Universidade Tecnológica Federal do Paraná – Toledo
Engenharia da Computação – COENC

Lógica Reconfigurável

Classes de Objetos, tipos de dados e atributos - “Aula de consulta rápida” –

Tiago Piovesan Vendruscolo

Nota sobre a aula

Essa aula está definida como “Aula de consulta rápida” devido ao fato de ter uma compilação de tabelas com bibliotecas, tipos de atributos, definições, etc. Então sempre que surgir alguma dúvida sobre isso nas próximas aulas, consultem essa aula.

Classe de Objetos

O que são: Elementos que contêm um valor armazenado;

- CONSTANT: objeto com valor estático;
- VARIABLE: Usado em regiões de código sequencial. Pode ser alterado no decorrer do programa. Não existe fisicamente.
- SIGNAL: Usando em regiões de código concorrente e sequencial. Está relacionado a conexões físicas internas. Pode ser alterado no decorrer do programa.
- FILE: Usado na criação de arquivos.

Classe de Objetos

- CONSTANT: Objeto com valor estático;

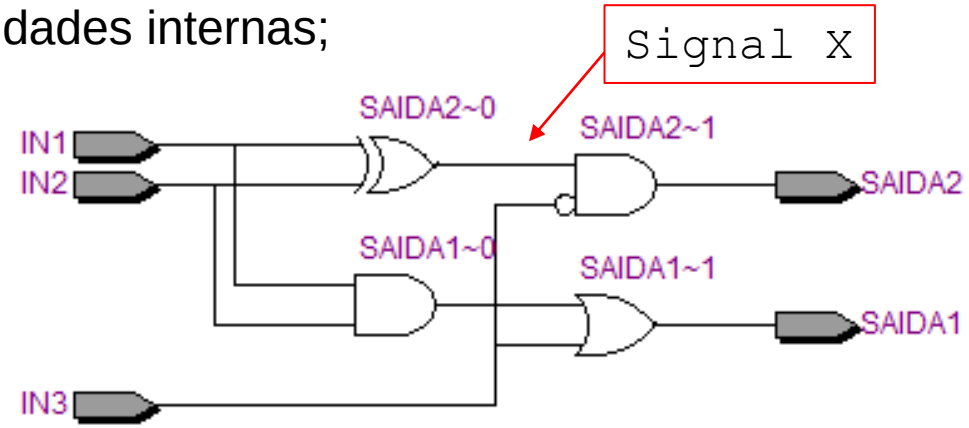
```
CONSTANT contant_name: constant_type := constant_value;
```

- NOME: pode ser qualquer palavra, exceto as reservadas;
- TIPO: pode ser qualquer tipo predefinido no VHDL, ou definido pelo usuário;
- VALOR: uma constante, ou uma expressão envolvendo constantes.

```
CONSTANT test: INTEGER := 16;
CONSTANT words: INTEGER := 2**bits;
CONSTANT flag: BIT := '1';
CONSTANT mask: BIT_VECTOR(1 TO 8) := "00001111";
```

Classe de Objetos

- SIGNAL: Serve para passar valores de entrada e saída do circuito, assim como entre suas unidades internas;




```
SIGNAL signal_name: signal_type [range] [:= default_value;]
```

- Todas as PORTS de uma ENTITY são sinais por default;

```
SIGNAL enable: BIT := '1';

SIGNAL temp: BIT_VECTOR(3 DOWNTO 0);

SIGNAL byte: STD_LOGIC_VECTOR(0 TO 7);

SIGNAL count: NATURAL RANGE 0 TO 255;
```

NOTA:
<= para atribuir no corpo do código.
:= para atribuir na declaração.

Classe de Objetos

- SIGNAL: Exemplo de código.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY signal1 IS
   PORT (A, B : IN std_logic;
         SAIDA : OUT std_logic);
END signal1;

ARCHITECTURE logica OF signal1 IS
SIGNAL teste : std_logic;
BEGIN
   teste<= A AND B;
   SAIDA<= teste;
END logica;
```

Nota: Em códigos sequenciais a atualização não é imediata.

Classe de Objetos

- VARIABLE : Representam informação locais, pois ela pode ser vista apenas dentro de unidades sequenciais (PROCESS ou subprogramas) onde a mesma foi criada;

```
VARIABLE variable_name: variable_type[range] [:=default_value;]
```

```
VARIABLE liga: STD_LOGIC := '1';
VARIABLE vetor: BIT_VECTOR(3 DOWNTO 0);
VARIABLE byte: STD_LOGIC_VECTOR(0 TO 7);
VARIABLE count: INTEGER RANGE 0 TO 255;
```

NOTA:
:= para atribuir no corpo do código.
:= para atribuir na declaração.

Nota: Em códigos sequenciais, a atualização é imediata (ao contrário de signal)

Classe de Objetos

- VARIABLE vs SIGNAL: Exemplo de código.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY cod_sequencial IS
   PORT (a, b: IN STD_LOGIC;
         c, d: OUT STD_LOGIC);
END cod_sequencial;

ARCHITECTURE exemplo OF cod_sequencial IS
   signal temp : STD_LOGIC:='1'; -- "variável" global; inicialização opcional
BEGIN
   PROCESS (a,b)
      variable x: STD_LOGIC:='1'; -- variável local; inicialização opcional
   begin
      if (a'event and a='1') then
         x:= a AND B;
      end if;
      d <= x;
      temp <= x;
   END PROCESS;
   c <= temp; -- OK
   c <= x;
   c <= d;
END exemplo;
```

```
ENTITY cod_sequencial IS
   PORT (a, b: IN STD_LOGIC;
         c: OUT STD_LOGIC;
         d: BUFFER STD_LOGIC);
END cod_sequencial;
```

10482 VHDL error at cod_sequencial.vhd(22): object "x" is used but not declared

10309 VHDL Interface Declaration error in cod_sequencial.vhd(23): interface object "d" of mode out cannot be read. Change object mode to buffer.

Atribuição

Operadores de Atribuição:

<= Usado com **SIGNAL** e **PORT**.
:= Usado com **VARIABLE**, **CONSTANT**, **GENERIC** e para estabelecer valores iniciais.
=> Usado para atribuir valores individuais a vetores e com a palavra reservada **OTHERS**.

Exemplos:

```
SIGNAL x : STD_LOGIC;
SIGNAL y : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL z : STD_LOGIC_VECTOR(0 TO 7);

x <= '1';
y <= "0000";
z <= "10000000"; -- LSB = '1', others = '0'
z <= (0 =>'1', OTHERS =>'0'); -- z = "10000000"
```

Palavra “OTHERS”

- OTHERS é uma palavra útil para descrever atribuições;
  Representa os demais valores que faltam ser especificados;

The constant below is a = "000000".

```
CONSTANT a: BIT_VECTOR(5 DOWNTO 0) := (OTHERS=>'0');
```

The next constant is b = "01111111" (index 7 gets '0', the others, '1').

```
CONSTANT b: BIT_VECTOR(7 DOWNTO 0) := (7=>'0', OTHERS=>'1');
```

The signal below is c = "01100000" ("|" means "or").

```
SIGNAL c: STD_LOGIC_VECTOR(1 TO 8) := (2|3=>'1', OTHERS=>'0');
```

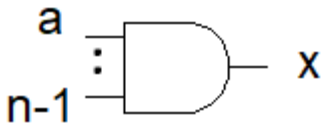
The variable below is d = "1111111100000000".

```
VARIABLE d: BIT_VECTOR(1 TO 16) := (1 TO 8=>'1', OTHERS=>'0');
```

Classe de Objetos

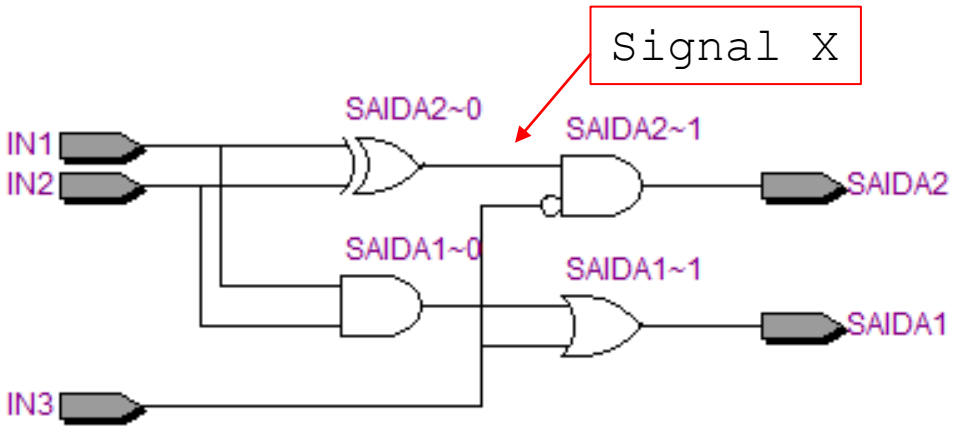
- GENERIC
  - É uma forma de passar uma informação específica para a entidade e arquitetura.
  - Muito útil quando se deseja parametrizar (tornar genérico) um código ou uma parte dele.
  - Exemplo: AND de N entradas:

```
ENTITY n_and_port IS
       GENERIC (N: INTEGER := 4);
       PORT (a: IN BIT_VECTOR (N-1 downto 0);
             x: OUT BIT);
END ENTITY;
```



Exercício 1

- Refaça o exercício abaixo, acrescentando uma “saída3” referente ao signal X.



- Valores para simulação:
- Edit > Set end time : 500 ns.
- Edit > Grid size : 50 ns.
- IN1: Start value: 0; count every: 50 ns;
- IN2: Start value: 1; count every: 100 ns;
- IN3: Start value: 0; count every: 150 ns;

Architecture

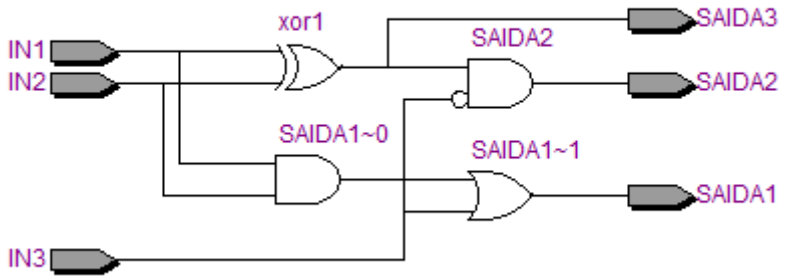
signal...

Begin

Exercício 1

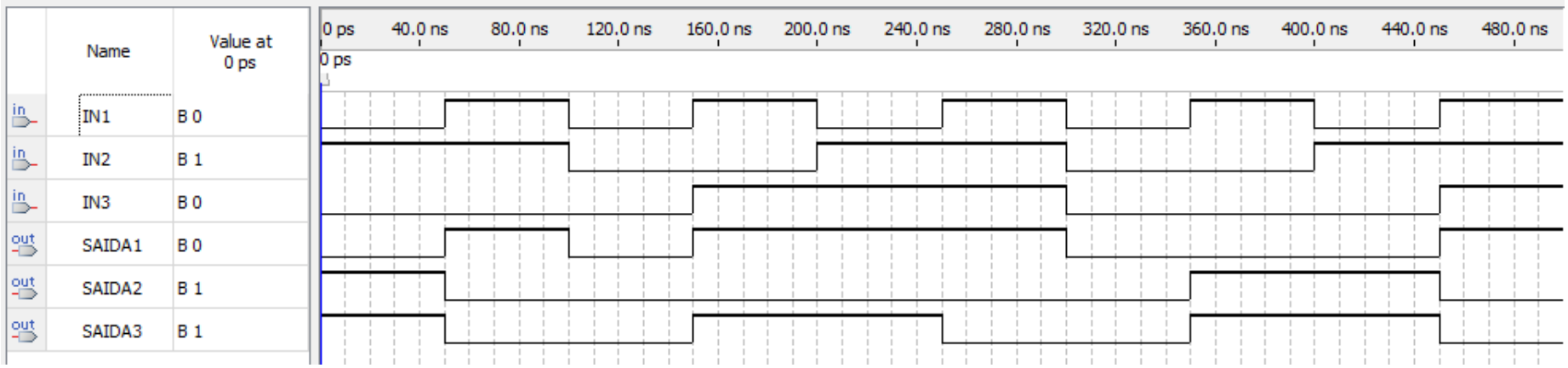
- Resposta:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY signal1 IS
    PORT (IN1, IN2, IN3 : IN std_logic;
          SAIDA1 : OUT std_logic;
          SAIDA2 : OUT std_logic;
          SAIDA3 : OUT std_logic);
END signal1;

ARCHITECTURE logica OF signal1 IS
SIGNAL xor1 : std_logic;
BEGIN
    xor1<= IN1 XOR IN2;
    SAIDA1<= (IN1 AND IN2) OR IN3;
    SAIDA2<= xor1 AND NOT(IN3);
    SAIDA3<= xor1;
END logica;
```



- Nesse caso, podemos utilizar uma variable no lugar do signal?

Não, variable é apenas em códigos sequenciais!



Tipos de dados

- Biblioteca std

Biblioteca	Pacote	Tipo Escalar	Subtipo Escalar	Tipo Composto ARRAY
std	standard	BIT		BIT_VECTOR
		CHARACTER		STRING
		BOOLEAN		
		TIME (Não sintetizável)		
		REAL (Não sintetizável)		
		INTEGER (32 BITS)	NATURAL	
			POSITIVE	

Tipos de dados - escalares

Tipo Predefinido	Valor	Exemplos
BIT	um, zero	1, 0
CHARACTER	Caracteres ASCII	a, b, c, A, B, C, ?, (
BOOLEAN	verdadeiro, falso	TRUE, FALSE
TIME	picossegundos, nanossegundos, microssegundos, etc...	1 ps, 10 ms, 3 ns, 4 fs
INTEGER	-2.147.483.647 a +2.147.483.647	123, 8#173#
NATURAL	0 a +2.147.483.647	123, 8#173#, 16#7B#, 2#11_11_011
POSITIVE	1 a +2.147.483.647	
REAL	$-3{,}65\times10^{47} \le x \le +3{,}65\times10^{47}$	1.23, 1.23E+2, 1,23E-1, 16#7.B#E+1

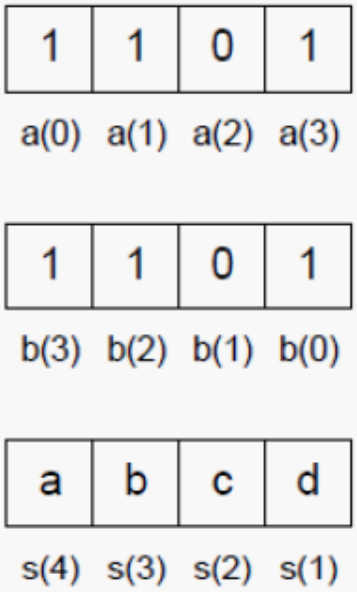
- IMPORTANTE: Para síntese de códigos VHDL, é importante sempre especificar o intervalo para objetos do tipo INTEGER (ou seus subtipos), caso contrário o compilador vai representá-los em 32 bits.
  - Ex. A : IN integer range 0 to 7;

Tipos de dados - compostos

Tipo Predefinido	Valor	Exemplos
BIT_VECTOR	um, zero	“1010”, B”10_10”, O”12”, X”A”
STRING	Cadeia de caracteres ASCII	“texto”, “”incluindo_aspas””

```
Big endian
SIGNAL a: BIT_VECTOR (0 TO 3) := “1101”;

SIGNAL b: BIT_VECTOR (0 DOWNTO 3) := “1101”;

Little endian
SIGNAL c: BIT_VECTOR (3 DOWNTO 0) := “1101”;

SIGNAL s: STRING (4 DOWNTO 1) := “abcd”;
```

X



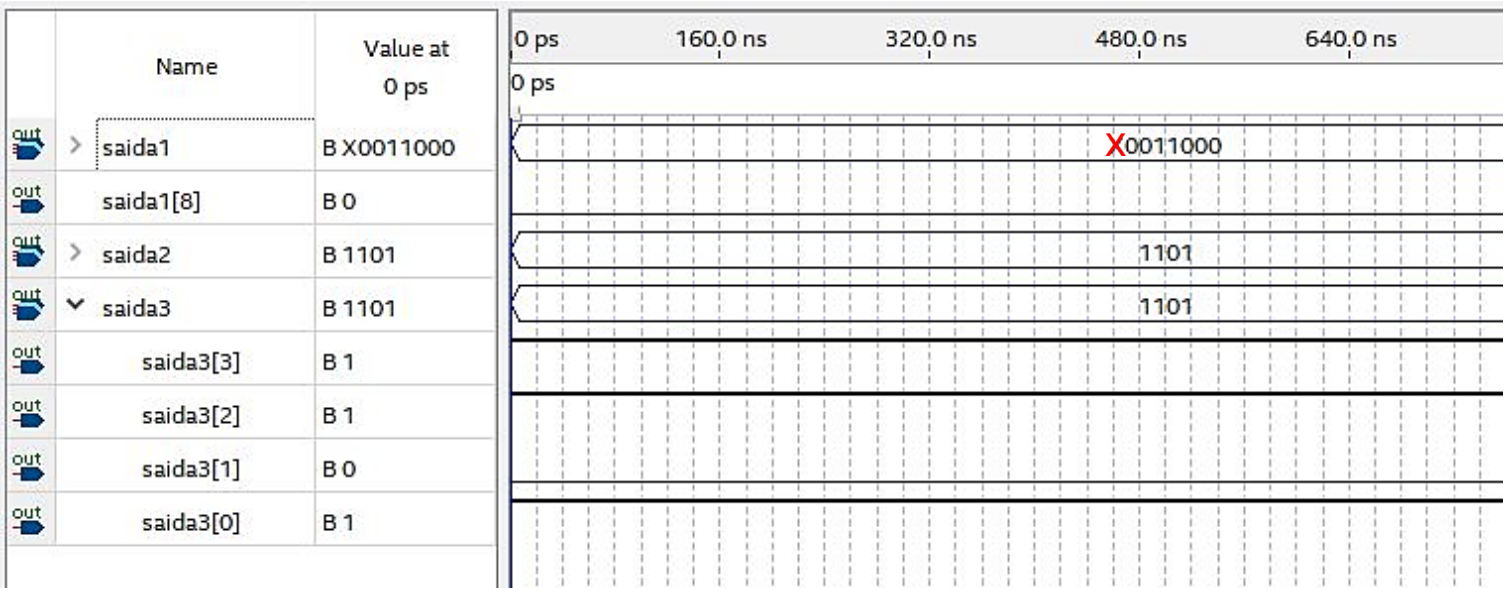
Exercício de fixação

- Crie um projeto, faça o exercício abaixo e simule:

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.all;
3
4   ENTITY aula4 IS
5      PORT (saida1 : OUT std_logic_vector(1 to 8);
6            saida2 : OUT std_logic_vector(0 to 3);
7            saida3 : OUT std_logic_vector(3 downto 0));
8   END aula4;
9
10  ARCHITECTURE logica OF aula4 IS
11     CONSTANT x1: std_logic_vector(1 to 8) := "00110000";
12     CONSTANT x2: std_logic_vector(0 to 3) := "1101";
13     CONSTANT x3: std_logic_vector(3 downto 0) := "1101";
14  BEGIN
15     saida1<= x1;
16     saida2<= x2;
17     saida3<= x3;
18  END logica;
```

Use “others” (slide 15)

Exercício de fixação

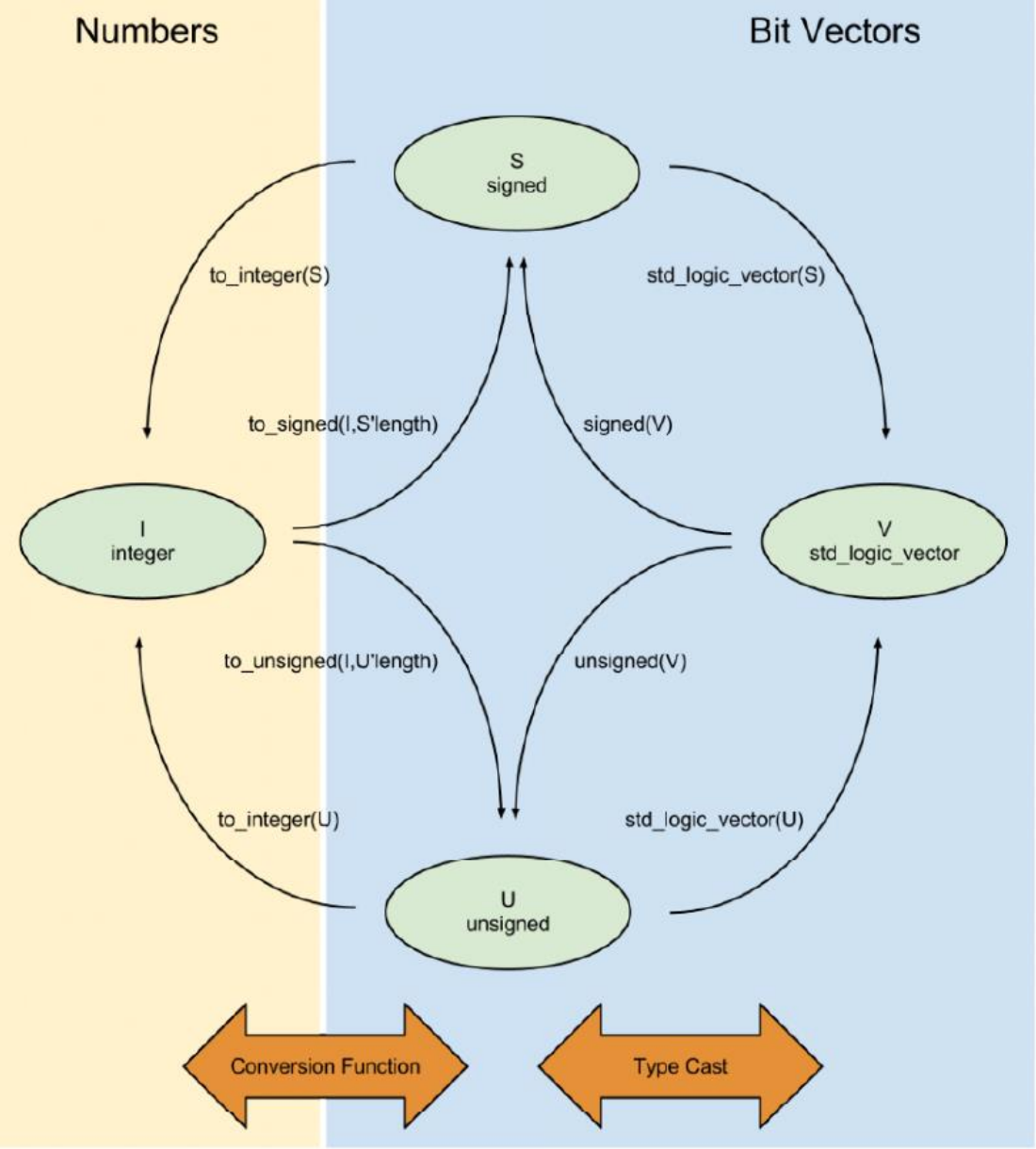


Selecione um grupo > Botão direito > Reverse group or bus bit order

Biblioteca IEEE

- Pacotes:
  - *std_logic_1164*: Especifica os tipos STD_LOGIC e STD_ULOGIC;
  - *std_logic_arith*: Especifica os tipos SIGNED e UNSIGNED e operações aritméticas e de comparações. Também contém várias funções de conversão de dados (CONV_INTEGER(p), CONV_UNSIGNED(p,b), CONV_SIGNED(p,b), CONV_STD_LOGIC_VECTOR(p,b), etc...);
  - *std_logic_signed*: Contém funções que permitem operações com dados STD_LOGIC_VECTOR como se estes fossem do tipo SIGNED;
  - *std_logic_unsigned*: Contém funções que permitem operações com STD_LOGIC_VECTOR como se estes fossem do tipo UNSIGNED.

Tipos de dados



Tipos de dados

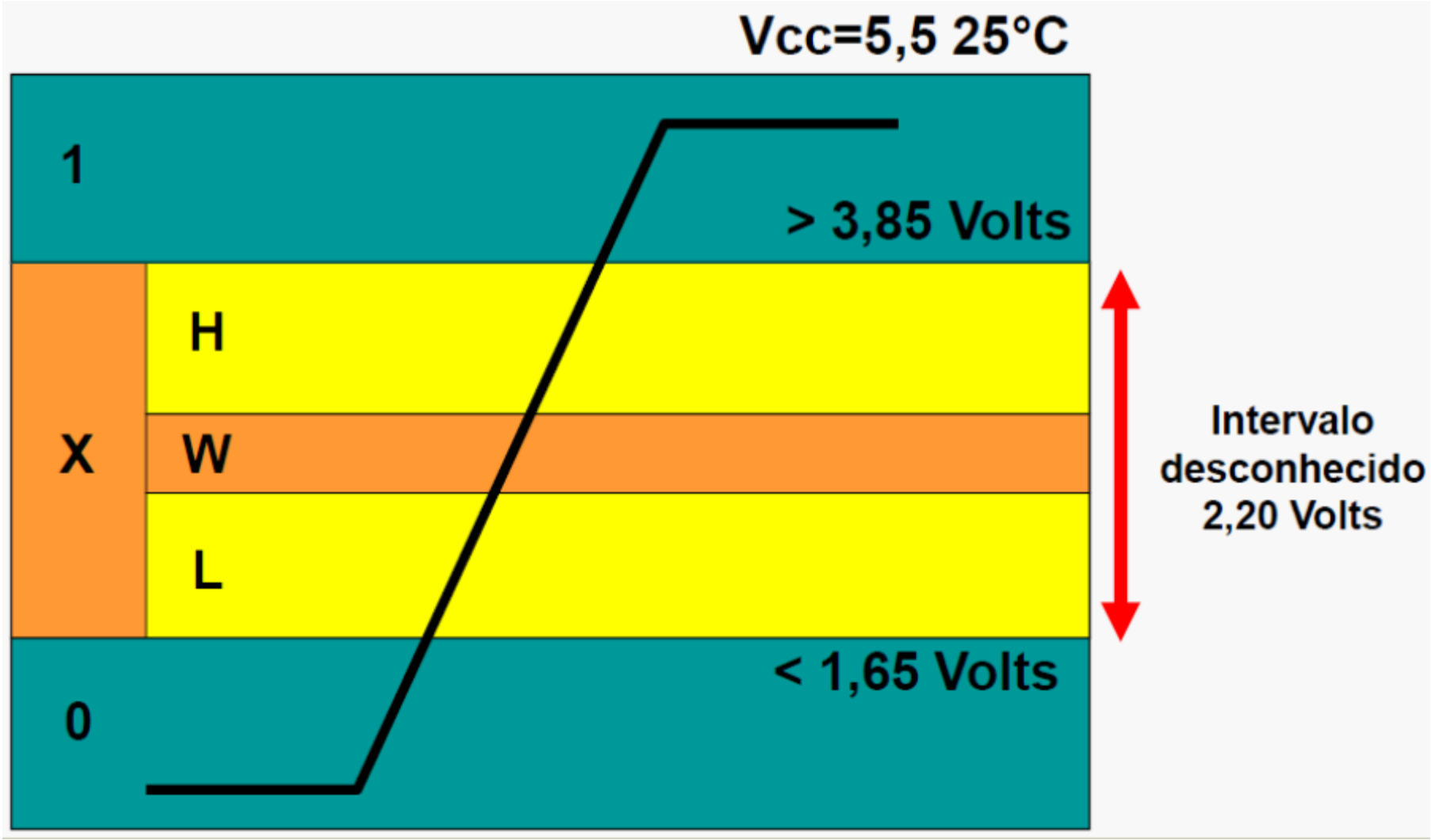
Biblioteca	Pacote	Tipo Escalar	Subtipo Escalar	Tipo Composto ARRAY
ieee	std_logic_1164	STD_ULOGIC	STD_LOGIC	STD_LOGIC_VECTOR STD_ULOGIC_VECTOR
	std_logic_arith	Funções de conversão		
				SIGNED
				UNSIGNED
	std_logic_signed	Funções para operações com STD_LOGIC_VECTOR como se fosse SIGNED		
	std_logic_unsigned	Funções para operações com STD_LOGIC_VECTOR como se fosse UNSIGNED		

- STD_LOGIC_VECTOR e STD_ULOGIC_VECTOR aceitam apenas operações lógicas (AND, OR, XOR, ...) enquanto que SIGNED e UNSIGNED aceitam apenas operações aritméticas.

VALORES DE STD_LOGIC

- ‘Z’ – Alta impedância (Tri-State sintetizável).
- ‘1’ – 1 forçado (‘1’ sintetizável).
- ‘H’ – 1 fraco (coletor ou dreno aberto).
- ‘X’ – Desconhecido forçado (Desconhecido sintetizável).
- ‘W’ – Desconhecido fraco.
- ‘L’ – 0 fraco.
- ‘0’ – 0 forçado (‘0’ sintetizável).
- ‘U’ – Não inicializado.
- ‘–’ – Não importa (don’t care).

STD_LOGIC

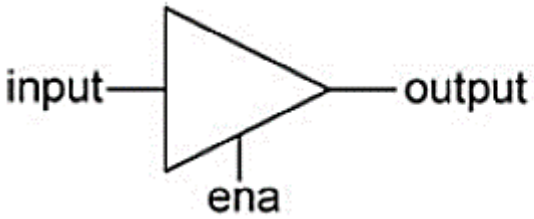

Vcc=5,5 25°C
1
> 3,85 Volts
H
X
W
L
Intervalo
desconhecido
2,20 Volts
< 1,65 Volts
0

Vantagem do STD_LOGIC

- A principal característica do tipo STD_LOGIC é a possibilidade de inclusão de ‘Z’ (alta impedância), e ‘-’ (don’t care), que permite a síntese de buffers tri-state e otimização em hardware de LUTs.

Exemplo – Alta impedância

- Crie um projeto e faça o exercício abaixo:



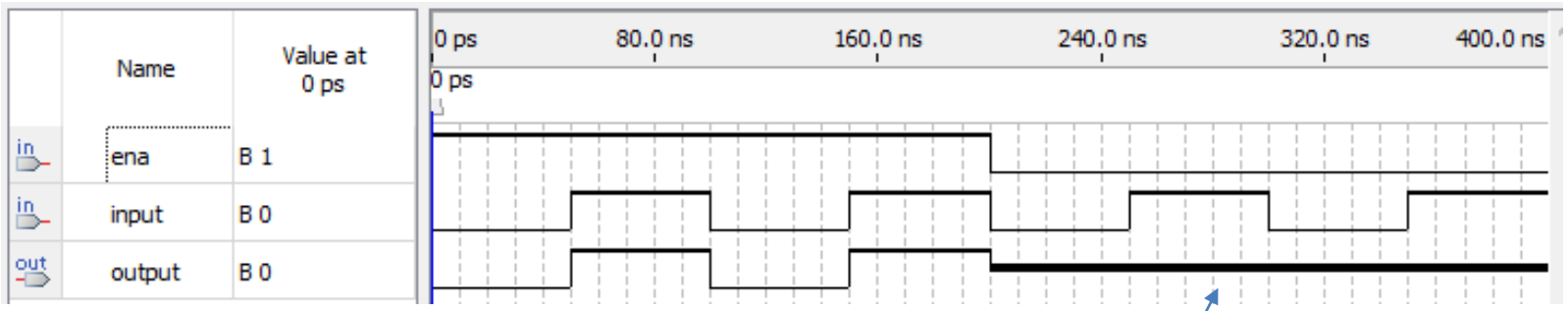
ena	output
'0'	'Z'
'1'	input

```
1  ----------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY tri_state IS
6  PORT (input, ena: IN STD_LOGIC;
7        output: OUT STD_LOGIC);
8  END ENTITY;
9  ----------------------------------------
10 ARCHITECTURE tri_state OF tri_state IS
11 BEGIN
12    output <= input WHEN ena='1' ELSE 'Z';
13 END ARCHITECTURE;
14 ----------------------------------------
```

VHDL não é case sensitive, mas nesse caso em específico, deve-se usar "Z" maiúsculo.

Exemplo – Alta impedância

- Resultado:



A linha grossa significa alta-impedância (Z)

Definição de novos tipos

- O projetista pode definir novos tipos através da palavra reservada TYPE;
- Exemplo:
  - `TYPE estado IS (parado, correndo, andando, pulando);`
  - `SIGNAL abc: estado := pulando;`
- O tipo enumerado permite operações de relação como maior, menor e igual;
- Neste caso:
  - parado < correndo.
  - pulando > andando.

ARRAY

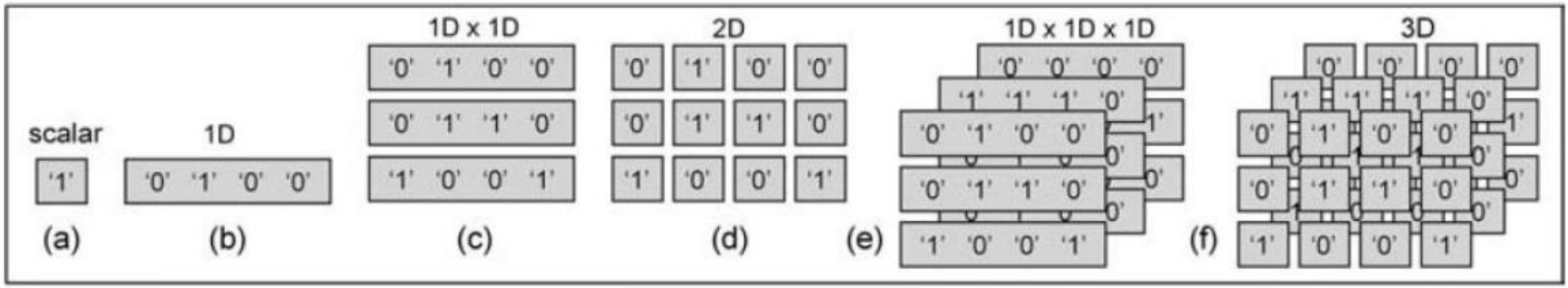


Figure 3.2
Bit-based arrays: (a) Scalar (single bit), (b) 1D (vector of bits), (c) 1D × 1D (pile of vectors), (d) 2D (matrix of bits), (e) 1D × 1D × 1D (block of vectors), and (f) 3D (block of bits).

Exemplo Array 1D com números inteiros naturais (0 a 2147483647):

Número de elementos Tamanho de cada elemento

```
TYPE array1D_count IS ARRAY (NATURAL range 15 DOWNTO 0) of NATURAL RANGE 0 TO 7;
VARIABLE countador  : array1D_count;
```

Exemplo: Array 3D

```
-- declaração do tipo
type array3D_sl is array (natural range <>, natural range <>, natural range <>) of std_logic;
type array3D_bit is array (1 to 2, 3 downto 0, 1 to 5) of bit;

-- declaração do objeto
signal a :   array3D_sl (1 to 2, 3 downto 0, 1 to 5);
signal b :   array3D_bit;

-- atribuição de valor
a <= (("00011", "11101", "11101", "01110"), ("001Z0", "11111", "01001", "00000"));
b <= (("00011", "11101", "11101", "01110"), ("00110", "11111", "01001", "00000"));
```

Linhas, colunas, tamanho do vetor

- Uso de ARRAY 1D x 1D:
  - m(1) <= v(2)(3); --Dois pares de parenteses (x)(y).
- Uso de ARRAY 2D:
  - m(2) <= w(2,1); --Um par de parenteses (x,y).

https://wiki.sj.ifsc.edu.br/index.php/Array_em_VHDL

Atributos predefinidos

- T'LEFT - is the leftmost value of type T. (Largest if downto)
- T'RIGHT - is the rightmost value of type T. (Smallest if downto)
- T'HIGH - is the highest value of type T.
- T'LOW - is the lowest value of type T
- A'RANGE - is the range A'LEFT to A'RIGHT or A'LEFT downto A'RIGHT
- A'RANGE(N) - is the range of dimension N of A.
- A'REVERSE_RANGE - is the range of A with to and downto reversed.
- A'REVERSE_RANGE(N) - is the REVERSE_RANGE of dimension N of array A.
- A'LENGTH -is the integer value of the number of elements in array A.
- A'LENGTH(N) - is the number of elements of dimension N of array A.
- S'EVENT -is true if signal S has had an event this simulation cycle.

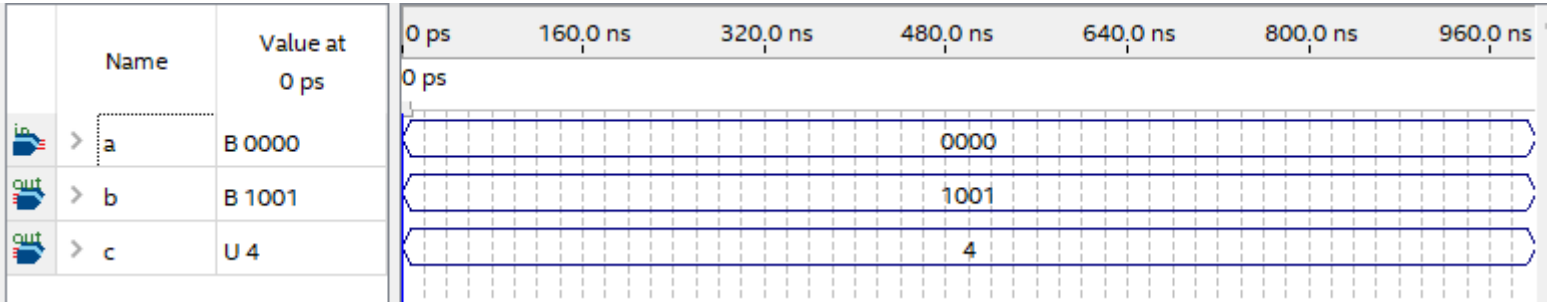
Esses atributos serão bastante utilizados no decorrer da matéria, use essa tabela como referência.

Exercício

- Faça código utilizando atributos (slide anterior) que:
  - Tenha um PORT de entrada “A” parametrizado de 2 a 7 bits;
  - Tenha um PORT de saída “B” com o mesmo tamanho de “A” em que o LSB e o MSB sejam 1 e os outros sejam 0.
  - Tenha um PORT de saída “C” INTEGER com o tamanho de “A”.

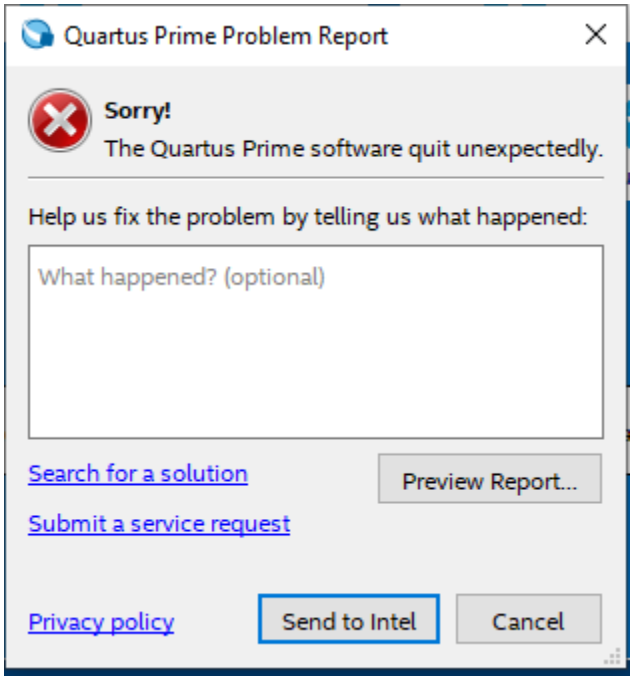
Exercício

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY exercicio IS
   GENERIC (N: INTEGER :=4);
   PORT (a: in std_logic_vector(N-1 downto 0);
         b: out std_logic_vector(N-1 downto 0);
         c: out integer range 0 to 7);
   END ENTITY;

ARCHITECTURE logica OF exercicio IS
   BEGIN
      b<= (b'RIGHT=>'1', b'LEFT=>'1', OTHERS=>'0');
      c<= a'LENGTH;
END;
```



Problemas de simulação

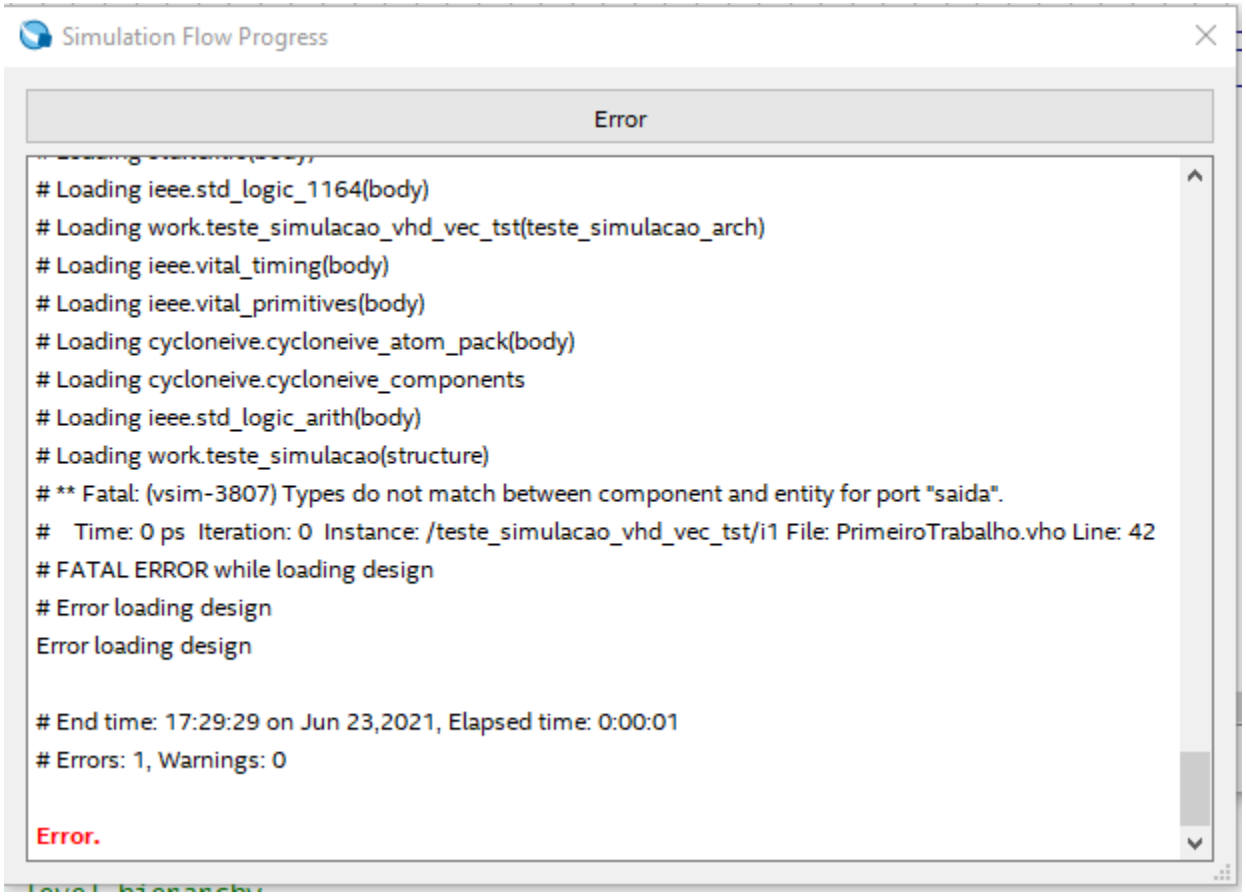
- Não armazene o projeto em serviço de nuvem.



Problemas de simulação

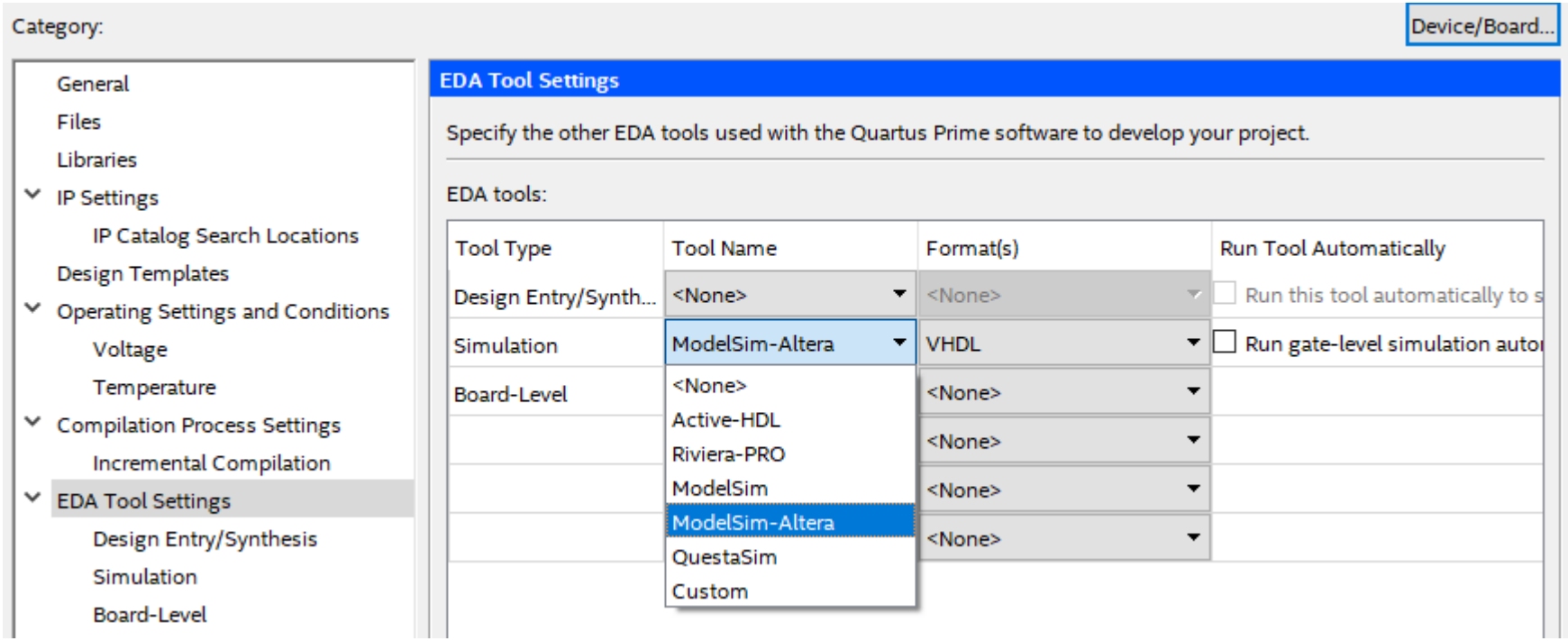
- Erro de conflito entre tipos de dados

```
entity Teste_simulacao is
   port (entrada: in std_logic_vector (2 downto 0);
         saida: out integer range 0 to 7);
end Teste_simulacao;
```



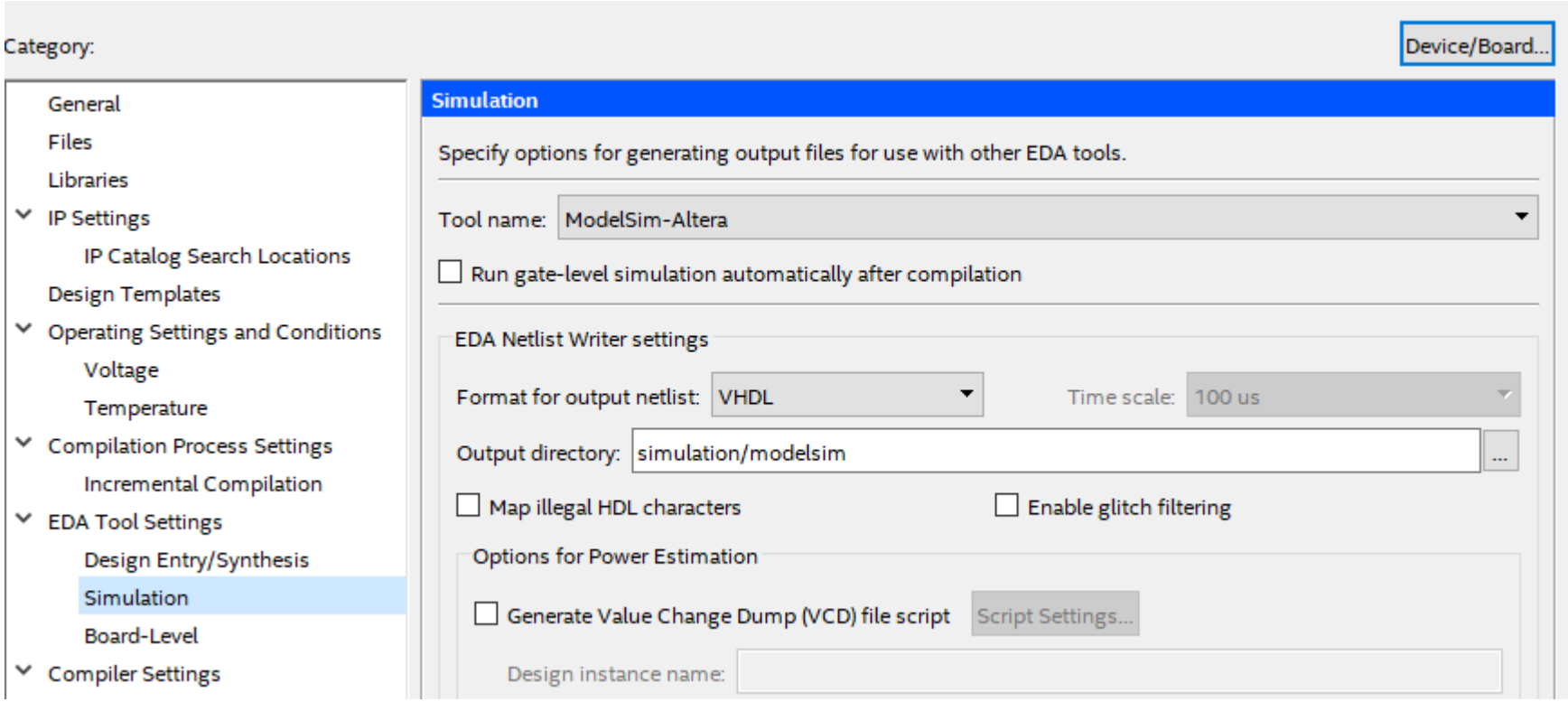
Problemas de simulação

- Para “burlar” o erro da simulação (conflito entre integer e std_logic)
  - *Assignments > settings, altere de ModelSim-Altera para ModelSim (e vice-versa). Format: VHDL. Compile o código novamente e simule o VWF já criado.*



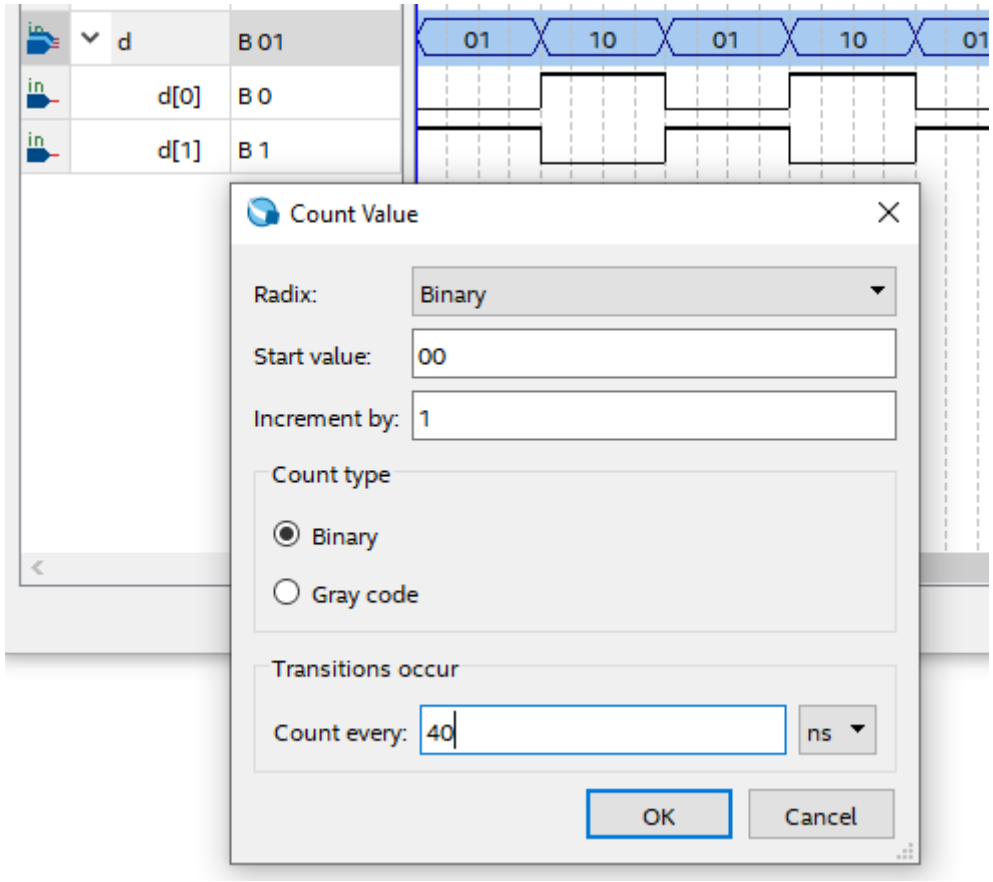
Problemas de simulação

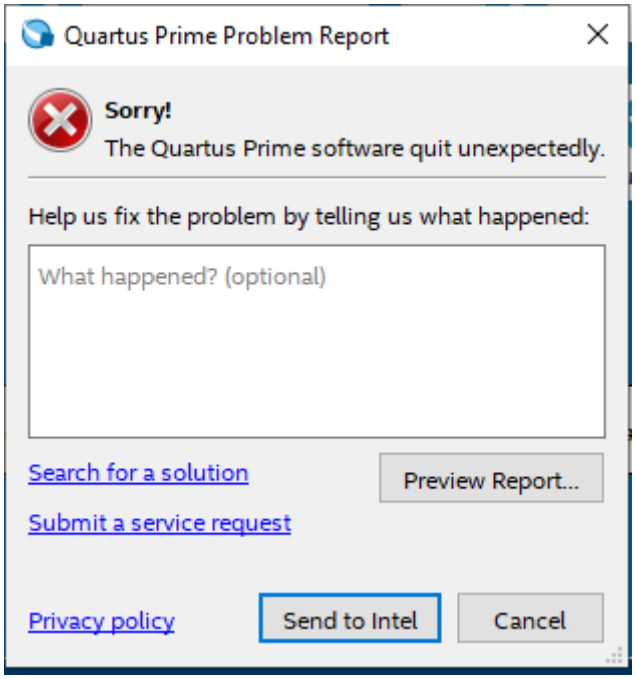
- Para “burlar” o erro da simulação
  - *Assignments > settings, altere de ModelSim-Altera para ModelSim (e vice-versa).*



Problemas de simulação

- Problema ao fazer uma contagem binária





- Solução: setar os valores em cada bit individualmente.

- Próxima aula: Código concorrente WHEN - ELSE.