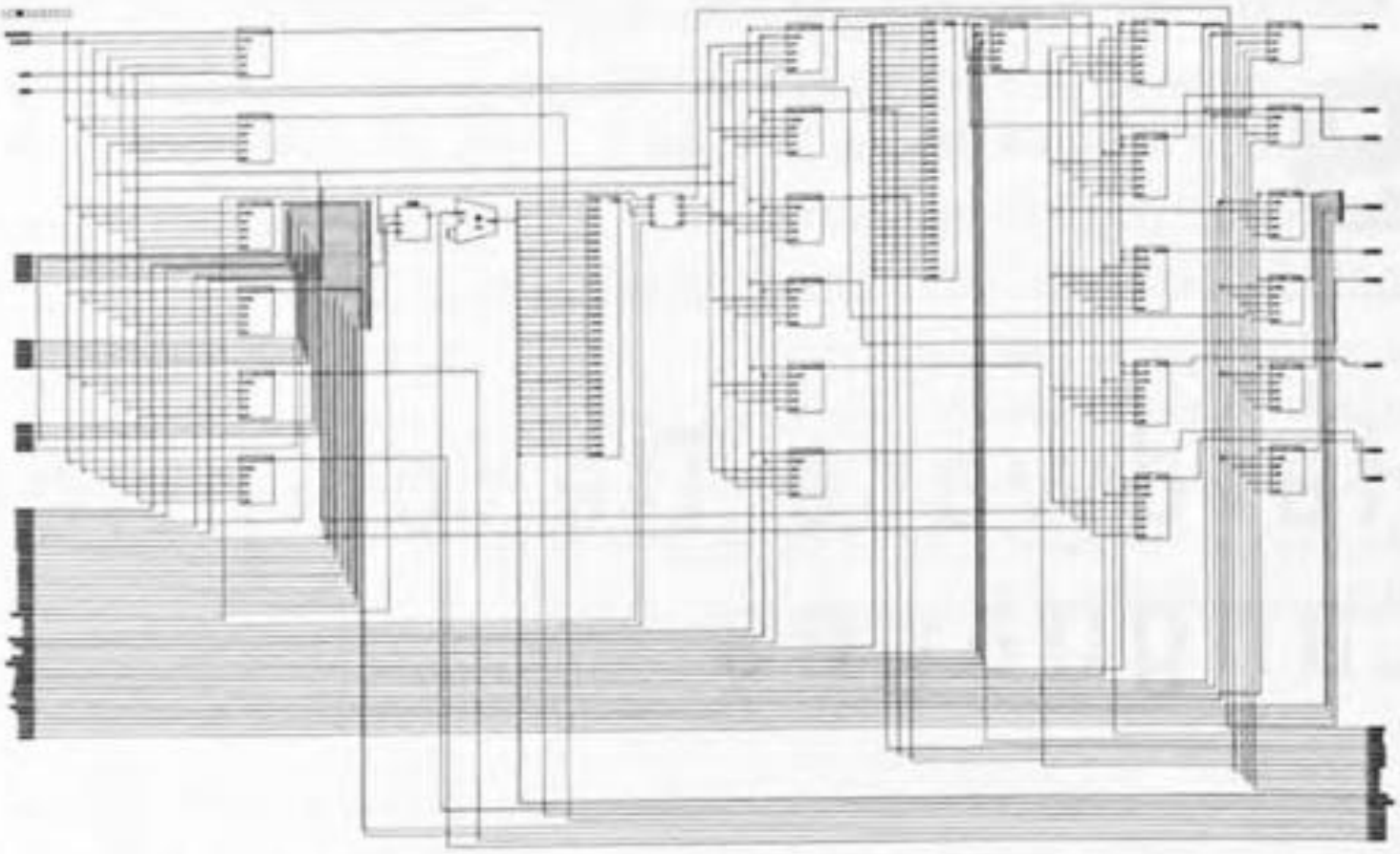
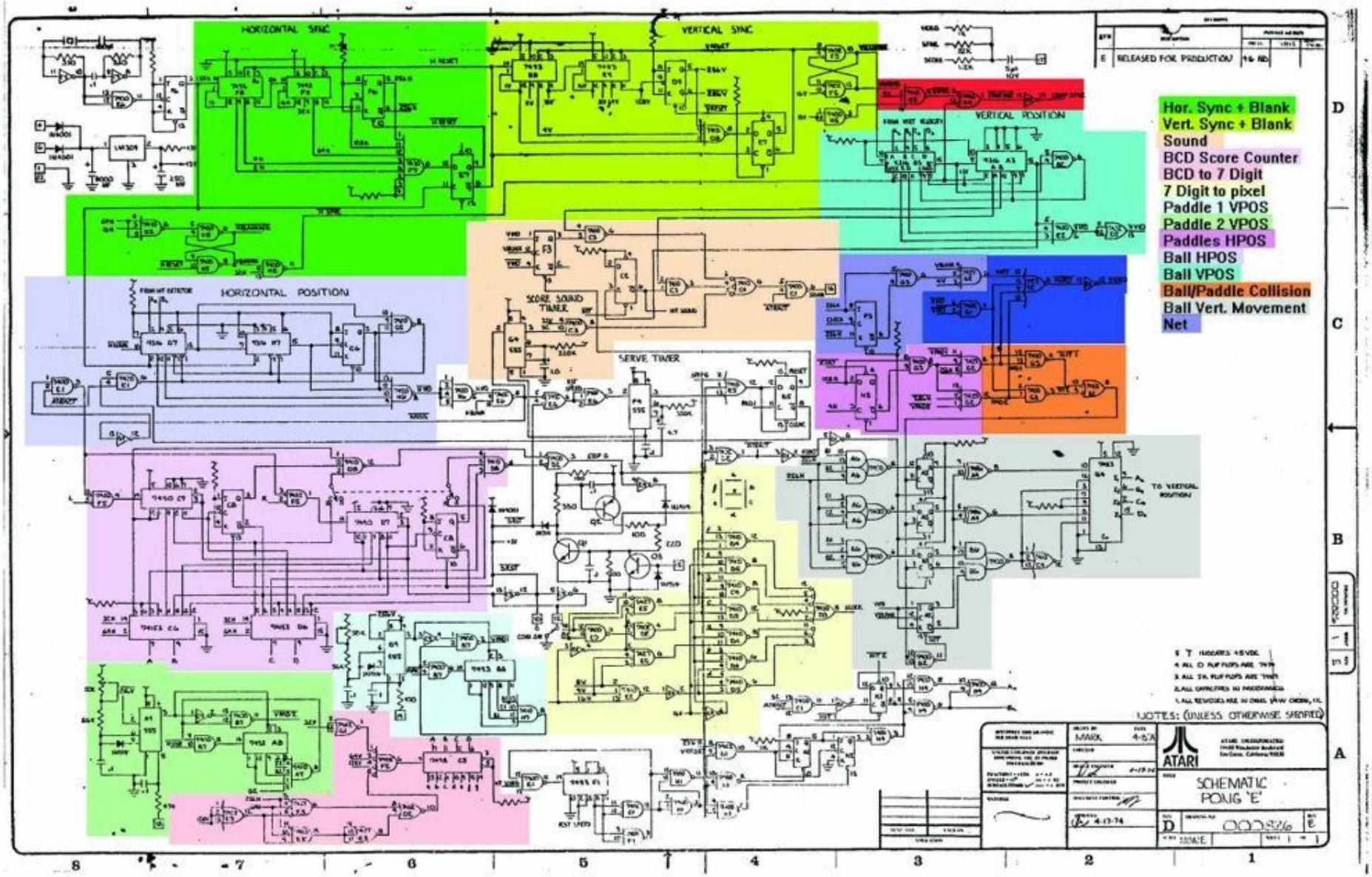


Figure 9.2 Schematics become hard to read beyond a dozen or so components—the graphical information becomes a nuisance rather than an aid.

* Vahid

Lembram do PONG da Atari ?



VHDL – Uma Linguagem de Descrição de Hardware

- VHDL ➔ *VHSIC Hardware Description Language*
- Padrão IEEE em 1987 (Institute of Electrical and Electronics Engineers), revisado em 1993
- Linguagem utilizada mundialmente por empresas de CAD para especificação, simulação, síntese, propriedade intelectual

- Outras linguagens de descrição de hardware
  - SystemC, VERILOG, Handel-C, SDL, ISP, Esterel, …

Benefícios / Desvantagens

- Benefícios
  - Projetos independentes da tecnologia (implementação física é postergada)
  - Ferramentas de CAD compatíveis entre si
  - Flexibilidade: re-utilização, escolha de ferramentas e fornecedores
  - Facilidade de atualização dos projetos
  - Exploração em alto nível de abstração permite analisar diferentes alternativas de implementação
  - Verificação do comportamento do sistema digital através de simulação
  - Redução do tempo de projeto → reduz "*time-to-market*" → redução de custo
  - Linguagem fortemente tipada → elimina erros de baixo nível

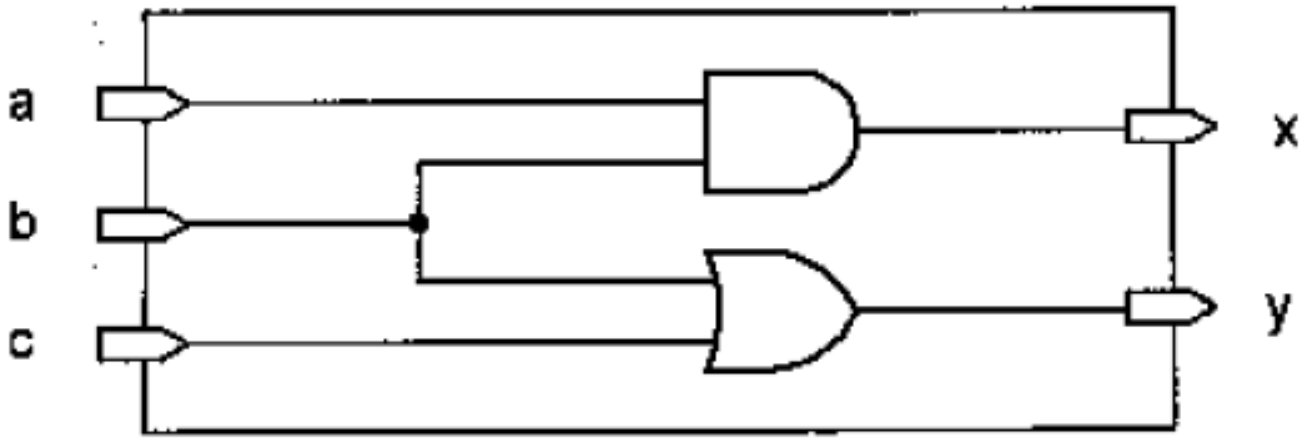
- Desvantagens
  - Hardware gerado é menos otimizado
  - Controlabilidade/Observabilidade de projeto reduzidas
  - Falta de pessoal treinado para lidar com a linguagem

Níveis de Abstração

- Permite descrever hardware em diversos níveis de abstração
  - Algorítmico, ou Comportamental
  - Transferência entre registradores (RTL)
  - Nível lógico com atrasos unitários ou sem atrasos
  - Nível lógico com atrasos arbitrários

Estrutura de um programa VHDL

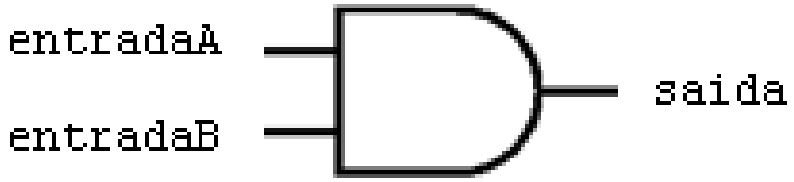
Entidade e Arquitetura



Exemplo de Descrição VHDL: Biblioteca

- Descrição em VHDL de uma porta lógica E

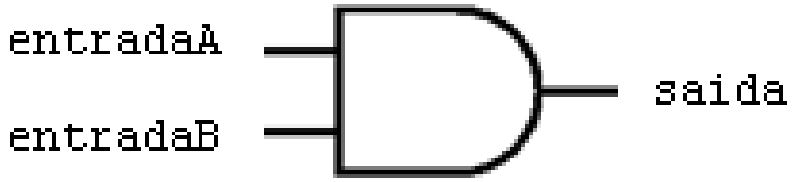
```
library ieee;
use ieee.std_logic_1164.all;
```



Exemplo de Descrição VHDL: Entidade

- Descrição em VHDL de uma porta lógica E

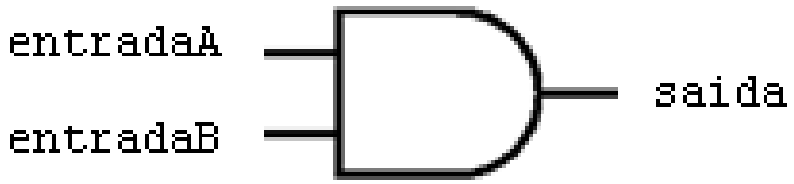
```
library ieee;
use ieee.std_logic_1164.all;

entity PortaE is
    port
    (
        entradaA: in std_logic;
        entradaB: in std_logic;
        saida: out std_logic
    );
end PortaE;
```



Exemplo de Descrição VHDL: Arquitetura

- Descrição em VHDL de uma porta lógica E

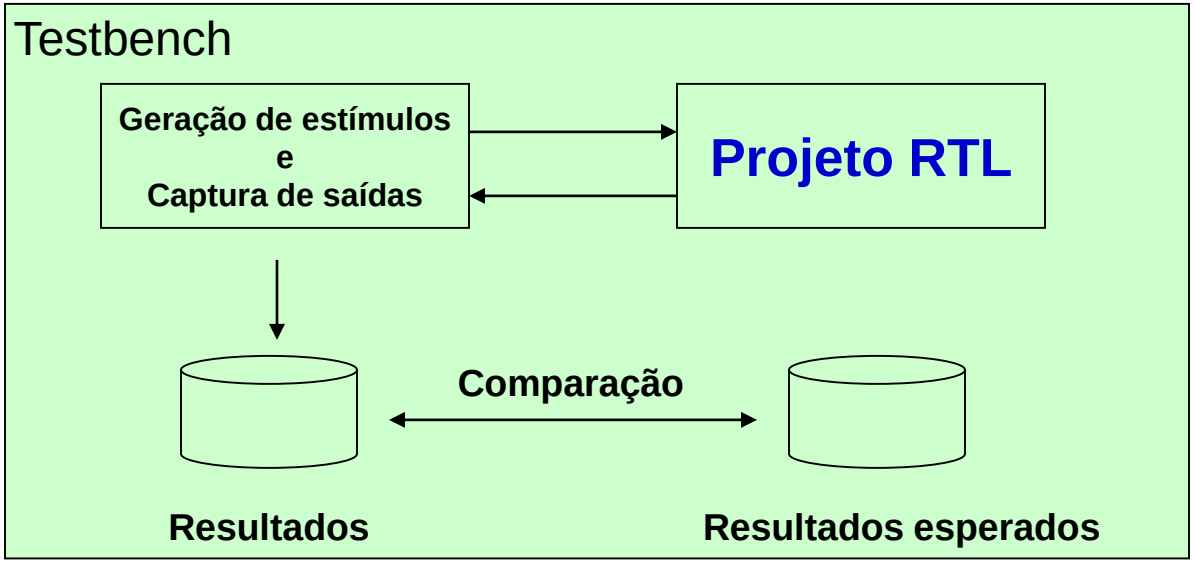
```
library ieee;
use ieee.std_logic_1164.all;

entity PortaE is
    port
    (
        entradaA: in std_logic;
        entradaB: in std_logic;
        saida: out std_logic
    );
end PortaE;

architecture PE of PortaE is
begin
    saida <= entradaA and entradaB;
end PE;
```



Parada para exercícios !!

Simulação com VHDL

- Código é executado em um simulador
  - Não há um compilador de VHDL, não há um código executável
- Projeto do usuário
  - Especificado no estilo RTL
- Testbench: descrição VHDL para teste do projeto em desenvolvimento
  - Especificação comportamental
  - Interage com o projeto
  - Portável



Etapas de Projeto - Simulação

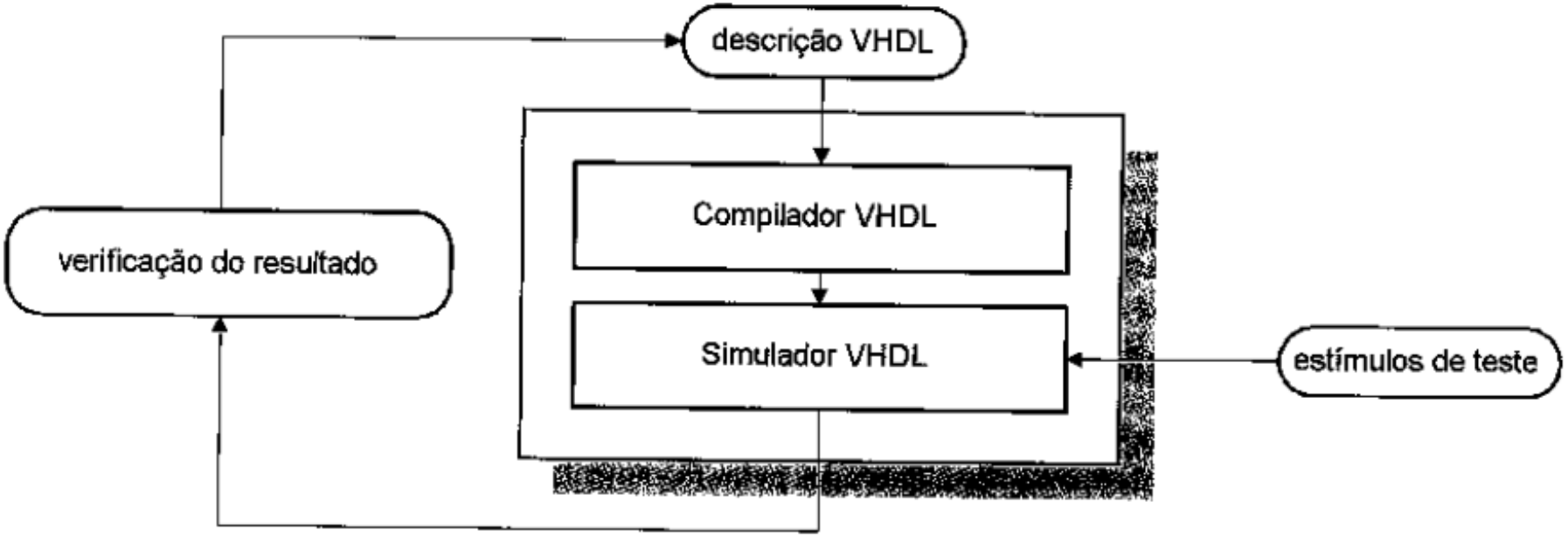


Figura 1.3.2 Elaboração da descrição VHDL.

*D'Amore

Etapas de Projeto - Síntese

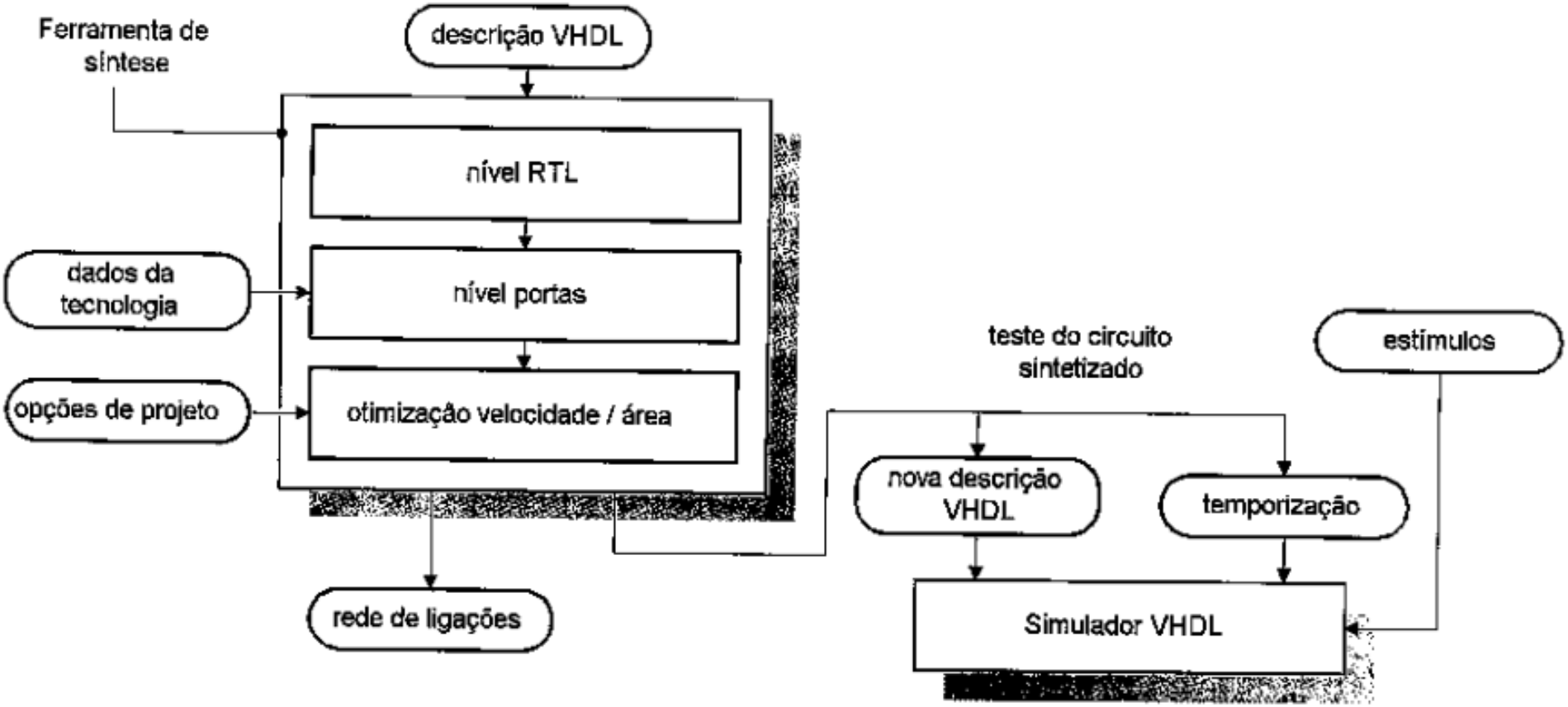


Figura 1.3.3 Síntese da descrição VHDL.

*D'Amore

Tipos Básicos (Escalares / Arrays)

- **bit**
  - Assume valores {‘0’, ‘1’}
  - **bit_vector**: tipo que designa conjunto de bits. Exemplo: “10001100” ou x”8C”
- **std_logic**
  - Semelhante ao tipo bit, mas permite assumir mais valores que permitem melhor análise na simulação
  - Assume valores {‘0’, ‘1’, ‘X’, ‘L’, ‘l’, H’, ‘h’, ‘Z’, ‘U’}
  - **std_logic_vector**: tipo que designa um conjunto de bits std_logic
- **boolean**
  - Assume valores {true, false}
  - Útil apenas para descrições abstratas, onde um sinal só pode assumir dois valores
- **Physical**
  - Representam uma medida: voltagem, capacitância, tempo
  - Tipos pré-definidos: fs, ps, ns, um, ms, sec, min, hr

Tipos Básicos (Escalares / Arrays)

- Real
  - Utilizado durante desenvolvimento da especificação
  - Exemplos: -1.0 / +2.35 / 37.0 / -1.5E+23
- Inteiros
  - Exemplos: +1 / 1232 / -1234
  - NÃO é possível realizar operações lógicas sobre inteiros (deve-se realizar a conversão explícita)
- Character
  - VHDL não é *case sensitive*, exceto para caracteres.
  - valor entre aspas simples: ‘a’, ‘x’, ‘0’, ‘1’, …
  - string: tipo que designa um conjunto de caracteres. Exemplo: “vhdl”

- OBSERVAÇÃO: **std_logic** e **physical** serão os tipos principais utilizados nesta disciplina

Variáveis e Sinais

- **Variáveis** tem comportamento equivalente ao software
- São utilizadas apenas em processos, sem temporização, atribuição imediata através do operador :=
- Exemplo
  - variable memoria : std_logic_vector (0 to 7);
  - variable x, y : integer;

- **Sinais** tem o comportamento equivalente ao hardware. São temporizados. Ou seja, a atribuição através do operador <= não é imediata
- Podem ser declarados em entity, architecture ou em package
- Não podem ser declarados em processos
- Exemplo
  - signal ground : std_logic := ‘0’;
  - signal bus : std_logic_vector (5 downto 1);

Declaração e Atribuição de Arrays

- **Declara-se a direção do array, informando os limites destes**
  - Exs.:

```
signal z_bus: std_logic_vector(3 downto 0);
signal c_bus: std_logic_vector(0 to 3);

z_bus <= c_bus;

  equivale a:

z_bus(3) <= c_bus(0)
z_bus(2) <= c_bus(1)
z_bus(1) <= c_bus(2)
z_bus(0) <= c_bus(3)
```

- **Observação**
  - tamanho dos arrays deve ser o mesmo
  - elementos são atribuídos por posição, pelo número do elemento

Porta Lógica → VHDL

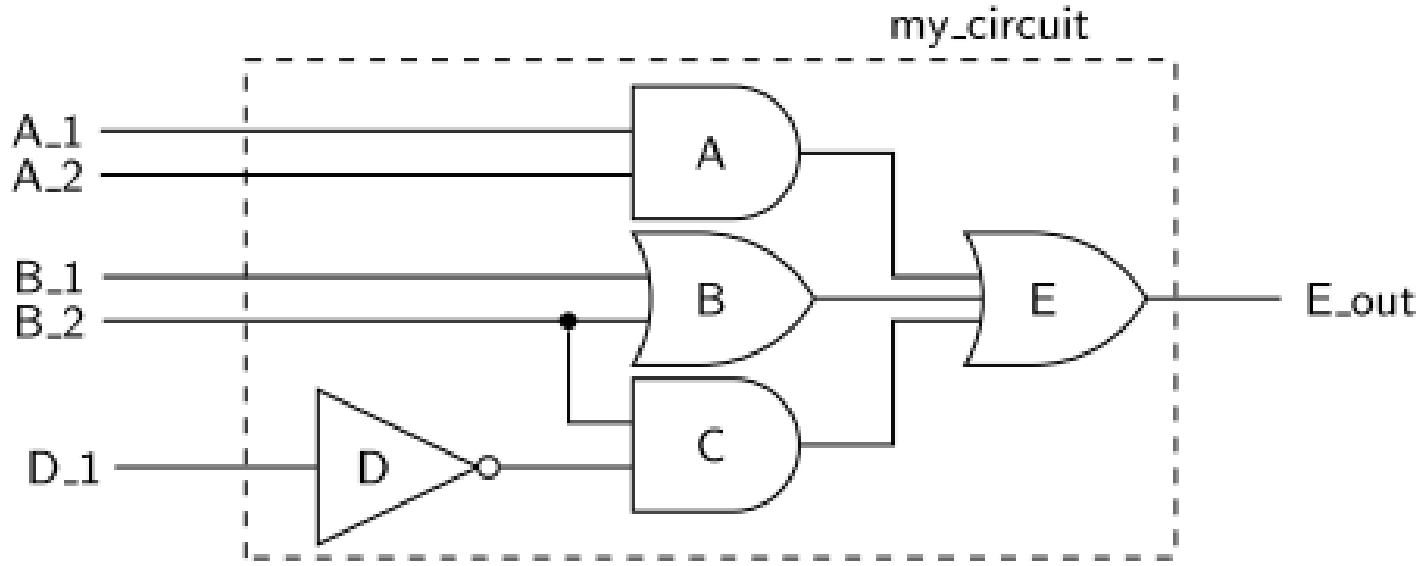


Figure 4.1: Some common circuit that is well known to `execute` parallel operations.

* Free range VHDL, pg 31

Porta Lógica → VHDL

Listing 4.1: VHDL code for the circuit of Figure 4.1.

```
-- library declaration
library IEEE;
use IEEE.std_logic_1164.all;

-- entity
entity my_circuit is
port ( A_1,A_2,B_1,B_2,D_1   : in  std_logic;
       E_out                 : out std_logic);
end my_circuit;

-- architecture
architecture my_circuit_arc of my_circuit is
    signal A_out, B_out, C_out : std_logic;
begin
    A_out <= A_1 and A_2;
    B_out <= B_1 or B_2;
    C_out <= (not D_1) and B_2;
    E_out <= A_out or B_out or C_out;
end my_circuit_arc;
```

Porta Lógica → VHDL

Listing 4.2: Equivalent VHDL code for the circuit of Figure 4.1.

```
C_out <= (not D_1) and B_2;
A_out <= A_1 and A_2;
B_out <= B_1 or B_2;
E_out <= A_out or B_out or C_out;
```

Listing 4.3: Equivalent VHDL code for the circuit of Figure 4.1.

```
A_out <= A_1 and A_2;
E_out <= A_out or B_out or C_out;
B_out <= B_1 or B_2;
C_out <= (not D_1) and B_2;
```

Listing 4.4: Equivalent VHDL code for the circuit of Figure 4.1.

```
B_out <= B_1 or B_2;
A_out <= A_1 and A_2;
E_out <= A_out or B_out or C_out;
C_out <= (not D_1) and B_2;
```

Expressão Booleana → VHDL

EXAMPLE 2. Write the VHDL code to implement the function expressed by the following logic equation: $F3 = \overline{L} \cdot \overline{M} \cdot N + L \cdot M$

* Free range VHDL, pg 36

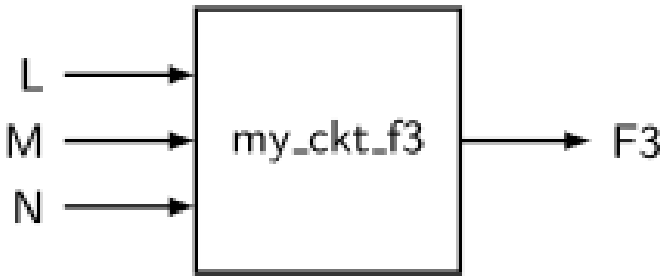
Expressão Booleana → VHDL

EXAMPLE 2. Write the VHDL code to implement the function expressed by the following logic equation: $F3 = \overline{L} \cdot \overline{M} \cdot N + L \cdot M$

SOLUTION. The black box diagram and associated VHDL code is shown in Listing 4.7.

Listing 4.7: Solution of Example 2.

```
-- library declaration
library IEEE;
use IEEE.std_logic_1164.all;
-- entity
entity my_ckt_f3 is
port ( L,M,N : in  std_logic;
       F3    : out std_logic);
end my_ckt_f3;
-- architecture
architecture f3_2 of my_ckt_f3 is
begin
F3<=((NOT L)AND(NOT M)AND N)OR(L AND M);
end f3_2;
```



* Free range VHDL, pg 36

Expressão Booleana → VHDL

Listing 4.8: Alternative solution of Example 2.

```
-- library declaration
library IEEE;
use IEEE.std_logic_1164.all;
-- entity
entity my_ckt_f3 is
port ( L,M,N : in  std_logic;
       F3    : out std_logic);
end my_ckt_f3;
-- architecture
architecture f3_1 of my_ckt_f3 is
    signal A1, A2 : std_logic; -- intermediate signals
begin
    A1 <= ((NOT L) AND (NOT M) AND N);
    A2 <= L AND M;
    F3 <= A1 OR A2;
end f3_1;
```

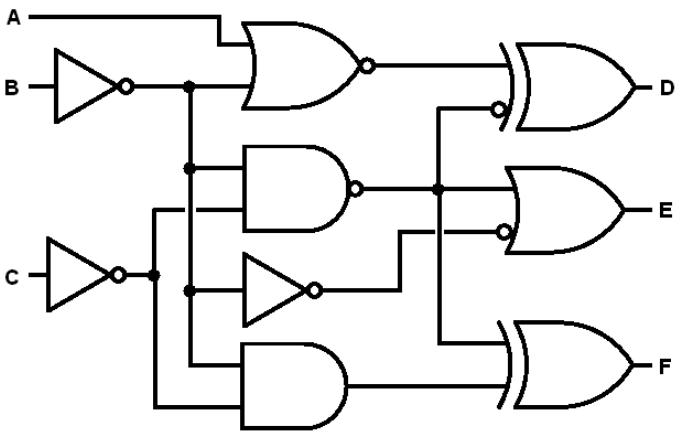
* Free range VHDL, pg 37

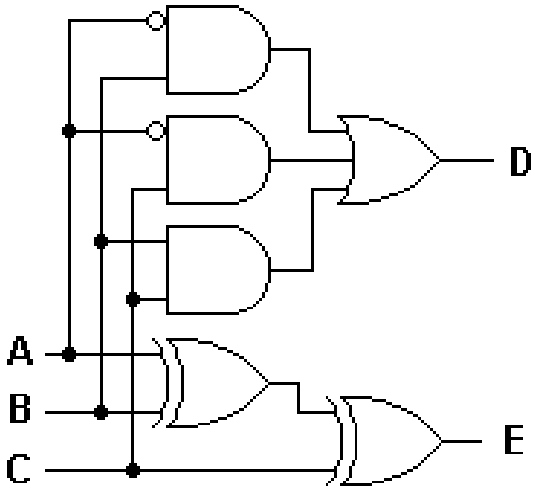
Exercícios

- Em aula
  - Descrever portas lógicas conhecidas em VHDL
- Extra-Classe
  - Faça a descrição de uma porta lógica OU de 4 entradas
  - Refaça o exercício acima, considerando agora que as entradas são um array
  - Descreva o trabalho 2 em VHDL

Exercícios de Conversão entre Representações – Extra-Classe

1. Dados os esquemáticos, obtenha descrições VHDL compatíveis





2. Dado o VHDL parcial obtenha o equivalente em esquemático, associando a cada nodo a função booleana equivalente

```
C(0) <= A(0) nand A(1) or (not b);
C(1) <= A(0) or A(1) or A(2);
C(2) <= (C(1) xor C(1)) and b;
s    <= (A(2) nor not(A(1))) xor b;
D    <= C;
```

Atividade da Próxima Aula

- Ler Capítulo 3 do Free Range VHDL

- Atividade p Próxima Aula
  - Entregar no ínicio da aula
  - Descrever em VHDL circuitos dos exemplos / exercícios das aulas anteriores
  - Escrito à mão !!!! Explicar a resolução do problema !!!
  - Pelo menos 5 módulos de complexidades diferentes