VHDL

Introdução

Paulo C. Centoducatte
ducatte@ic.unicamp.br

março de 2005

Introdução

- Vantagens do Uso de HDLs e Ferramentas de Sínteses
  - Aumento da produtividade, diminuindo o ciclo de desenvolvimento
  - Redução dos custos NRE (Non-Recurring Engineering)
  - Reusabilidade
  - Facilidade em introduzir alterações nos projetos
  - Exploração de alternativas de arquiteturas
  - Exploração de alternativas tecnológicas
  - Geração de circuitos testáveis automaticamente
  - Facilidades na verificação do projeto

Introdução

- Dispositivos ASICs e FPGAs
  - ASIC - Application-Specific Integrated Circuits
  - FPGA - Field-Programmable Gate-Array

- ASIC
  - Parcialmente manufaturado pelo fabricante
  - Gate Arrays
    - Channeled gate array
    - Channel-less gate array (sea-of-gates)
  - Standard Cell
    - Fabricante fornece uma biblioteca de had-macros e soft-macros

(70% a 90% do # de gates disponíveis são efetivamente usados)

Introdução

- FPGA
  - Completamente manufaturado pelo fabricante
  - Blocos programáveis interconectados por matrizes de chaves programáveis

ASIC x FPGA

	NRC	Por unidade
ASIC	$20,000 a $500,000	$10
FPGA	---	$150 a $250

- FPGAs são muito utilizadas para pequena produção e prototipagem de sistemas

Introdução

- Metodologia de Projeto
  - Descrição do sistema completo em um nível de abstração usando uma linguagem de descrição de hardware (HDL - Hardware Description Language) e uso de ferramentas automáticas para particionamento e síntese.
  - A descrição do hardware deve ser independente da tecnologia a ser usada na implementação.
    - PCB ou MCMs (multichip modules)
    - IC, ASICs, FPGA, PLD, full-custom

Hardware Description Languages (HDLs)

- HDL - Linguagem de programação usada para modelar a operação de um hardware
  - VHDL, Verilog, SystemC, AHDL, ...
- VHDL - História
  - 1980
    - USA Department of Defense (DOD)
      - Documentação
      - Metodologia de Projeto comum
      - re-usável com novas tecnologias
    - O DOD, dentro do programa “Very High Speed Integreted Circuit” (VHSIC) criou um projeto com a finalidade de criar uma linguagem de descrição de hardware
      - VHSIC Hardware Description Language (VHDL)

Hardware Description Languages (HDLs)

- 1983
  - Início do desenvolvimento de VHDL
    - IBM, Texas Instruments e Intermetrics
- 1987
  - Todo projeto de eletrônica digital ligado ao DOD deveria ser descrito em VHDL
  - IEEE - Institute of Electrical and Electronics Engineers
    - IEEE Standard 1076 (http//www.vhdl.org)
  - F-22
    - Todos os subsistemas eletrônicos descritos em VHDL
    - O desenvolvimento dos subsistemas foram distribuídos em diversos subcontratos
    - Estabeleceu um marco no uso de VHDL e da metodologia de projeto Top-Down

Hardware Description Languages (HDLs)

- 1993
  - Revisão de VHDL - IEEE 1076'93
- 1996
  - Ferramentas comerciais para Simulação e Sínteses para o padrão IEEE 1076'93
  - IEEE 1076.3 - VHDL package para uso com ferramentas de sínteses
  - IEEE 1076.4 (VITAL) - padrão para modelagem de bibliotecas para ASICs e FPGAs em VHDL
- 2000
  - Revisão de VHDL – IEEE 1076a
- 200x
  - Próxima revisão de VHDL

Ferramentas para Automação de Projetos

- Computer Aided Design (CAD) e Computer Aided Engineering (CAE)
  - Simulação
  - Fault Simulation
  - Register Transfer Level Synthesis
    - Otimizações no nível RTL
    - Otimizações no nível lógico
  - Síntese de Teste

Síntese e Otimizações

- Restrições

  – Global - Aplicada igualmente a todo o projeto
    - Exemplos: Biblioteca do fabricante; tensão e temperatura de uso.

  – Específica por circuito - Aplicada a um particular circuito
    - Exemplos:
      – Área - área máxima (# gates equivalentes, # transistores)
      – Timing - input e output loading, máx. fan-out, capacidade de driving das entradas, mínima freqüência do clock, etc.
      – Potência - máxima potência consumida
      – Testabilidade - tipo das células para scan, scan parcial ou full, boundary scan.

VHDL- Uma visão Geral

- Característica
  - Adequada à descrição de hardware
    - programação seqüencial e paralela
  - Permite descrição em diferentes níveis de abstração
    - Comportamental
    - RTL
    - Estrutural (portas lógicas)
  - Simulável
  - Sintetizável
  - Padrão

VHDL - Uma visão Geral

- Conceitos:
  - time step: É o menor intervalo de tempo do simulador, em algumas ferramentas é definido pelo usuário. Para VHDL time-step é o tempo gasto para a resolução de uma iteração de todos os comandos concorrentes
  - Concorrência: A cada time-step do simulador todos os comandos são executados concorrentemente. Também os processos (conjunto de comandos seqüenciais) ocorrem em concorrência com o restante dos comandos de um modelo VHDL
  - Tipo: A linguagem VHDL é fortemente dependente dos tipos dos dados.

VHDL - uma visão geral

- 5 tipos de unidades
  - Entity - define a interface do projeto, módulo, etc.
  - Architecture - descreve funcionalmente a entidade. Pode haver mais de uma arquitetura para uma mesma entidade.
  - Package - declarações comuns a todo o projeto. Exemplo: constantes, tipos de dados e subprogramas.
  - Packge Body - contém o corpo dos subprogramas definidos no Packge
  - Configuration - Faz a ligação de uma entidade com uma particular arquitetura, formando um componente.

VHDL - uma visão geral

- Packages: Assim como em linguagens de programação são utilizadas bibliotecas (de funções, procedimentos, definições de tipos e declarações de constantes etc), em VHDL isto é feito com a utilização de Packages e Bibliotecas de componentes.

- Entidades: Define a interface de um componente: nome, tipo dos sinais de entrada e/ou saída, ...

- Arquiteturas: Define a funcionalidade de um componente e a temporização. Uma mesma entidade pode possuir múltiplas arquiteturas e para efeito de simulação e síntese é usada a última arquitetura compilada.

VHDL - uma visão geral

- Processo: É uma porção de código delimitada pelas palavras Process e End Process que contém comandos seqüenciais (são simulados em delta-delay que somados resultam em zero). Todos os processos de um modelo VHDL de um componente são executados concorrentemente em um time-step.

- Função: Uma função em VHDL tem comportamento similar as funções em outras linguagens, contudo funções em VHDL não afetam os parâmetros de entrada, simplesmente retornam um valor com tipo definido.

VHDL - uma visão geral

- Procedimento: Um procedimento em VHDL tem comportamento similar aos procedimentos de outras linguagens e distinguem-se das funções pela possibilidade de alteração nos parâmetros de chamada.

- Bloco: O comando BLOCK pode ser utilizado tanto para definir hierarquia de circuitos como também em conjunto com expressões GUARD definindo condições de uso de sinais de escopo restrito.

- Componente: É descrito pelo par entidade e arquitetura. Um modelo VHDL é dito estrutural se faz uso de instanciação de componentes.

VHDL - Estrutura do Código VHDL

- Declarações
  - Objetos que serão usados em comandos concorrentes ou seqüenciais
  - Declarados antes da clausula **begin** em arquiteturas, blocos, processos, procedimentos e funções

- Comandos concorrentes
  - comandos que serão executados em paralelo, independentemente uns dos outros
  - **block** e **process**

VHDL - Estrutura do Código VHDL

- Comandos seqüenciais
  - comandos que serão executados de forma seqüencial, obedecendo o fluxo de controle
  - comandos após a clusula **begin** em processos

VHDL - Elementos Léxicos

- Comentários: --
- Identificadores: Formados por letras, números e underline (necessariamente iniciados por letra e não podem terminar em underline)
- Palavras reservadas
- Símbolos especiais: Utilizados em operadores, para delimitação e pontuação da linguagem:

/ : - . + | & ‘ “ ( ) * , => > = < /= >= <= <> ;

VHDL - Elementos Léxicos (cont.)

- Números, caracteres e string
  - Números inteiros: 17 25 0 55E5
  - Números reais: 14.33 0.0 33.4E10
  - caracteres: ‘A’ ‘f’ ‘K’ ‘7’ ‘?’
  - String: “String” “A”
- Bit strings: Somente utilizados com o tipo bit_vector
  - Binário: B”0011011”
  - Octal: O”234”
  - Hexadecimal: X”E3F”

VHDL - Entidade

- Define a interface do componente com o restante do sistema

```
Entity nome IS
    GENERIC (lista_dos_genericos);
     PORT (lista_dos_ports);
END nome;
```

VHDL - Port

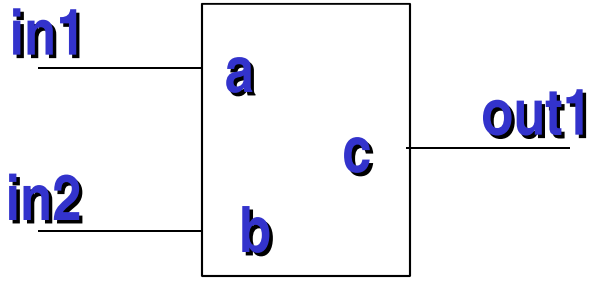
- Formato:

Port (nome: MODO tipo);

  - Modo: in, out, inout, buffer, linkage

- Exemplo:

```
Port (a, b : in   bit;
      c:  out bit);
```



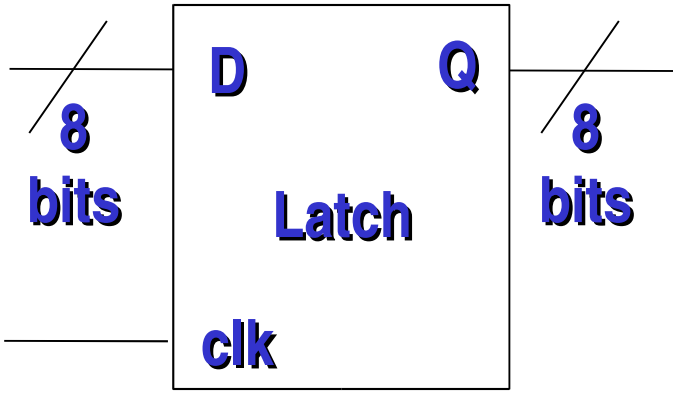
VHDL - Exemplo

```
Entity porta_and IS
   GENERIC (numero_de_entradas: integer := 4);
   PORT (entradas : in bit_vector (1 to numero_de_entradas);
         saida    : out bit);
END porta_end;
```

- Port MODE:
  - IN: sinal é somente de entrada
  - OUT: sinal é somente de saída
  - Buffer: sinal é de entrada e saída (um de cada vez)
  - Inout: sinal é bidirecional, implica em um BUS
  - Linkage: direção do sinal é desconhecida

VHDL - Exemplo

- Qual é a Entidade?




```
Entity latch is
  port (d    : in   bit_vector(7 downto 0);
        clk :  in   bit;
        q    :  out out bit_vector(7 downto 0);
End latch;
```

VHDL - Exemplo

- Usando generic?

```
Entity latch is
  generic(w : integer := 8);
  port (d    : in   bit_vector(w-1 downto 0);
        clk  : in   bit;
        q    : out bit_vector(w-1 downto 0);
End latch;
```

VHDL - Arquitetura

- Estabelece a relação entre entradas e saídas
  - Funções
  - Procedimentos
  - Execução paralela de processos
  - Instanciação de componentes

- Arquiteturas múltiplas
  - Utiliza a última compilada

VHDL - Arquitetura

```
ARCHITECTURE  label OF nome_entidade IS
  --  parte declarativa (declarações de tipos, subtipos, sinais,
  --  funções, procedimentos, ...)

BEGIN
      --  comandos concorrentes
END label;
```

```
ARCHITECTURE  rtl OF porta_and IS
  constant atraso : time := 5 ns;

BEGIN
      y <=  a AND b AFTER atraso;
END porta_and;
```

Qual a entidade?

VHDL - Arquitetura

```
Entity porta_and is
  port (a, b  : in   bit;
        y     :  out bit);
End porta_and;
```

```
Entity porta_and is
  generic(w : integer := 8);
  port (a, b   :  in   bit_vector(w-1 downto 0);
        y      :  out bit_vector(w-1 downto 0);
End latch;
```

VHDL - Declarações

- Constantes:
  - CONSTANT atraso : TIME := 3 ns;
- Variáveis:
  - VARIABLE tmp : INTEGER := 0;
- SINAIS:
  - SIGNAL clk : BIT := ‘1’;

VHDL - Declarações

- Constante: nome assinalado a um valor fixo
  - » CONSTANT vdd: real := 4.5;
  - » CONSTANT cinco: integer := 3 +2;

- Variável: nome assinalado a um valor que muda de acordo com um determinado processo
  - » VARIABLE largura_pulso: time range 1ns to 15ns := 3ns;
  - » VARIABLE memoria: bit_vector (0 to 7);

- SINAl: conectam entidades e transmitem mudanças de valores entre os processos (todo port é um sinal).
  - » SIGNAL contador : integer range 0 to 63
  - » SIGNAL condicao : boolean := TRUE;

VHDL - Declarações

- Áreas declarativas
  - Onde são definidos os sinais internos, variáveis, constantes, subprogramas, aliases
  - Àreas declarativas existem para packages, entidades, arquiteturas, subprogramas, processos e blocos
  - A área não declarativa é chamada de área de comandos

VHDL - Declarações

- Área declarativa em Arquitetura
  - Declarações no topo da arquitetura são “visíveis” em toda a arquitetura

```
ARCHITECTURE exemplo OF circuito IS
  CONSTANT cte : time := 10 ns;
  SIGNAL  tmp : integer;
  SIGNAL  cnt  : bit;

BEGIN

    ....
```

VHDL - Declarações

- Área declarativa em Processos
  - Declarações no topo de um processo são “visíveis” em toda o processo

```
Exemplo: PROCESS is
  CONSTANT cte : time := 10 ns;
  VARIABLE  tmp : integer;
  SIGNAL  cnt  : bit;

BEGIN

   ....
```

VHDL - Declarações

- Escopo: da declaração do identificador até a declarção END desta região

- Limites:
  - Componente
    - Entidade
      - Arquitetura
        - » Bloco
        - » Processo
        - » Subprograma

VHDL - Escopo

- Na linguagem VHDL é possível a utilização de identificadores homônimos com diferentes significados, dependendo do contexto onde é definido cada identificador ele pode assumir diferentes significados a nível lógico.
- Um sinal definido dentro da parte declarativa de um componente, entidade, arquitetura, bloco, processo ou subprograma tem o escopo controlado dentro deste contexto. Desta forma é possível a utilização de nomes idênticos para indicações de sinais distintos.
- Para a distinção de sinais homônimos, cada sinal definido em VHDL pode ser acessado por seu endereço completo, indicando biblioteca, package, componente, arquitetura, processo e nome do sinal na forma:

biblioteca.componente.arquitetura.processo.sinal

VHDL - Visibilidade

- Estabelece o significado dos identificadores
- As declarações são visíveis somente no seu escopo
- Uso de endereços em identificadores

» var1 := architecture2.cte;
» var2 := process1.cte;

VHDL - Atribuições

- Atribuição a sinal:

  <=

- Atribuição a variável:

  :=

- Inicialização (constante, sinal e variável):

  :=

VHDL - Atribuições de Dados

- Constante:

```
Constant cnt_reset : bit_vector(0 to 3) := “1001”;
```

- Variável:

```
var1 := 2005;
```

- Sinal:

```
dado <= ‘1’;
 d <= ‘1’, ‘0’ AFTER 5 ns;
```

VHDL - Operadores Aritméticos

Operador	Operação	Exemplo
+	Adição	I := i + 2;
-	Subtração	J <= j -10;
*	Multiplicação	M := fator * n;
/	Divisão	K := i / 2;
**	Potenciação	I := i ** 3;
ABS	Valor absoluto	Y <= ABS(tmp)
MOD	Módulo	Z <= MOD(t);
REM	resto	R <= REM(tot);

VHDL - Operadores Lógicos

- Predefinidos para os tipos: bit; std_logic, std_ulogic, boolean
  - NOT
  - AND
  - NAND
  - OR
  - NOR
  - XOR
  - XNOR

VHDL - Operadores Relacionais

Operador	Operação
=	Igual
/=	Diferente
>	Maior que
<	Menor que
>=	Maior ou igual que
<=	Menor ou igual que

VHDL - Modelo Completo

- Indica as bibliotecas utilizadas
- Faz uso das definições contidas nas bibliotecas
- Descreve a entidade
- Descreve a arquitetura

VHDL – or de 2 entradas

```
Library IEEE;
Use IEEE.std_logic.all;

Entity or2 is
  --  porta or de duas entradas
  port (i1, i2 :  in   bit;
              out1 : out bit);
End or2;

Architecture rtl of or2 is
Begin
     out1 <= i1 or i2;
End rtl;
```

VHDL - Packages

- Coleção de declarações comuns definidas fora dos modelos (corpo + declaração)

- Uso de:
  - Tipos; Subtipos
  - Subprogramas (Funções e Procedimentos)
  - Constantes; Sinais; Aliases
  - Atributos
  - Componentes

VHDL - Packages

```
PACKAGE label IS

   -- declarações;

END label;

Package BODY label IS

   -- Corpo de subprogramas;

END label;
```

VHDL – Library e USE

- LIBRARY nome_da_biblioteca;

```
Library IEEE;
```

- A cláusula USE torna os pacotes visíveis em entidades e arquiteturas
  - USE nome_biblioteca.nome_package;
  - USE nome_package.identificador;

```
USE IEEE.std_ulogic.all;
USE math.all;
USE textio.all;
```

VHDL - Tipos

- Definição de tipos
  - Integer
  - Natural
  - Real
  - Array
  - ....
- Tipos padrão
  - bit
  - std_logic
  - std_ulogic

VHDL - Tipos

- Vetores
  - bit_vector
  - std_logic_vector
  - std_ulogic_vector

- Tipos são associados a:
  - sinais
  - variáveis
  - constantes

VHDL - Tipos de Dados

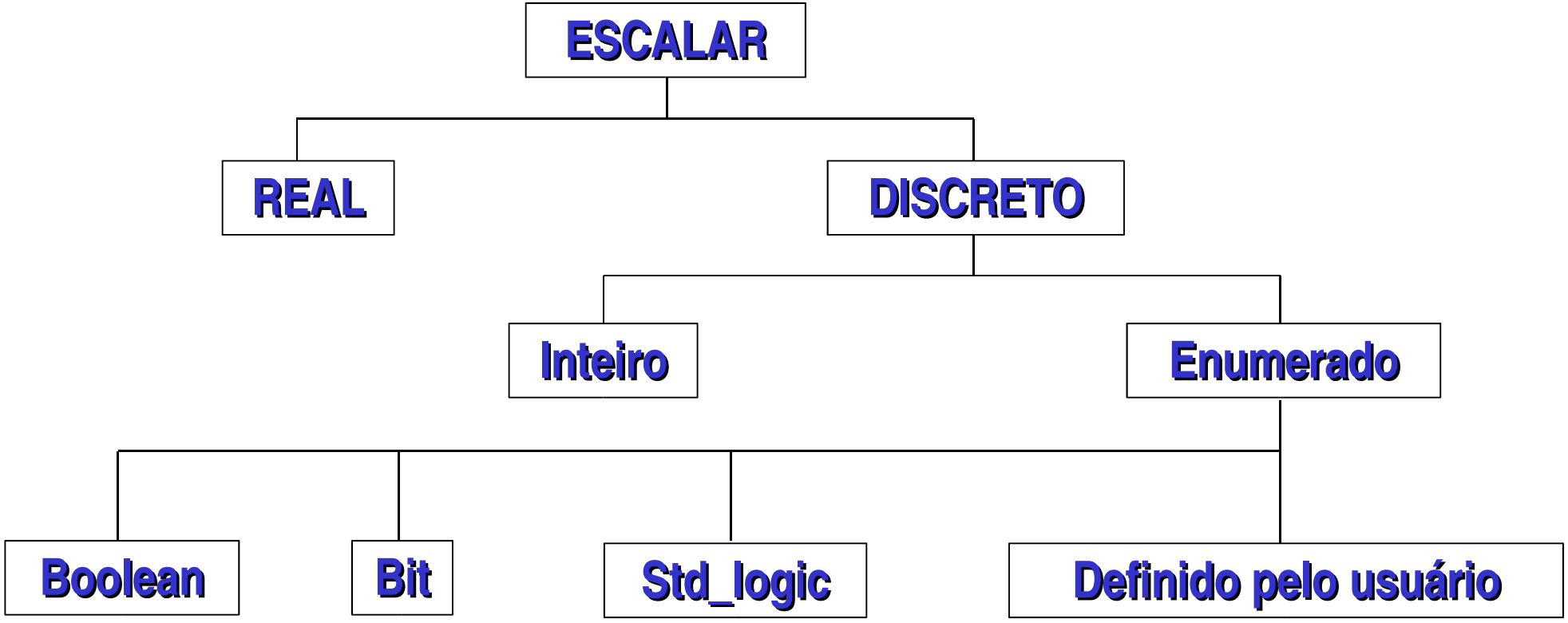
- Scalar
  - Numeric
  - Enumeration
  - Physical
- Composite
  - Array
  - Records
- Acess
  - Similar aos pointers em linguagens de programação
  - Permite alocação dinâmica de memória

VHDL - Tipo Scalar

- NUMERIC
  - TYPE integer IS RANGE -214748368 to 2147483647;
  - TYPE real is RANGE -1.797769E308 to 1.79769E308;
- ENUMERATION
  - TYPE instr IS (“add”, “sub”, “mult”, “div”);
  - TYPE character IS (‘a’, ‘b’, ‘c’, ‘d’, ...);
- PHYSICAL

```
TYPE time IS RANGE 0 to 1e55
    UNITS
        fs;
        ps = 1000 fs;
        ...
    END UNITS;
```

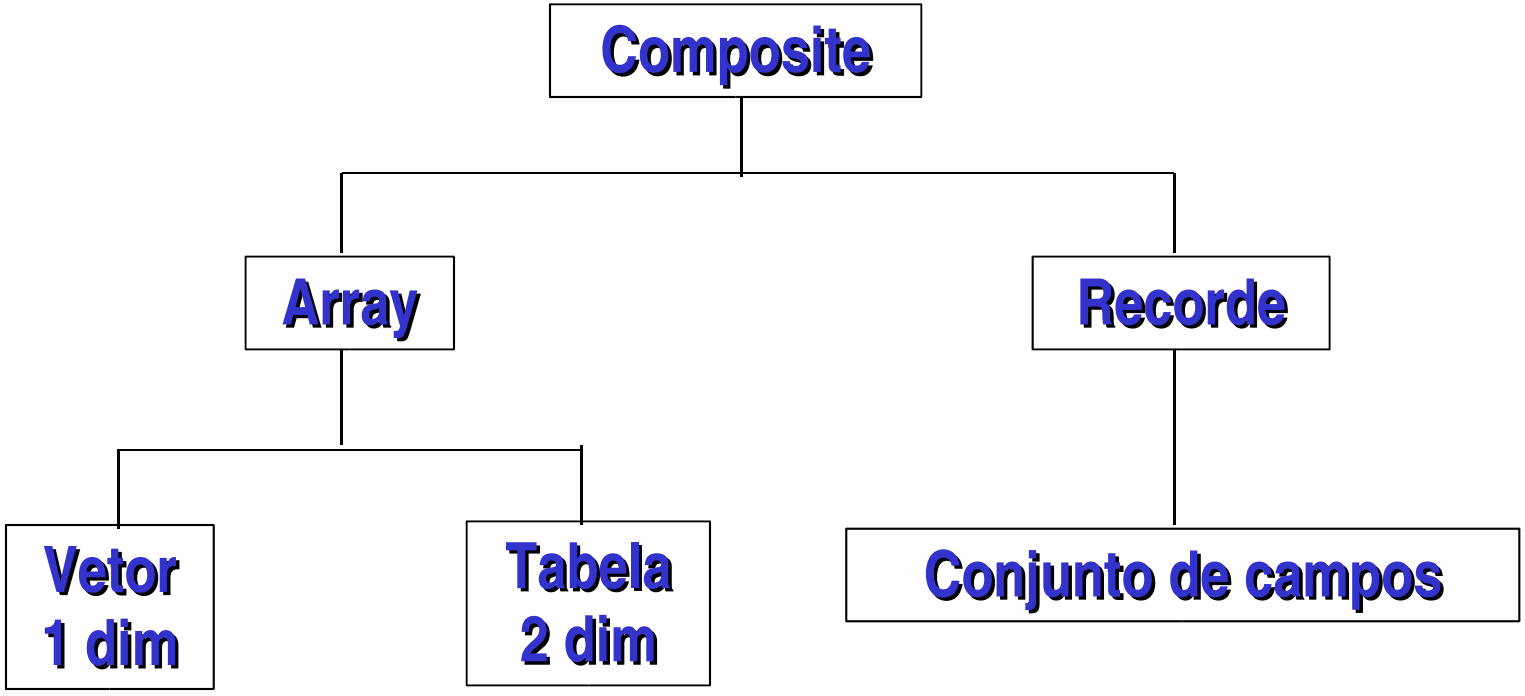
VHDL - Tipo Escalar



VHDL - Tipo Composite

- Consiste de elementos de dados relacionados na forma de um array ou record
- Os elementos de um composite podem ser manipulados como um objeto ÚNICO ou individualmente
- Tipicamente utilizados para modelar barramentos e memórias (RAM e ROM)

VHDL - Tipo Composite



VHDL - Tipo - ARRAY

- Formato:

```
TYPE word IS ARRAY (31 downto 0) OF bit;

SUBTYPE coef IS integer range 0 to 15;

TYPE matriz IS array(coef,word) of word;
```

VHDL - Tipo - RECORD

- Formato:

```
TYPE time_stamp IS RECORD
        segundos : integer range 0 to 59;
        minutos  : integer range 0 to 59;
        horas    : integer range 0 to 23;
END RECORD time_stamp;
```

- Podem ser acessados elementos individuais do record ou como um todo

VHDL - Tipo - RECORD

- Exemplo de uso:

```
constant meia_noite : time_stamp := (0,0,0);

constant meio_dia : time_stamp :=
(horas => 12, minutos => 0, segundos => 0);

horario_atual := meio_dia;
hora_atual := meia_noite.horas;
```

VHDL - Tipo Pré-Definidos

- BIT ('0', '1')
- BOOLEAN (TRUE, FALSE)
- INTEGER (-1, 1, 6, 10, 15, ...)
- REAL (-4.3, 5.5, 0.35 ....)
- CHARACTER ('a', 'b', 'c', ...)
- TIME (5 ns, 1 ps, 2s, ...)
- SEVERITY_LEVEL (note, warning, error, failure)

VHDL - Tipo Standard Logic

- Definidos no pacote IEEE:

```
TYPE std_logic IS ( 'U',  --  não inicializado
                    'X'  --  indeterminado forte
                    '0'  --  zero lógico forte
                    '1'  --  um lógico forte
                    'Z'  --  alta impedância
                    'W' --  indeterminado fraco
                    'L'  --  zero lógico fraco
                    'H'  --  um lógico fraco
                    '-'  --  dont´t care
                  );
```

VHDL - Subtipos

- Formato:

SUBTYPE natural IS INTEGER RANGE 0 to 2147483647;

SUBTYPE positive IS INTEGER RANGE 1 to 2147483647;

VHDL - Conversão de Tipos

- Possível para conversão de alguns tipos
  - Inteiro --> Ponto Flutuante
  - array --> array com conteúdo do mesmo tipo

- Formato:

  tipo(objeto)

- Exemplo:

  soma := REAL(N1) + N2;

VHDL - Exemplo

- Qual o circuito implementado (em termos de portas lógicas)?

```
Library IEEE;
use IEEE.std_logic;
Entity cct1 IS
   Port (in1, in2,in3,in4 : IN std_logic;
         out1, out2 : OUT std_logic);
End cct1;
Architecture rtl OF cct1 IS
  signal out_i : std_logic;
Begin
   out_i  <=  in1 and in2 and in3;
   out1   <=  out_i;
   out2   <=  in4 XOR out_i;
End rtl
```

VHDL - Modelando Atrasos

- Em quanto tempo a saída out1 do circuito anterior é resolvida?
- Atraso interno (delta)

```
dado <= dado_interno;
```

- Atraso inercial (filtra spikes)

```
Q <= d_in AFTER 10ns;
```

- Atraso transport (passa o sinal independentemente da sua duração)

```
Q <= TRANSPORT d_in AFTER 10ns;
```

VHDL - Modelando Atrasos

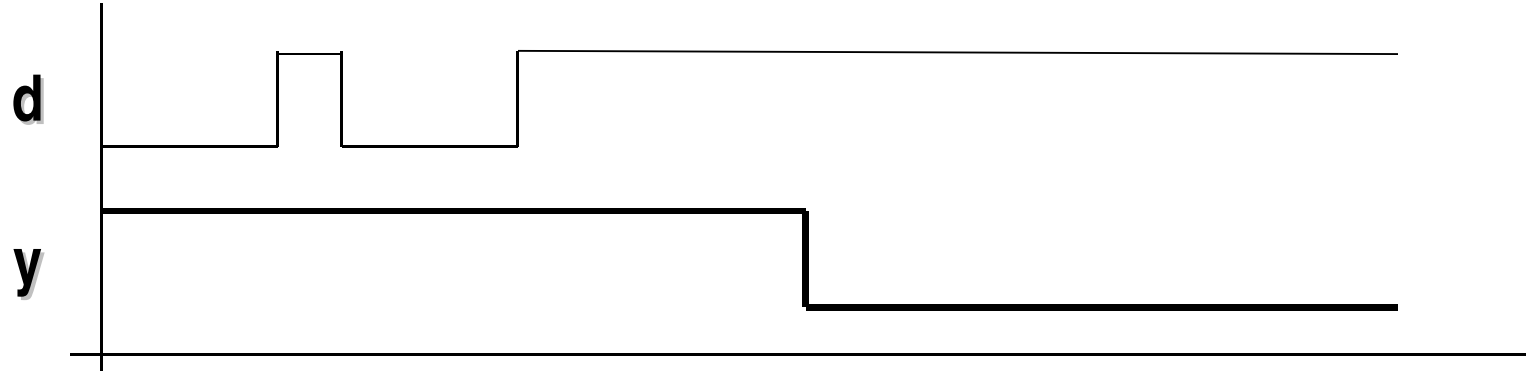
- Um sinal não muda imediatamente quando assinalamento ocorre

- Um assinalamento de sinal é programado para ocorrer no próximo ciclo de simulação

VHDL - Inertial Delay

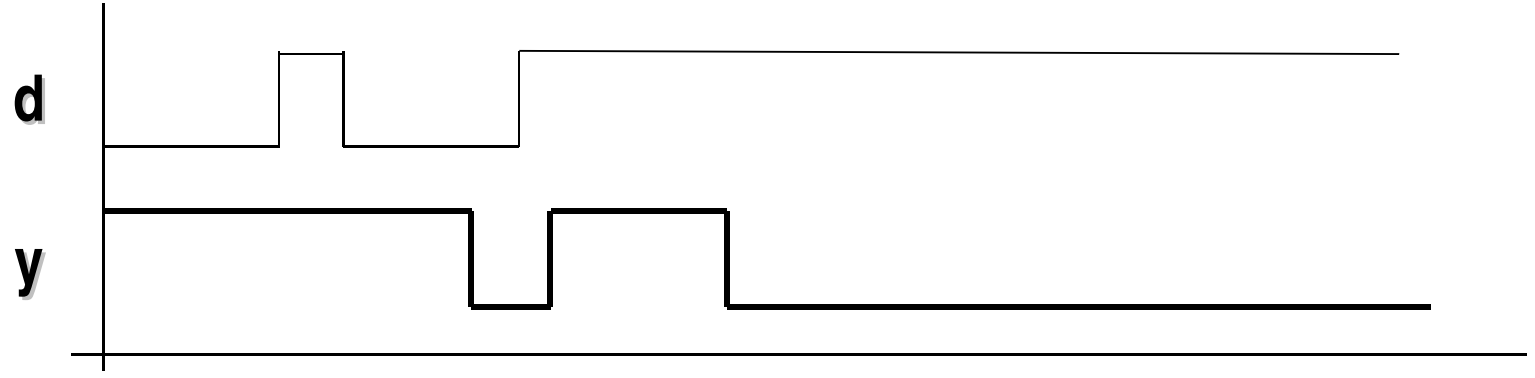
- O atraso inercial rejeita pulsos que são menores que o valor de atraso especificado

- O comando y <= not d after 20 ns; pode ser lido como y recebe o valor d se este tiver duração de pelo menos 20 ns



VHDL - Transport Delay

- modela resposta em freqüência infinita
- não importa a duração do pulso copiado para a saída
- se o atraso de uma nova transição é menor ou igual ao atraso das transições programadas, então estas transições são substituídas pela nova transição, caso este atraso seja maior as transições são somadas



VHDL - Processos

- São comandos concorrentes que contém instruções seqüênciais
- Todos os processos são executados concorrentemente
- Cada instrução dentro de um processo leva um tempo “delta time” que somados são iguais a zero

VHDL - Process

```
Label: PROCESS (lista de sensitividade)
        cláusulas declarativas;
      BEGIN
          --  comentários
          inicializações;
          cláusulas acertivas;
       END PROCESS label;
```

OBS.: A lista de sensibilidade funciona como um comando WAIT ON

VHDL - Process

```
Architecture bhv OF generic_decoder IS
Begin
PROCESS (sel, en)
  BEGIN
    y <= (others => '1');
    FOR i IN y'range LOOP
      IF (en = '1' and (bvtoi(To_Bitvector(sel)) = i))
      THEN
        y(i) <= '0';
      END IF;
    END LOOP;
END PROCESS;
END bhv;
```

atributo

Função de conversão de tipo bit_vector para inteiro

VHDL - Assinalamento seqüencial

```
Process  (a,b)
    variable  i : integer := 0;
begin
   c <= '1';
    i :=  1;
    t <= '0' after 5 ns;
    i := i -1;
    d <= transport '1' after 15 ns;
    c <= '1' after 20 ns;
    d <= '0' after 25 ns;
end process;
```

VHDL - Assinalamento seqüencial

```
Process  (a,b)
    variable  i : integer := 0;
begin
   c <= '1';                              --  c = '1' antes do próximo time step
    i :=  1;                              --  i = 1  imediatamente
    t <= '0' after 5 ns;                  --  t = '0' depois de 5 ns
    i := i -1;                            --   i = 0  imediatamente
    d <= transport '1' after 15 ns;       --  d = '1' depois de 15 ns
    c <= '1' after 20 ns;                 --   c = '1' depois de 20 ns
    d <= '0' after 25 ns;                 --   d = '0' depois de 25 ns
end process;
```

VHDL - Comando WAIT

- Utilizado dentro da estrutura PROCESS
- Formato

WAIT [ON lista]

espera atividade em algum sinal da lista

WAIT [UNTIL condição]

espera até que condição ocorra

WAIT [FOR tempo]

espera pelo tempo especificado

VHDL - Comando WAIT

```
Process
    variable i : integer := 0;
begin
    i := i + 1;
    j <= j + 1;
    wait on k;
    wait for 100 ns;
    wait on m until j > 5;
end process;
```

VHDL - Comando WAIT

wait; -- espera infinita

wait until clk = '1'; -- espera até que um evento satisfaça a condição clk = '1'

wait on clk until reset = '0'; -- espera até que ocorra um evento em clk e que se satisfaça a condição reset ='0'

wait until trigger = '1' for 1 ms; -- espera até que ocorra um evento em trigger ou se passe 1 milisegundo

VHDL - Comando wait

- Exemplo:

```
exemplo: process
    variable var : integr := 0;
    signal x, y : bit := '0';
begin
    var := var + 1;
    var := var + var;
    x <= not y;
    x <= y;
    wait;   --  por quanto tempo?
End process exemplo;
```

Exerc.: Escrever um modelo para um latch tipo D

```
Library ieee;
use ieee.std_logic;
entity latch_d is
  port (dado, enable : in std_logic;
        q, notq : out std_logic);
end latch_d;
```

```
architecture rtl of latch_d is
begin
latch: process
begin
   wait until enable = '1'
   q <= dado;
   notq <= not dado;
end process latch;
end rtl;
```

VHDL - Comandos seqüenciais

- Chamada de procedimentos (também concorrente)
- Assertion (também concorrente)
- Assinalamento de sinal (também concorrente)
- Assinalamento de variáveis
- WAIT
- IF, CASE
- NEXT
- EXIT, RETURN, LOOP
- NULL

VHDL - Atribuição Condicional

- Cláusula IF
- Cláusula WHEN
- Cláusula CASE

VHDL - Cláusula IF

```
IF (condição 1) THEN
        Cláusula acertiva 1;
ELSEIF (condição 2) THEN
        Cláusula acertiva 2;
ELSE
        Cláusula acertiva 3;
END IF;
```

VHDL - Cláusula IF

```
Process (acao, cor)
   IF cor = verde  THEN
         acao <= va_em_frente;
   ELSEIF cor = amarelo THEN
          acao <= atencao;
   ELSE
          acao <= pare;
   END IF;
```

VHDL - Cláusula IF

```
latchD: process (dado,enable)
begin
    IF enable = '1' THEN
            Q <= dado;
            notQ <= not dado
     END IF;
end process latchD;
```

VHDL - Cláusula IF

```
Architecture rtl of is
  signal prec : std_ulogic;
  signal Xsp : std_logic_vector(7 downto 0) := "00000000";
begin                                          -- Conversor paralelo/serie
process(clk)
begin
   if (clk´event and clk´last_value = '0' and clk = '1') then
      if  en_sr = '0'  then                    -- shift left
          ultimo_bit <= Xsp(7);
          Xsp(7 downto 1) <= Xsp(6 downto 0);
          Xsp(0) <= IO;
     elseif em_load = '1' then                 -- carga paralela
          Xsp <= bus_data;
    end if;    end if;
end process;
end rtl;
```

VHDL - Escrever o processo de um contador de 4 bits

VHDL - Escrever o processo de um contador de 4 bits

```
Architecture rtl of contador is
  signal cont : unsigned(3 downto 0) := “0000”;
begin
   process(reset,clk)
   begin
      if reset = ‘1’ then
         cont <= “0000”;
      elseif (clk´event and clk = ‘1’) then
         if enable = ‘1’ then
            if incr = ‘1’ then cont <= cont + “0001”;
            else  cont <= cont - “0001”;
            end if;
         end if;
       end if;  end process;  end rtl;
```

VHDL - Cláusula When (concorrente)

Atribuição WHEN condição

sinal <= atribuição WHEN valor_expressão

Exemplo:

acao <= prosseguir WHEN sinal_verde;

VHDL - Seletor

```
out <= in1 WHEN sel
       else in0;
```

```
if sel = '1' then
    out <= in1;
else
    out <= in0;
```

```
out <=  in0 WHEN sel0 else
        in1 WHEN sel1 else
        in2 WHEN sel2 else
        in3 WHEN sel3 else
        inx;
```

VHDL- Cláusula With (concorrente)

- WITH expressão SELECT
  signal <= atribuição WHEN valor_expressão

Exemplo:

```
WITH cor SELECT
    acao  <=  prosseguir WHEN   “verde”,
              parar      WHEN   “vermelho”,
              alerta     WHEN   OTHERS;
```

VHDL- Cláusula With

```
with regsel select
    z <= A after Tprop when “00’,
         B after Tprop when “01”,
         C after Tprop when “10”,
         D after Tprop when “11”,
         “X” after Tprop when others;
```

VHDL- Cláusula With

```
signal Y : std_logic_vector(0 to3);
signal opcode : std_logic_vector(0 to 1);
with opcode select
        Y <= “0001” when “00”,
             “0010” when “01”,
             “0100” when “10”,
             “1000” when “11”;
```

VHDL- Exercícios

- Escrever o modelo vhdl de um multiplexador 2x1 com 8 bits de dados

  – usando WHEN
  – usando IF
  – usando somente comando de atribuição simples

```
Signal temp : std_logic_vector(7 downto 0);
Begin
   temp <= (s,s,s, others => s);  -- s é o sinal de seleção
   y <= (temp and IN1) or (not temp and IN0);
```

VHDL- Cláusula Case

```
CASE sinal IS
   WHEN “condição 1” => dado <= dado1;
   WHEN “condição 2” => dado <= dado2;
   WHEN others => dado <= dado3;
END CASE;
```

VHDL- Cláusula Case

```
signal p : integer range 0 to 3;
signal y : std_logic_vector(0 to 1);
CASE p IS
    WHEN 0 => y <= “10”;
    WHEN 1 => y <=  A;
    WHEN 2 => y <=  B
    WHEN 3 => y <= “01”;
  END CASE;
```

VHDL - Comando NULL

- Utilizado no caso de não ser necessário nenhuma ação em uma alternativa que precisa ser coberta

- Exemplo:

```
case opcode is
    when add => Acc := Acc + operando;
    when sub => Acc := Acc - operando;
    when nop => NULL;
end case;
```

VHDL - Exercício

- Utilizado o comando case escreva um conversor do código binário para o código Gray