Linguagem de Descrição de Hardware

SICO5A – Sistemas Digitais

Curso: Engenharia Elétrica

Professor: Layhon Santos
layhonsantos@utfpr.edu.br

Câmpus Apucarana

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS APUCARANA

MINISTÉRIO DA EDUCAÇÃO

BRASIL
GOVERNO FEDERAL

Linguagem VHDL

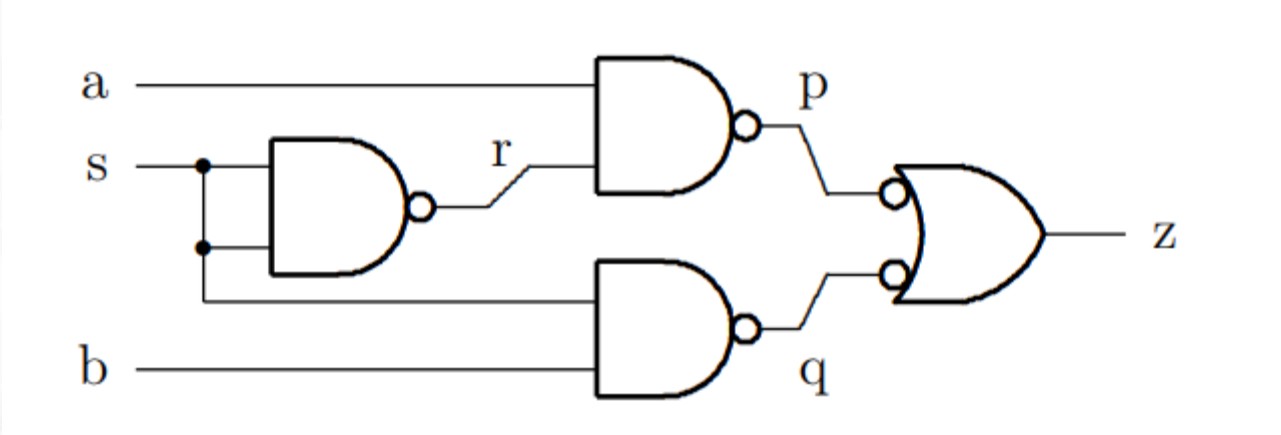
- ✓VHDL é uma sigla que contém uma sigla: ***VHSIC Hardware Description Language.***
- ✓Diferentemente de C, um programa em VHDL descreve uma estrutura, ao invés de definir ou prescrever uma computação.
- ✓um programa em C é uma “receita de bolo” com ingredientes (variáveis) e modo de preparar (algoritmo).
- ✓Em VHDL um modelo estrutural pode descrever a interligação de componentes, ou pode especificar sua função sem mencionar uma implementação daquela função.

Linguagem VHDL

- ✓A partir da estrutura, um compilador como ghdl gera um simulador para o circuito descrito
- ✓VHDL também permite descrever a função ou o comportamento de um circuito, sem que seja necessário definir sua estrutura.
- ✓Um sintetizador é um compilador que gera uma descrição de baixo nível – portas lógicas ou
- ✓transistores – para que um circuito integrado seja fabricado a partir da descrição.

Descrição Gráfica de Circuitos

Quais seriam as convenções empregadas para ser interpretado por alguém de sistemas digitais?



Descrição Gráfica de Circuitos

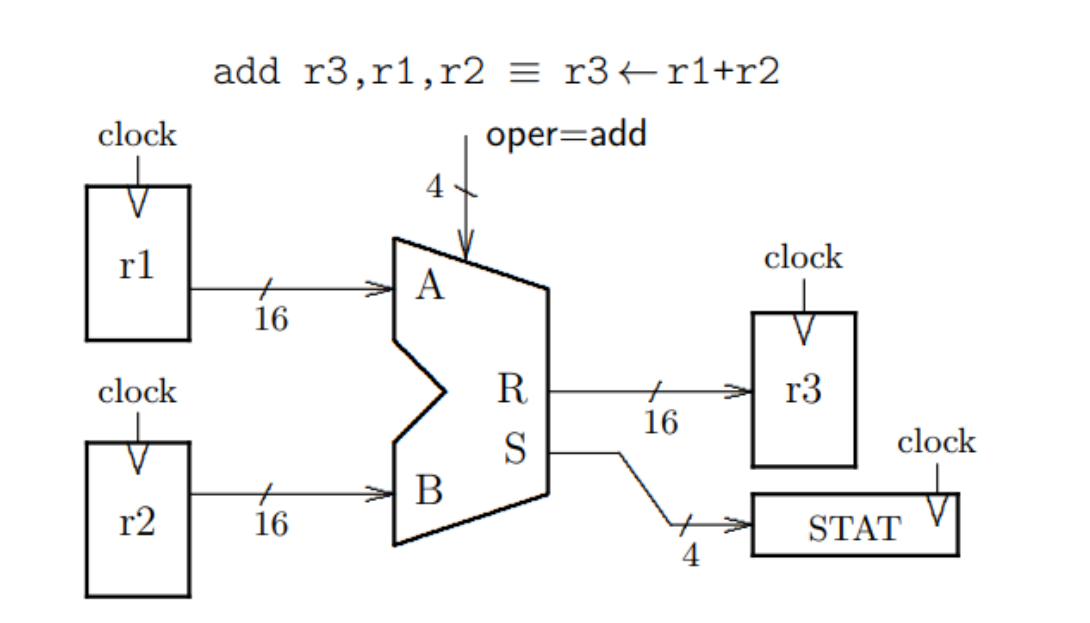
Sua lista deve conter, pelo menos:

- ✓uma convenção para representar as portas lógicas;
- ✓uma convenção para representar as ligações;
- ✓uma convenção para representar as entradas;
- ✓uma convenção para representar as saídas; e
- ✓uma convenção para nomear os valores lógicos transportados pelas ligações.

Descrição Gráfica de Circuitos

Exemplo de convenções:

- ✓4 registradores (r1, r2, r3 e STAT)
- ✓Os sinais fluem da esquerda para a direita;
- ✓O sinal de clock, oper e STAT têm um, quatro e 16 bits de largura;
- ✓Apresenta-se um unidade lógica e aritmética;
- ✓Acima do diagrama se tem a linguagem de transferência entre regist.



Descrição Textual

- ✓Para facilitar a interpretação automática de representações para circuitos emprega-se texto.
- ✓Circuitos são representados em VHDL por sinais (fios) e componentes (design units).
- ✓Um componente é descrito por duas construções da linguagem,
  - ✓uma entidade: descreve sua interface com o mundo externo
  - ✓uma arquitetura: descreve seu comportamento através das construções da linguagem.
- ✓tipografia para o código VHDL:
  - ✓palavras reservadas são grafadas em negrito (**entity**),
  - ✓nomes de sinais e componentes são grafados sem serifa (**meuSinal**),
  - ✓comentários iniciam com '––' e terminam no final da linha e são grafados com ––caracteres inclinados .

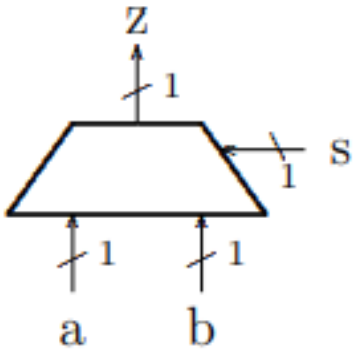
Descrição Textual

- ✓Uma entidade (entity) descreve as ligações externas do componente:
  - ✓entradas são **in**
  - ✓saídas são **out**
  - ✓ligações são representadas por sinais, no nosso caso bits (bit) ou vetores de bits (bit_vectors).
- ✓Uma arquitetura (**architecture**) descreve as ligações internas do componente.
  - ✓A arquitetura declara quais componentes são usados na implementação, que sinais internos são necessários para interligar os componentes internos, e de que forma os componentes são interligados entre si e com os sinais da entidade.
  - ✓Um entidade pode ser modelada de uma ou mais formas.

Entidade e Arquitetura

- ✓O multiplexador abaixo é descrito em VHDL pelas suas entidade e arquitetura.
- ✓entidade, é chamada mux2 e descreve a interface deste componente, com três sinais de entrada e um de saída, todos com a largura de um bit.

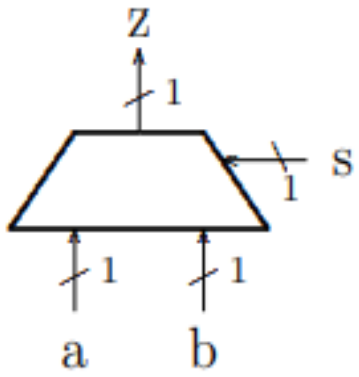
```
entity mux2 is
  port(a,b : in  bit;
       s   : in  bit;
       z   : out bit);
end mux2;
```



Entidade e Arquitetura

- ✓A entidade tem nome mux2.
- ✓A associação entre uma entidade (interface) e sua arquitetura (implementação) se dá no comando de definição da arquitetura

```
entity mux2 is
  port(a,b : in  bit;
       s   : in  bit;
       z   : out bit);
end mux2;
```



Entidade e Arquitetura

- ✓No trecho de código abaixo, a entidade tem nome mux2 e a implementação é chamada de estrutural – que é a versão estrutural da entidade mux2.

```
entity mux2 is
   ...
end mux2;

architecture estrutural of mux2 is
   ...
begin
   ...
end architecture estrutural;
```

Entidade e Arquitetura

- ✓A arquitetura, chamada estrutural , define a estrutura do circuito modelado – ela descreve como os componentes são interligados.
- ✓Entre as palavras chave **architecture** e **begin** devem ser declarados os componentes usados na implementação do multiplexador – neste caso inversor e porta nand – e os sinais internos necessários para ligar os componentes e a interface, que são os sinais r, p e q.

```
architecture estrutural of mux2 is

  -- declaração de componentes
  component inv is
    port(A : in bit; S : out bit);
  end component inv;

  component nand2 is
    port(A,B : in bit; S : out bit);
  end component nand2;

  -- declaração de sinais internos
  signal r, p, q : bit;

begin

  -- instanciação e
  --    interligação dos componentes
  Ui:   inv port map(s, r);

  Ua0: nand2 port map(a, r, p);

  Ua1: nand2 port map(b, s, q);

  Uor: nand2 port map(p, q, z);

end architecture estrutural;
```

Entidade e Arquitetura

- ✓Entre o **begin** e o **end architecture** são definidas as ligações entre os componentes da implementação. Como mostra o diagrama ao lado do código, há uma correspondência direta entre o circuito e o código VHDL que o representa. Infelizmente, nas descrições textuais todos os sinais devem ser explicitamente nomeados.

```
architecture estrutural of mux2 is

  -- declaração de componentes
  component inv is
    port(A : in bit; S : out bit);
  end component inv;

  component nand2 is
    port(A,B : in bit; S : out bit);
  end component nand2;

  -- declaração de sinais internos
  signal r, p, q : bit;

begin

  -- instanciação e
  --    interligação dos componentes
  Ui:   inv port map(s, r);

  Ua0: nand2 port map(a, r, p);

  Ua1: nand2 port map(b, s, q);

  Uor: nand2 port map(p, q, z);

end architecture estrutural;
```

Entidade e Arquitetura

- ✓A construção **port map** faz a ligação entre os sinais internos à arquitetura e os sinais declarados para cada componente individual usado no modelo. O **port map** é quem liga sinais em componentes ao mapear os sinais internos declarados na arquitetura aos sinais declarados nas entidades dos componentes.
- ✓A instanciação dos componentes com o mapeamento das portas é similar a um operador aritmético prefixado; ao invés da forma usual, que é infixada (c = a+b), emprega-se a forma prefixada (+(c, a, b))

```
architecture estrutural of mux2 is

  -- declaração de componentes
  component inv is
    port(A : in bit; S : out bit);
  end component inv;

  component nand2 is
    port(A,B : in bit; S : out bit);
  end component nand2;

  -- declaração de sinais internos
  signal r, p, q : bit;

begin

  -- instanciação e
  --    interligação dos componentes
  Ui:   inv port map(s, r);

  Ua0: nand2 port map(a, r, p);

  Ua1: nand2 port map(b, s, q);

  Uor: nand2 port map(p, q, z);

end architecture estrutural;
```

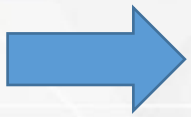
Entidade e Arquitetura

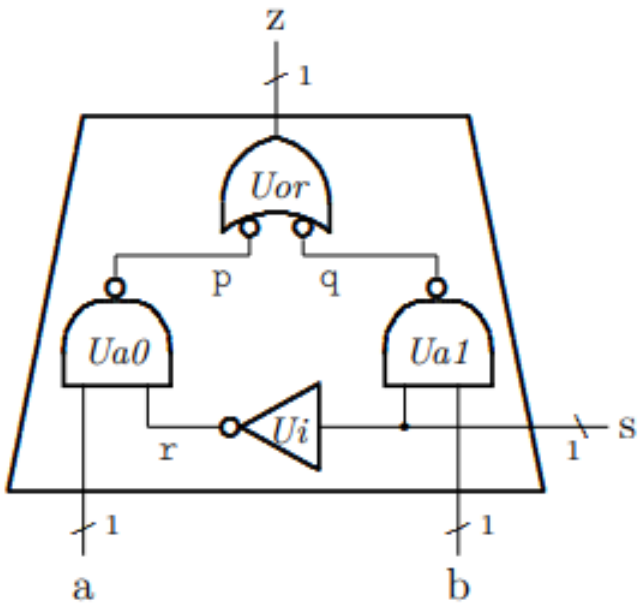
- ✓Entre **architecture** e **begin**, os componentes e os sinais são declarados. Entre o **begin** e o **end architecture**, os sinais e componentes são instanciados e conectados através dos sinais internos e do mapeamento dos sinais nas portas dos entidades com o **port map**. O **port map** associa, ou 'gruda', os sinais nos terminais do componente.

```
architecture estrutural of mux2 is

  -- declaração de componentes
  component inv is
    port(A : in bit; S : out bit);
  end component inv;

  component nand2 is
    port(A,B : in bit; S : out bit);
  end component nand2;

  -- declaração de sinais internos
  signal r, p, q : bit;

begin

  -- instanciação e
  --    interligação dos componentes
  Ui:   inv port map(s, r);

  Ua0: nand2 port map(a, r, p);

  Ua1: nand2 port map(b, s, q);

  Uor: nand2 port map(p, q, z);

end architecture estrutural;
```

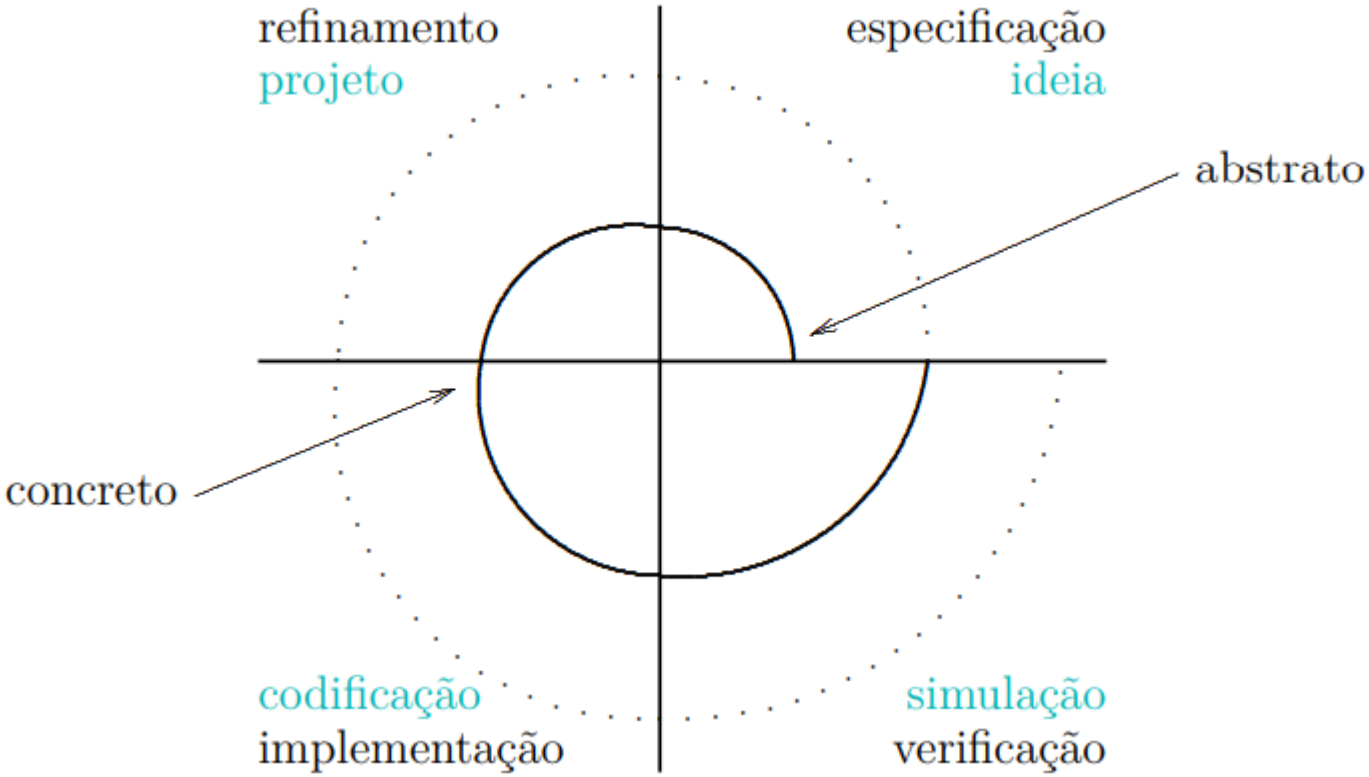
Entidade e Arquitetura

```
architecture estrutural of mux2 is

  -- declaração de componentes
  component inv is
    port(A : in bit; S : out bit);
  end component inv;

  component nand2 is
    port(A,B : in bit; S : out bit);
  end component nand2;

  -- declaração de sinais internos
  signal r, p, q : bit;

begin

  -- instanciação e
  --    interligação dos componentes
  Ui:   inv port map(s, r);

  Ua0: nand2 port map(a, r, p);

  Ua1: nand2 port map(b, s, q);

  Uor: nand2 port map(p, q, z);

end architecture estrutural;
```





Espiral Projeto em VHDL



Modelagem Circuitos Combinacionais em VHDL

- ✓VHDL é uma linguagem extremamente versátil e é usada para:
  - ✓a modelagem e simulação de circuitos;
  - ✓especificação, projeto e síntese de circuitos;
  - ✓descrever listas de componentes e de ligações;
  - ✓e verificação de corretude: se especificação ⇒ implementação.
- ✓A linguagem é padronizada pelo IEEE, no padrão IEEE 1076-1987 V[HSIC] H D L, ou Very High Speed Integrated Circuit Hardware Description Language. O padrão IEEE Std 1164 define pacote std_logic, e o padrão IEEE Std 1076.3 VHDL Numeric Standard define uma biblioteca de funções numéricas.

Modelagem Circuitos Combinacionais em VHDL

✓**Tipos de Dados:**

TIPO	VALOR	EXEMPLO DE USO
bit	'1', '0'	D <= '1';
bit_vector	vetor de bits	Dout <= "0011";
boolean	TRUE, FALSE	start <= TRUE;
integer	-2,-1,0,1,1	Cnt <= Cnt + 17;
real	2.3, 2.3E-4	med <= sum/16.0;
time	100 ps, 3 ns	sig <= '1' **after** 12 ns;
character	'a','1','@'	c <= 'k';
string	vetor de **char**	msg <= "error: " & "badAddr";

✓Atenção porque isso é importante: VHDL **não** é case sensitive – maiúsculas e minúsculas são consideradas iguais.

Modelagem Circuitos Combinacionais em VHDL

- ✓**Sinais e Vetores de Sinais:**
  - ✓Os sinais em VHDL correspondem aos fios que transportam os bits no circuito e a linguagem define sinais de um bit, e vetores de sinais com muitos bits

```
-- declaração de sinais do tipo bit
signal x: bit := '1'; -- inicializado em '1'
signal vb: bit_vector(3 downto 0) := "1100"; -- inicializa em 12
```

- ✓ O código abaixo declara um sinal x com um bit de largura, e este é inicializado em ’1’. O sinal vb é um vetor com 4 bits de largura e a posição de cada um dos bits é aquela em que o bit mais significativo está à esquerda, e o menos significativo à direita.

```
signal bv : bit_vector(0 to 3); -- bit 0 é o Mais Significativo
```

Modelagem Circuitos Combinacionais em VHDL

✓**Sinais e Vetores de Sinais:**

```
signal x:  bit; -- um bit
signal v8: bit_vector(7 downto 0); -- vetor de 8 bits
```

```
signal v4: bit_vector(3 downto 0); -- vetor de 4 bits
signal t4: bit_vector(3 downto 0); -- vetor de 4 bits
  ...
x  <= v8(7);           -- atribuição do bit mais significativo
v4 <= v8(5 downto 2); -- atribuição do quarteto central de v8
t4 <= v8(2 to 5);      -- mesmo quarteto, na ordem inversa
```

```
  ...
x  <= v8(7);           -- atribuição do bit mais significativo
v4 <= v8(5 downto 2); -- atribuição do quarteto central de v8
t4 <= v8(2 to 5);      -- mesmo quarteto, na ordem inversa
```

Modelagem Circuitos Combinacionais em VHDL

- ✓**Sinais e Vetores de Sinais:**
  - ✓Vetores de bits podem ser representados nas bases hexadecimal (com prefixo x), octal (prefixo o) e binária (prefixo b). A representação binária implícita é a normal: se nada for dito em contrário, o vetor é um vetor de bits

```
hexadecimal  <= x"C0";        -- 1100 0000    OITO bits
octal        <= o"300";       -- 011 000 000  NOVE bits
binaria_expl <= b"11000000"; -- 11000000 base-2 explícita
binaria_impl <= "11000000";  -- 11000000 base-2 implícita
```

Modelagem Circuitos Combinacionais em VHDL

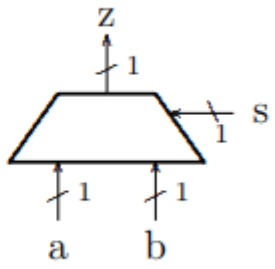
- ✓ **Abstração em VHDL: Packages**
  - ✓ o código VHDL que define uma série de nomes para sinais com mais de um bit. Este conjunto de definições é chamado de pacote, ou package e pode ser armazenado em um arquivo que é compilado separadamente dos demais arquivos com código fonte.
  - ✓ Uma vez que o pacote seja compilado, suas definições podem ser usadas em todas as demais unidades de projeto. Pacotes têm função similar aos arquivos .h da linguagem C.

```
package p_WIRES is -- abreviaturas para
                   -- novos tipos e agrupamentos de sinais

  subtype natur8  is integer range 0 to 255;   -- natural de 8 bits
  subtype natur16 is integer range 0 to 65535; -- natural de 16 bits

  subtype reg2  is bit_vector(1  downto 0); -- par de bits
  subtype reg4  is bit_vector(3  downto 0);
  subtype reg8  is bit_vector(7  downto 0);
  subtype reg15 is bit_vector(14 downto 0);
  subtype reg16 is bit_vector(15 downto 0); -- barramento de 16 bits

package body p_WIRES is
```

Modelagem Circuitos Combinacionais em VHDL

- ✓ **Modelo estrutural do multiplexador**
  - ✓ Um modelo estrutural descreve a estrutura de um circuito, do ponto de vista das interconexões entre seus componentes.
  - ✓ A lista de ligações e componentes é chamada de net list.
  - ✓ A entidade do multiplexador de duas entradas é mostrada abaixo

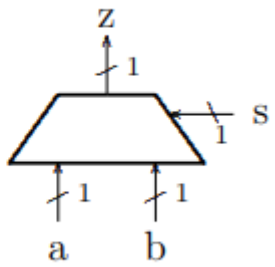
```
use work.p_wires.all;
entity mux2 is
  port(a,b : in  bit;
       s   : in  bit;
       z   : out bit);
end mux2;
```



Modelagem Circuitos Combinacionais em VHDL

- ✓**Modelo estrutural do multiplexador**
  - ✓Um modelo estrutural descreve a estrutura de um circuito, do ponto de vista das interconexões entre seus componentes.
  - ✓A lista de ligações e componentes é chamada de net list.
  - ✓A entidade do multiplexador de duas entradas é mostrada abaixo

```
use work.p_wires.all;
entity mux2 is
  port(a,b : in  bit;
       s   : in  bit;
       z   : out bit);
end mux2;
```



  - ✓diagrama convencional da interligação dos componentes do mux-2 é mostrado ao lado da arquitetura

Modelagem Circuitos Combinacionais em VHDL

- ✓**Modelo estrutural do multiplexador**
  - ✓os componentes e os sinais internos são declarados entre o **architecture** e o **begin**. Os componentes são instanciados entre o **begin** e o **end architecture**
  - ✓A região entre o begin e o end architecture é chamada de área concorrente e os comandos nesta área são executados concorrentemente.

```
architecture estrutural of mux2 is

  -- declaração dos componentes
  component inv is
    port(A: in bit; S: out bit);
  end component inv;

  component nand2 is
    port(A,B: in bit; S: out bit);
  end component nand2;

  -- declaração sinais internos
  signal r, p, q: bit;

begin
  -- início da área concorrente

  -- instanciação dos componentes
  Ui:  inv   port map (s, r);
  Ua0: nand2 port map (a, r, p);
  Ua1: nand2 port map (b, s, q);
  Uor: nand2 port map (p, q, z);

  -- fim da área concorrente
end architecture estrutural;
```

Modelagem Circuitos Combinacionais em VHDL

- ✓**Modelo estrutural do multiplexador**
  - ✓VHDL é dita declarativa: as quatro instâncias dos componentes (Ui, Ua0, Ua1 e Uor) são declaradas numa ordem arbitrária no código fonte enquanto que a ocorrência de eventos nas saídas dos componentes depende somente da sequência de eventos nos sinais que os interligam.

```
architecture estrutural of mux2 is

  -- declaração dos componentes
  component inv is
    port(A: in bit; S: out bit);
  end component inv;

  component nand2 is
    port(A,B: in bit; S: out bit);
  end component nand2;

  -- declaração sinais internos
  signal r, p, q: bit;

begin
  -- início da área concorrente

  -- instanciação dos componentes
  Ui:  inv   port map (s, r);
  Ua0: nand2 port map (a, r, p);
  Ua1: nand2 port map (b, s, q);
  Uor: nand2 port map (p, q, z);

  -- fim da área concorrente
end architecture estrutural;
```

Modelagem Circuitos Combinacionais em VHDL

- ✓ **Modelo estrutural do multiplexador**
  - ✓ um exemplo de associação nominal na instanciação de um componente com uma entrada e uma saída. No exemplo, a ordem dos sinais, de entrada e de saída, está invertida na instanciação da unidExterna. O operador associação nomail é uma flexa invertida.

```
architecture exemplo of assocNominal

  component unidExterna is
    port(INPext : in bit; OUText : out bit);
  end component unidExterna;

  signal ENTRlocal, SAIDAlocal : bit;

begin
  ...
  U: unidExterna port map (SAIDAlocal => OUText, ENTRlocal => INPext);
  ...
end architecture exemplo;
```

Modelagem Circuitos Combinacionais em VHDL

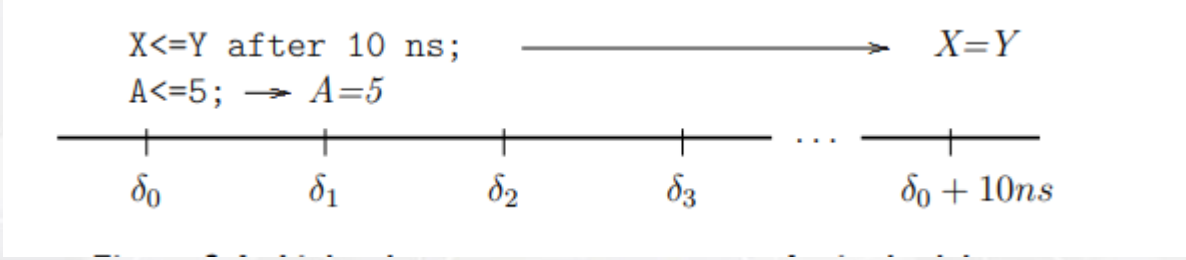
- ✓ **Processo de compilação**
  - ✓ **Análise**:
    - ✓ sintática, são sinalizados erros de grafia das palavras reservadas, sinais que não foram declarados, etc.
    - ✓ semântica, são verificados os tipos dos sinais, e se os operadores são aplicados a operandos com os tipos apropriados – somar dois números faz sentido, somar dois ifs não.
  - ✓ **Elaboração:** na fase de elaboração, o compilador constrói um modelo completo de toda.
  - ✓ **Simulação:** o compilador ghdl traduz a estrutura de processos e os sinais que os interligam para um programa em C que, depois de compilado, permite simular a execução do modelo completo.

Modelagem Circuitos Combinacionais em VHDL

- ✓**Mecanismo de simulação:**
  - ✓A simulação de modelos escritos em VHDL emprega a Simulação de Eventos Discretos – o tempo simulado progride em função de eventos que ocorrem em instantes determinados pela própria simulação.

```
A <= 5;                  -- atribuição válida no próximo delta
X <= Y after 10 ns;      -- atribuição válida depois de 10ns
```

  - ✓A atribuição a um sinal causa uma transação, e a atribuição será efetivada no próximo delta**.**



Modelagem Circuitos Combinacionais em VHDL

- ✓**Testbenches:**
  - ✓Tipicamente, um arquivo com um testbench (TB, ou programa de teste)) contém código para verificar a corretude de modelos. Para simplificar a depuração do seu código VHDL, é uma boa ideia verificar cada novo modelo assim que seu código for completado. Um componente com N entradas tem uma tabela verdade com 2 N linhas; quanto menor for N, mais simples é o conjunto de testes necessário para garantir a corretude do modelo.

```
use work.p_wires.all;
entity tb_estrut is
  -- entidade vazia, não se comunica com "mundo externo"
end tb_estrut;
```

Modelagem Circuitos Combinacionais em VHDL

- ✓**Testbenches:**
  - ✓Tipicamente, um arquivo com um testbench (TB, ou programa de teste)) contém código para verificar a corretude de modelos. Para simplificar a depuração do seu código VHDL, é uma boa ideia verificar cada novo modelo assim que seu código for completado. Um componente com N entradas tem uma tabela verdade com 2 N linhas; quanto menor for N, mais simples é o conjunto de testes necessário para garantir a corretude do modelo.

```
architecture TB of tb_estrut is

  -- declaração dos componentes por testar
  component mux2 is
    port(A,B : in  bit; S : in  bit; Z : out bit);
  end component mux2;
```

Modelagem Circuitos Combinacionais em VHDL

- ✓**Testbenches:**
  - ✓**Vetor de testes:** é um conjunto bem definido de valores, que quando apresentados às entradas do modelo, produzem um conjunto bem definido de saídas.

```
type test_record_mux is record
  s    : bit;       -- entrada de seleção
  a,b  : bit;       -- entradas
  z    : bit;       -- saída esperada
end record;

type test_array_mux is array(positive range <>) of test_record_mux;

constant test_vectors_mux : test_array_mux := (
  --s,  a,  b,  z
   ('0','0','0','0'), -- linhas da tabela verdade
   ('0','0','1','0'),
   ('0','1','0','1'),
   ('0','1','1','1'),
   ('1','0','0','0'),
   ('1','0','1','1'),
   ('1','1','0','0'),
   ('1','1','1','1') );
```

Modelagem Circuitos Combinacionais em VHDL

- ✓**Testbenches:**
  - ✓**Vetor de testes:** é um conjunto bem definido de valores, que quando apresentados às entradas do modelo, produzem um conjunto bem definido de saídas.

```
type test_record_mux is record
  s    : bit;       -- entrada de seleção
  a,b  : bit;       -- entradas
  z    : bit;       -- saída esperada
end record;

type test_array_mux is array(positive range <>) of test_record_mux;

constant test_vectors_mux : test_array_mux := (
  --s,  a,  b,  z
  ('0','0','0','0'), -- linhas da tabela verdade
  ('0','0','1','0'),
  ('0','1','0','1'),
  ('0','1','1','1'),
  ('1','0','0','0'),
  ('1','0','1','1'),
  ('1','1','0','0'),
  ('1','1','1','1') );
```

Modelagem Circuitos Combinacionais em VHDL

- ✓ **Testbenches**
  - ✓ **Laço de Testes**
    - ✓ a variável de indução i itera no espaço definido pelo número de elementos do vetor de testes (test_vectors 'range) – o atributo 'range representa a faixa de valores do índice do vetor. Se mais elementos forem acrescentados ao vetor, o laço executará mais iterações.
    - ✓ O i-ésimo elemento do vetor é atribuído à variável v e todos os campos do vetor são então atribuídos aos sinais que excitam o modelo.

```
architecture TB of tb_estrut is
  -- declaração do componente por testar
  component mux2 is
    port(A,B : in  bit; S : in  bit; Z : out bit);
  end component mux2;

  -- sinais do testbench
  signal sel, ea, eb, saida, esperada : bit;
begin

  U_mux2: mux2 port map(ea,eb,sel,saida); -- componente por testar

  U_testValues: process          -- este componente efetua os testes
      variable v : test_record_mux;
    begin

      for i in test_vectors_mux'range loop
        v        := test_vectors_mux(i);  -- valores de teste
        sel      <= v.s;      -- bit de seleção
        ea       <= v.a;      -- entrada a
        eb       <= v.b;      -- entrada b
        esperada <= v.z;      -- saída cfe tabela verdade

        wait for 200 ps;      -- duração do passo de simulação

        assert (saida = esperada)
          report LF & "mux2: saida errada sel="& B2STR(sel) &
            " saiu=" & B2STR(saida) & " esperada=" & B2STR(esperada)
          severity error;
      end loop;

    wait; -- encerra a simulação

  end process U_testValues;

end architecture TB;
```

Modelagem Circuitos Combinacionais em VHDL

✓**Testbenches**

Assert

✓O comando assert é similar a um printf() em C e pode ser usado para exibir o valor de sinais ao longo de uma simulação. Este comando tem três cláusulas, a saber:

- ✓ assert condição – a condição deve ser falsa para que o simulador imprima a string;
- ✓report string – texto informativo por imprimir;
- ✓severity nível – a severidade pode ser um de quatro níveis, note, warning, error, failure , e esta última aborta a simulação.

Modelagem Circuitos Combinacionais em VHDL

✓Testbenches

Assert

✓Abaixo, o assert verifica se a saída observada no multiplexador é igual à saída esperada. Se os valores forem iguais, o comportamento é o esperado, e portanto correto com relação aos vetores de teste que o projetista escreveu. Se o projetista escolher valores de teste inadequados, ou errados, pode ser difícil diagnosticar problemas no modelo.

```
assert (saida = esperada)
  report "mux2: saida errada sel="& B2STR(sel) &
  " saiu=" & B2STR(saida) & " esperada=" & B2STR(esperada)
  severity error;
```

✓Se os valores nos sinais saída e esperada diferem, a mensagem do Programa anterior é emitida no terminal, indicando o erro:

```
mux2: saída errada   sel=1   saiu=0   esperada=1.
```

✓A função B2STR converte um bit em uma string para que o valor do bit seja emitido na tela. O operador '&' concatena duas strings. É obrigação do projetista determinar a saída esperada para cada combinação de entradas no vetor de teste.

Modelagem Circuitos Combinacionais em VHDL

- **Uma entidade e muitas arquiteturas**
  - VHDL provê várias construções que facilitam o emprego de abstração. Vimos, como definições podem ser agrupadas em packages e estas importadas pelas unidades de projeto que delas fazem uso. Outra construção útil é a possibilidade de definir mais de uma arquitetura para uma entidade.
  - Lembre da espiral de projeto: a cada avanço na espiral, a realização da especificação se torna mais concreta e mais próxima da implementação final.
  - Uma entidade define a interface de um componente com o mundo exterior àquele componente. Uma arquitetura é um refinamento, e uma das possíveis implementações, para a especificação daquele componente.

Modelagem Circuitos Combinacionais em VHDL

```
entity meuProjeto is
  port( ... )
end meuProjeto;

architecture especificacao of meuProjeto is
begin
   ...
end architecture especificacao;

architecture concreta of meuProjeto is
begin
   ...
end architecture concreta;
```

- ✓**Uma entidade e muitas arquiteturas**
  - ✓A entidade meuProjeto define a interface do componente. A arquitetura especificacao define a função e o comportamento desejado de meuProjeto. A arquitetura concreta é uma versão detalhada e que poderia ser sintetizada diretamente pelo compilador VHDL.