ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES I

Introdução à Linguagem VHDL

prof. Dr. César Augusto M. Marcon
prof. Dr. Edson Ifarraguirre Moreno

VHDL – Uma Linguagem de Descrição de Hardware

- **Originou como linguagem para descrever hardware, no contexto do programa americano**
  "Very High Speed Integrated Circuits" (VHSIC), iniciado em 1980
- **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
- **Favorece projeto descendente (“top-down design”)**
  - Projeto é inicialmente especificado de forma abstrata, com detalhamento posterior dos módulos
  - Exemplo : **A <= B + C after 5.0 ns;**

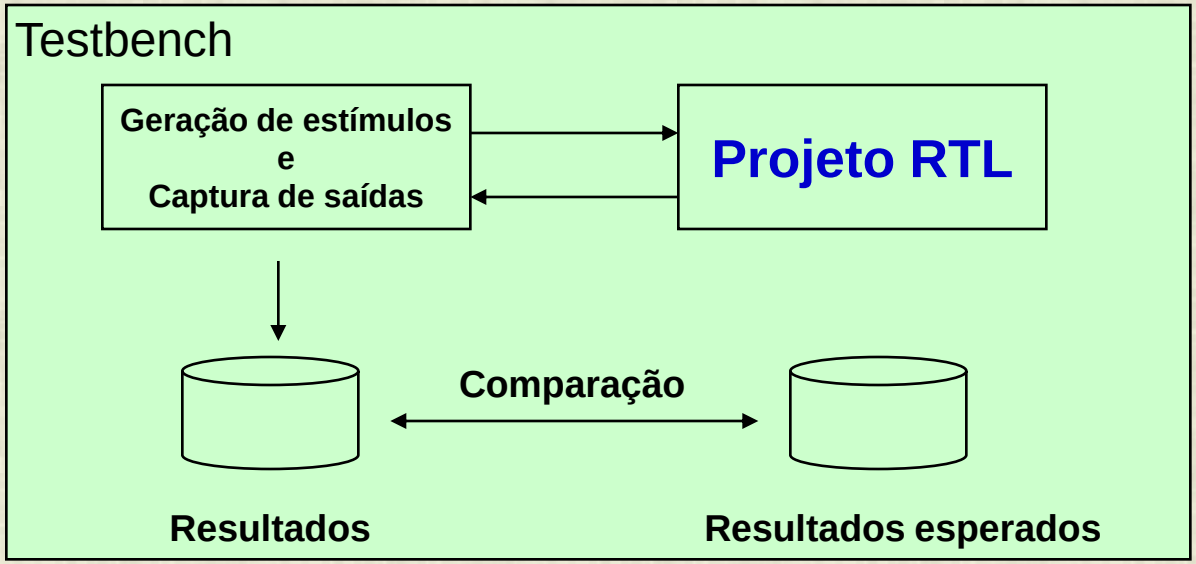
    A forma de realizar a soma pode ser decidida no momento da implementação (e.g. propagação rápida de vai-um, ou não, paralelo ou série, etc)

Estrutura de um programa VHDL

- Cada módulo tem sua própria “entity” e “architecture”
- As arquiteturas podem ser descritas tanto no nível comportamental quanto estrutural ou uma mistura
- Toda a comunicação ocorre através das portas declaradas em cada entity, observando-se o tipo, tamanho e a direção
- Várias funções e tipos básicos são armazenados em bibliotecas (library). A biblioteca “IEEE” sempre é incluída
  - **Praticamente todos os módulos escritos em VHDL iniciam com:**
    - **library** ieee;
    - **use** ieee.std_logic_1164.**all**;
- Biblioteca do usuário (default): work. Todos os arquivos contidos no diretório de trabalho fazem parte da biblioteca do usuário

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



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**
- **Exemplo**
  - variable memoria : std_logic_vector (0 to 7);
  - variable x, y : integer;

- **Sinais tem o comportamento equivalente ao hardware. São temporizados**
- **Podem ser declarados em entity, architecture ou em package**
- **Não podem ser declarados em processos, podendo serem utilizados no interior destes**
- **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

Expressões

– Expressões são fórmulas que realizam operações sobre objetos de mesmo tipo

- **Operações lógicas: and, or, nand, nor, xor, not**
- **Operações relacionais: =, /=, <, <=, >, >=**
- **Operações aritméticas: - (unária), abs**
- **Operações aritméticas: +, -**
- **Operações aritméticas: *, /**
- **Operações aritméticas: mod, rem, ****
- **Concatenação: &**
  - **Exemplo:** ***"1001" & "0011"*** ***resulta em*** ***"10010011"***

Menor

PRIORIDADE

Maior

– OBS.: Incluindo-se o pacote "**use ieee.STD_LOGIC_UNSIGNED.all**" pode-se somar vetores de bits (std_logic_vector) apenas com operador +

Exercícios

- Qual/quais das linhas abaixo é/são incorreta/s?
  - variable A, B, C, D : std_logic_vector (3 downto 0);
  - variable E,F,G : std_logic_vector (1 downto 0);
  - variable H,I,J,K : std_logic;
  - signal byte : std_logic_vector (0 to 7);
  - signal z_bus: std_logic_vector (3 downto 0);
  - signal c_bus: std_logic_vector (0 to 3);

1. [ ] A := B xor C and D;
2. [ ] H := I and J or K;
3. [ ] A := B and E;
4. [ ] H := I or F;
5. [ ] byte <= (others => ‘1’);
6. [ ] z_bus <= c_bus;

Atribuições Concorrentes

- **Atribuição de sinais**

```
alu_result <= op1 + op2;
```

- **Atribuição de sinais com escolha**

```
with alu_function select
  alu_result <=  op1 + op2          when alu_add | alu_incr,
                 op1 – op2          when alu_subtract,
                 op1 and op2        when alu_and,
                 op1 and not op2    when others;
```

- **Atribuição condicional de sinais**

```
mux_out <=  in_2   when  h = ’1' and sel = “00”   else
            in_0   when  h = ’1' and sel = “01”   else
            ‘Z’;
```

Exemplo de Descrição VHDL

- Descrição em VHDL de uma porta lógica E

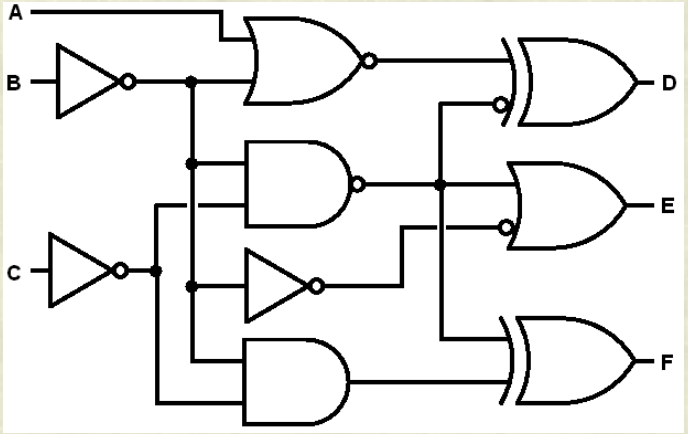
```
library ieee;
use ieee.std_logic_1164.all;

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;
```

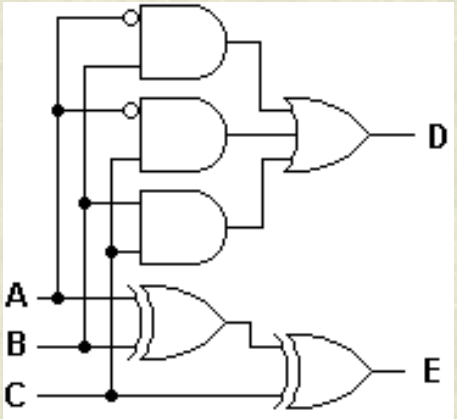
Exercícios

1. Faça a descrição de uma porta lógica OU de 4 entradas
2. Refaça o exercício acima, considerando agora que as entradas são um array
3. Planeje um circuito com 3 portas lógicas quaisquer, conecte as mesmas conforme desejado, e depois descreva o circuito. Este circuito deve ter 3 entradas e 3 saídas

Exercícios de Conversão entre Representações

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;
```