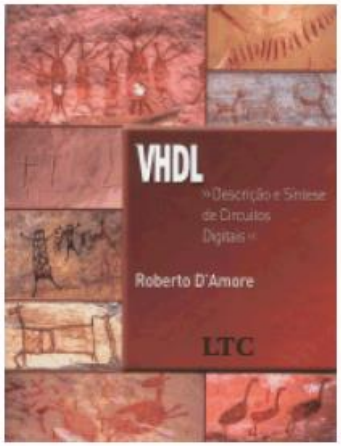
SEL5752/SEL0632 – Linguagens de Descrição de Hardware
Aula 09 – Padrão IEEE-1164

Prof. Dr. Maximiliam Luppe

Livro adotado:

VHDL - Descrição e Síntese de Circuitos Digitais
Roberto d'Amore

ISBN 85-216-1452-7
Editora LTC www.ltceditora.com.br



Para informações adicionais consulte: **www.ele.ita.br/~damore/vhdl**

Padrão IEEE 1164

Tópicos

- Introdução
- Tipos definidos
- Funções e sub-tipos definidos
- Descrições empregando o pacote std_logic_1164
- Síntese empregando o pacote std_logic_1164

Introdução: motivos do desenvolvimento do padrão IEEE 1164

- **Tipo bit não é adequado p/ descrever**:
  - barramentos de comunicação:
    - necessário estado de alta impedância
    - VHDL: não permite sinal acionado por mais de um controlador acionar
      - necessário uma função de resolução nestes casos

  - situações encontradas numa simulação:
    - Valor inicial 0 assumido para um objeto do tipo bit sem inicialização:
      pode encobrir um comportamento não previsto na descrição

- **Pacote std_logic_1164**:
  - define novos tipos e funções que suprem estas deficiências

Tipos definidos no pacote std_logic_1164

- **Tipos caracterizados por um estado lógico**
  - estado lógico representa: nível lógico e força

- **Conceito de força associado**:
  - possível estabelecer um critério de decisão quando
    - controladores acionam um mesmo sinal com níveis lógicos diferentes

- **Pacote std_logic_1164** - define o tipo enumerado std_ulogic:

valor do tipo std_ulogic		estado lógico corresponde			
U		uninitialized		não inicializado	
X		forcing unknown		impondo desconhecido	
0	1	forcing 0	forcing 1	impondo 0	impondo 1
W		weak unknown		desconhecido fraco	
L	H	weak 0	weak 1	0 fraco	1 fraco
Z		high impedance		alta impedância	
-		do not care		não importa	

```
library std ;
use std.standard.all ;

PACKAGE std_logic_1164 IS

    ------------------------------------------------------------------
    -- logic state system  (unresolved)
    ------------------------------------------------------------------
    TYPE std_ulogic IS ( 'U',  -- Uninitialized
                         'X',  -- Forcing  Unknown
                         '0',  -- Forcing  0
                         '1',  -- Forcing  1
                         'Z',  -- High Impedance
                         'W',  -- Weak     Unknown
                         'L',  -- Weak     0
                         'H',  -- Weak     1
                         '-'   -- Don't care
                       );
```

Observação

- **Na linguagem VHDL não é possível**:

  - sinal ser acionado por mais de um controlador sem uma função de resolução

- **Função de resolução**:

  - estabelece o valor final de um sinal

    dado os valores impostos pelos diversos controladores que o acionam

- **Exemplo**:

  - sinal s controlado por x e y

  - qual o valor atribuído ao sinal s se x = 1 e y = 0 ??

```
s  <= y;
s  <= x;
```

- sinal s deve ser declarado segundo um tipo com função de resolução

Tipos definidos no pacote std_logic_1164

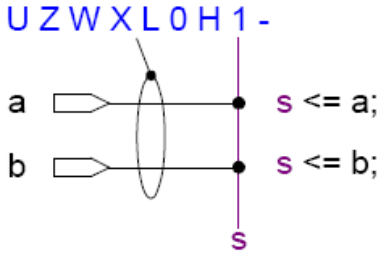
- **Tipo** std_**u**logic:
  - não tem uma função de resolução incorporada →
    - somente um controlador pode acionar um sinal declarado como std_**u**logic

- **Tipo** std_logic:
  - sub-tipo de std_**u**logic
  - contém o mesmo conjunto de valores
  - possui uma função de resolução incorporada →
    - mais de um controlador pode acionar um sinal declarado como std_logic

```
-------------------------------------------------------------------
-- unconstrained array of std_ulogic for use with the resolution function
-------------------------------------------------------------------
TYPE std_ulogic_vector IS ARRAY ( NATURAL RANGE <> ) OF std_ulogic;

-------------------------------------------------------------------
-- resolution function
-------------------------------------------------------------------
FUNCTION resolved ( s : std_ulogic_vector ) RETURN std_ulogic;


-------------------------------------------------------------------
-- *** industry standard logic type ***
-------------------------------------------------------------------
SUBTYPE std_logic IS resolved std_ulogic;

-------------------------------------------------------------------
-- unconstrained array of std_logic for use in declaring signal arrays
-------------------------------------------------------------------
TYPE std_logic_vector IS ARRAY ( NATURAL RANGE <> ) OF std_logic;
```

Função de resolução - tipos std_logic

- **2 controladores acionando o mesmo sinal**



U	U	U	U	U	U	U	U
X	X	X	X	X	X	X	U
0 / 1	X	0 / 1	0 / 1	0 / 1	*Nota A	X	U
W	X	W	W	W	0 / 1	X	U
L / H	X	L / H	*Nota B	W	0 / 1	X	U
Z	X	Z	L / H	W	0 / 1	X	U
-	X	X	X	X	X	X	U
	-	**Z**	**L / H**	**W**	**0 / 1**	**X**	**U**

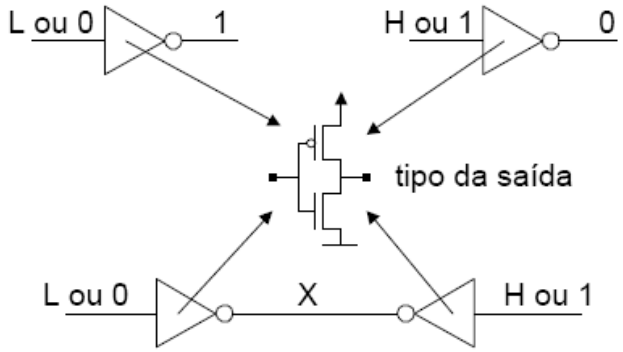
*Nota A: controladores com níveis lógicos: iguais → resulta no mesmo valor;
diferentes → resulta no valor X

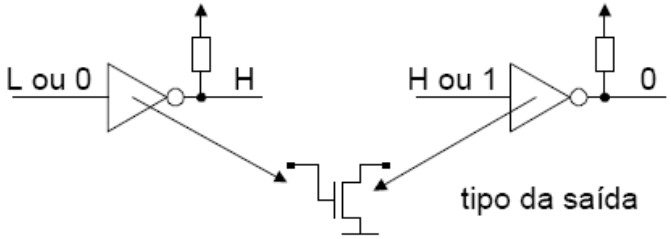
*Nota B: controladores com níveis lógicos: iguais → resulta no mesmo valor
diferentes → resulta no valor W

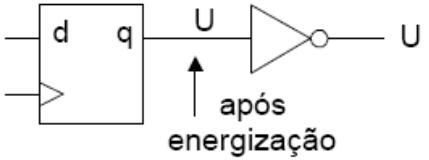
```
PACKAGE BODY std_logic_1164 IS

    ------------------------------------------------------------------
    -- local types
    ------------------------------------------------------------------
    TYPE stdlogic_1d IS ARRAY (std_ulogic) OF std_ulogic;
    TYPE stdlogic_table IS ARRAY(std_ulogic, std_ulogic) OF std_ulogic;

    ------------------------------------------------------------------
    -- resolution function
    ------------------------------------------------------------------
    CONSTANT resolution_table : stdlogic_table := (
    --      ---------------------------------------------------
    --      |  U    X    0    1    Z    W    L    H    -      |    |
    --      ---------------------------------------------------
            ( 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U' ), -- | U |
            ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' ), -- | X |
            ( 'U', 'X', '0', 'X', '0', '0', '0', '0', 'X' ), -- | 0 |
            ( 'U', 'X', 'X', '1', '1', '1', '1', '1', 'X' ), -- | 1 |
            ( 'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', 'X' ), -- | Z |
            ( 'U', 'X', '0', '1', 'W', 'W', 'W', 'W', 'X' ), -- | W |
            ( 'U', 'X', '0', '1', 'L', 'W', 'L', 'W', 'X' ), -- | L |
            ( 'U', 'X', '0', '1', 'H', 'W', 'W', 'H', 'X' ), -- | H |
            ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' )  -- | - |
        );
```

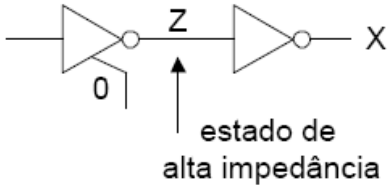
```
FUNCTION resolved ( s : std_ulogic_vector ) RETURN std_ulogic IS
    VARIABLE result : std_ulogic := 'Z';  -- weakest state default
    ATTRIBUTE synthesis_return OF result:VARIABLE IS "WIRED_THREE_STATE" ;
BEGIN
    -- the test for a single driver is essential otherwise the
    -- loop would return 'X' for a single driver of '-' and that
    -- would conflict with the value of a single driver unresolved
    -- signal.
    IF     (s'LENGTH = 1) THEN    RETURN s(s'LOW);
    ELSE
        FOR i IN s'RANGE LOOP
            result := resolution_table(result, s(i));
        END LOOP;
    END IF;
    RETURN result;
END resolved;
```

Tipos definidos - exemplos









Funções e sub-tipos definidos

- **Sub-tipos de** std_**u**logic
  - Troca de valores possíveis

- **Todos definidos com função de resolução** (denominada **resolved**)
  - mais de um controlador pode acionar

```
SUBTYPE std_logic IS resolved std_ulogic;
SUBTYPE X01       IS resolved std_ulogic RANGE 'X' TO '1'; --('X','0','1')
SUBTYPE X01Z      IS resolved std_ulogic RANGE 'X' TO 'Z'; --('X','0','1','Z')
SUBTYPE UX01      IS resolved std_ulogic RANGE 'U' TO '1'; --('U','X','0','1')
SUBTYPE UX01Z     IS resolved std_ulogic RANGE 'U' TO 'Z'; --('U','X','0','1','Z')
```

- **Nota**: definição de uma função de resolução → capítulo 14

Operadores lógicos para tipos std_ulogic

- **Válidos para os sub-tipos**:
  - std_logic X01 X01Z UX01 UX01Z

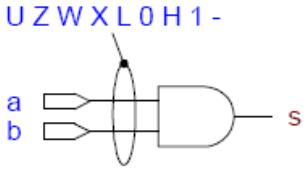
- **Valor de retorno tipo UX01**:
  - modela porta que impõe estados com maior força: 1 0 U X

```
FUNCTION "and"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "nand" ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "or"   ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "nor"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "xor"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "xnor" ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "not"  ( l : std_ulogic                 ) RETURN UX01;
```

- **Nota**: Função xnor não suportada no padrão VHDL-1987

Operadores lógicos para tipos std_ulogic

- **Exemplo**: valor de retorno para a função "and"



s <= a AND b;

```
FUNCTION "and"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
```

- **Observe**: retorna tipo UX01 (modela saída com força elevada)

	U	X	0	1	Z	W	L	H	-
U	U	U	0	U	U	U	0	U	U
X	U	X	0	X	X	X	0	X	X
0	0	0	0	0	0	0	0	0	0
1	U	X	0	1	X	X	0	1	X
Z	U	X	0	X	X	X	0	X	X
W	U	X	0	X	X	X	0	X	X
L	0	0	0	0	0	0	0	0	0
H	U	X	0	1	X	X	0	1	X
-	U	X	0	X	X	X	0	X	X

```
-------------------------------------------------------------------
-- tables for logical operations
-------------------------------------------------------------------

-- truth table for "and" function
CONSTANT and_table : stdlogic_table := (
--      ---------------------------------------------------
--      |  U    X    0    1    Z    W    L    H    -      |    |
--      ---------------------------------------------------
        ( 'U', 'U', '0', 'U', 'U', 'U', '0', 'U', 'U' ),  -- | U |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ),  -- | X |
        ( '0', '0', '0', '0', '0', '0', '0', '0', '0' ),  -- | 0 |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ),  -- | 1 |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ),  -- | Z |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ),  -- | W |
        ( '0', '0', '0', '0', '0', '0', '0', '0', '0' ),  -- | L |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ),  -- | H |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' )   -- | - |
);

FUNCTION "and"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01 IS
    VARIABLE result : UX01 ;
    ATTRIBUTE synthesis_return OF result:VARIABLE IS "AND" ;
BEGIN
    result := (and_table(l, r));
    RETURN result ;
END "and";
```

Operadores lógicos para tipos std_logic_vector std_ulogic_vector

```
FUNCTION "and"  ( l, r : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "and"  ( l, r : std_ulogic_vector) RETURN std_ulogic_vector;
FUNCTION "nand" ( l, r : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "nand" ( l, r : std_ulogic_vector) RETURN std_ulogic_vector;
FUNCTION "or"   ( l, r : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "or"   ( l, r : std_ulogic_vector) RETURN std_ulogic_vector;
FUNCTION "nor"  ( l, r : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "nor"  ( l, r : std_ulogic_vector) RETURN std_ulogic_vector;
FUNCTION "xor"  ( l, r : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "xor"  ( l, r : std_ulogic_vector) RETURN std_ulogic_vector;
FUNCTION "xnor" ( l, r : std_logic_vector ) RETURN std_logic_vector;   -- VHDL-1993
FUNCTION "xnor" ( l, r : std_ulogic_vector) RETURN std_ulogic_vector;  -- VHDL-1993
FUNCTION "not"  ( l    : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "not"  ( l    : std_ulogic_vector) RETURN std_ulogic_vector;
```

- **Note**: **std_logic_vector** não é um sub-tipo de **std_ulogic_vector**:
  - necessário definir funções para cada tipo

Conversão para tipos bit e bit_vector

- **To_bit To_bit_vector**:

  - detectam nível lógico dos objetos de entrada:
    - 1 ou H convertidos → para 1 tipo bit
    - 0 ou L convertidos → para 0 tipo bit
    - outros valores: → retorna o valor de xmap

```
FUNCTION To_bit       (s : std_ulogic;        xmap: BIT := '0') RETURN BIT;
FUNCTION To_bitvector (s : std_logic_vector;  xmap: BIT := '0') RETURN BIT_VECTOR;
FUNCTION To_bitvector (s : std_ulogic_vector; xmap: BIT := '0') RETURN BIT_VECTOR;
```

```
--------------------------------------------------------------------
-- conversion functions
--------------------------------------------------------------------
FUNCTION To_bit       ( s : std_ulogic;        xmap : BIT := '0') RETURN BIT IS
    VARIABLE result : BIT ;
    ATTRIBUTE synthesis_return OF result:VARIABLE IS "FEED_THROUGH" ;
BEGIN
        CASE s IS
            WHEN '0' | 'L' => result := '0';
            WHEN '1' | 'H' => result := '1';
            WHEN OTHERS => result := xmap;
        END CASE;
        RETURN result ;
END;
--------------------------------------------------------------------
FUNCTION To_bitvector ( s : std_logic_vector ; xmap : BIT := '0') RETURN BIT_VECTOR IS
    ALIAS sv : std_logic_vector ( s'LENGTH-1 DOWNTO 0 ) IS s;
    VARIABLE result : BIT_VECTOR ( s'LENGTH-1 DOWNTO 0 );
    ATTRIBUTE synthesis_return OF result:VARIABLE IS "FEED_THROUGH" ;
BEGIN
    FOR i IN result'RANGE LOOP
        CASE sv(i) IS
            WHEN '0' | 'L' => result(i) := '0';
            WHEN '1' | 'H' => result(i) := '1';
            WHEN OTHERS => result(i) := xmap;
        END CASE;
    END LOOP;
    RETURN result;
END;
```

Conversão para os tipos X01 X01Z UX01Z

- **Estabelecida correspondência conforme tabela da próxima imagem**

```
FUNCTION To_X01  ( s : std_logic_vector  ) RETURN  std_logic_vector;
FUNCTION To_X01  ( s : std_ulogic_vector ) RETURN  std_ulogic_vector;
FUNCTION To_X01  ( s : std_ulogic        ) RETURN  X01;
FUNCTION To_X01  ( b : BIT_VECTOR        ) RETURN  std_logic_vector;
FUNCTION To_X01  ( b : BIT_VECTOR        ) RETURN  std_ulogic_vector;
FUNCTION To_X01  ( b : BIT               ) RETURN  X01;
FUNCTION To_X01Z ( s : std_logic_vector  ) RETURN  std_logic_vector;
FUNCTION To_X01Z ( s : std_ulogic_vector ) RETURN  std_ulogic_vector;
FUNCTION To_X01Z ( s : std_ulogic        ) RETURN  X01Z;
FUNCTION To_X01Z ( b : BIT_VECTOR        ) RETURN  std_logic_vector;
FUNCTION To_X01Z ( b : BIT_VECTOR        ) RETURN  std_ulogic_vector;
FUNCTION To_X01Z ( b : BIT               ) RETURN  X01Z;
FUNCTION To_UX01 ( s : std_logic_vector  ) RETURN  std_logic_vector;
FUNCTION To_UX01 ( s : std_ulogic_vector ) RETURN  std_ulogic_vector;
FUNCTION To_UX01 ( s : std_ulogic        ) RETURN  UX01;
FUNCTION To_UX01 ( b : BIT_VECTOR        ) RETURN  std_logic_vector;
FUNCTION To_UX01 ( b : BIT_VECTOR        ) RETURN  std_ulogic_vector;
FUNCTION To_UX01 ( b : BIT               ) RETURN  UX01;
```

Conversão para os tipos X01 X01Z UX01Z

- **Tabela de correspondência de valores**:

	valor convertido		
valor de entrada	função: `To_X01`	função: `To_X01Z`	função: `To_UX01`
U	X	X	U
X	X	X	X
0	0	0	0
1	1	1	1
Z	X	Z	X
W	X	X	X
L	0	0	0
H	1	1	1
-	X	X	X

```
-------------------------------------------------------------------
-- conversion tables
-------------------------------------------------------------------
TYPE logic_x01_table IS ARRAY (std_ulogic'LOW TO std_ulogic'HIGH) OF X01;
TYPE logic_x01z_table IS ARRAY (std_ulogic'LOW TO std_ulogic'HIGH) OF X01Z;
TYPE logic_ux01_table IS ARRAY (std_ulogic'LOW TO std_ulogic'HIGH) OF UX01;
-------------------------------------------------------------
-- table name : cvt_to_x01
--
-- parameters :
--         in  :  std_ulogic  -- some logic value
-- returns     :  x01         -- state value of logic value
-- purpose     :  to convert state-strength to state only
--
-- example     : if (cvt_to_x01 (input_signal) = '1' ) then ...
--
-------------------------------------------------------------
CONSTANT cvt_to_x01 : logic_x01_table := (
                      'X',  -- 'U'
                      'X',  -- 'X'
                      '0',  -- '0'
                      '1',  -- '1'
                      'X',  -- 'Z'
                      'X',  -- 'W'
                      '0',  -- 'L'
                      '1',  -- 'H'
                      'X'   -- '-'
                     );
```

```
    ------------------------------------------------------------------------
    -- strength strippers and type convertors
    ------------------------------------------------------------------------
    -- to_x01
    ------------------------------------------------------------------------
    FUNCTION To_X01  ( s : std_logic_vector ) RETURN  std_logic_vector IS
        ALIAS sv : std_logic_vector ( 1 TO s'LENGTH ) IS s;
        VARIABLE result : std_logic_vector ( 1 TO s'LENGTH );
        ATTRIBUTE synthesis_return OF result:VARIABLE IS "FEED_THROUGH" ;
    BEGIN
        FOR i IN result'RANGE LOOP
            result(i) := cvt_to_x01 (sv(i));
        END LOOP;
        RETURN result;
    END;
    ------------------------------------------------------------------------
...
    ------------------------------------------------------------------------
    FUNCTION To_X01  ( b : BIT ) RETURN  X01 IS
        VARIABLE result : X01 ;
        ATTRIBUTE synthesis_return OF result:VARIABLE IS "FEED_THROUGH" ;
    BEGIN
            CASE b IS
                WHEN '0' => result := ('0');
                WHEN '1' => result := ('1');
            END CASE;
        RETURN result ;
    END;
```

Detecção de bordas de subidas ou descidas

- **Úteis descrição circuitos sensíveis a bordas de um sinal de relógio**

```
FUNCTION rising_edge  (SIGNAL s : std_ulogic) RETURN BOOLEAN;
FUNCTION falling_edge (SIGNAL s : std_ulogic) RETURN BOOLEAN;
```

- **Detecção valores sem correspondência com nível lógico alto ou baixo**
  - Retornam verdadeiro para: U X Z W -

```
FUNCTION Is_X ( s : std_ulogic_vector ) RETURN  BOOLEAN;
FUNCTION Is_X ( s : std_logic_vector  ) RETURN  BOOLEAN;
FUNCTION Is_X ( s : std_ulogic        ) RETURN  BOOLEAN;
```

```
-------------------------------------------------------------------
-- edge detection
-------------------------------------------------------------------
FUNCTION rising_edge  (SIGNAL s : std_ulogic) RETURN BOOLEAN IS
  -- altera built_in builtin_rising_edge
BEGIN
    RETURN (s'EVENT AND (To_X01(s) = '1') AND
                        (To_X01(s'LAST_VALUE) = '0'));
END;

FUNCTION falling_edge (SIGNAL s : std_ulogic) RETURN BOOLEAN IS
  -- altera built_in builtin_falling_edge
BEGIN
    RETURN (s'EVENT AND (To_X01(s) = '0') AND
                        (To_X01(s'LAST_VALUE) = '1'));
END;
```

Descrições empregando o pacote std_logic_1164

- **Na maioria das ferramentas, o pacote std_logic_1164**
  - previamente compilado e armazenado na biblioteca ieee

- **Para tornar os itens do pacote visíveis:**
  - incluir cláusulas : LIBRARY ieee
    USE ieee.std_logic_1164.ALL

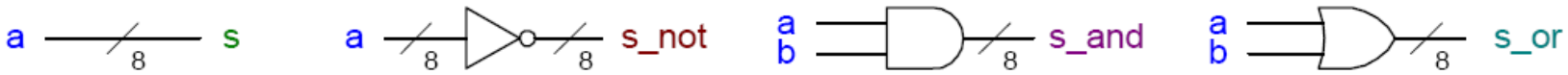
Exemplo empregando o pacote std_logic_1164 funções lógicas

- **Exemplo 1** (resultado da simulação próxima página)

```
 1 LIBRARY ieee;
 2 USE ieee.std_logic_1164.ALL;
 3
 4 ENTITY std1164b IS
 5   PORT( a,b                    : IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
 6         s, s_not, s_and, s_or : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
 7   END std1164b;
 8
 9 ARCHITECTURE exemplo OF std1164b IS
10 BEGIN
11   s     <= a;
12   s_not <=   NOT a;
13   s_and <= a AND b;
14   s_or  <= a OR  b;
15 END exemplo;
```

Exemplo empregando o pacote std_logic_1164 funções lógicas

- **Saída s**: ligação física com a
  - valores são transmitidos sem alteração para saída
- **Saídas restantes**: o valor de retorno é o sub-tipo UX01
- **Conclusão**: modeladas portas com capacidade de drenar e fornecer corrente
  - níveis lógicos definidos: L H, → convertidos para: 0 1 (maior força)
  - níveis lógico indefinidos: Z W, → convertidos para: X (maior força)



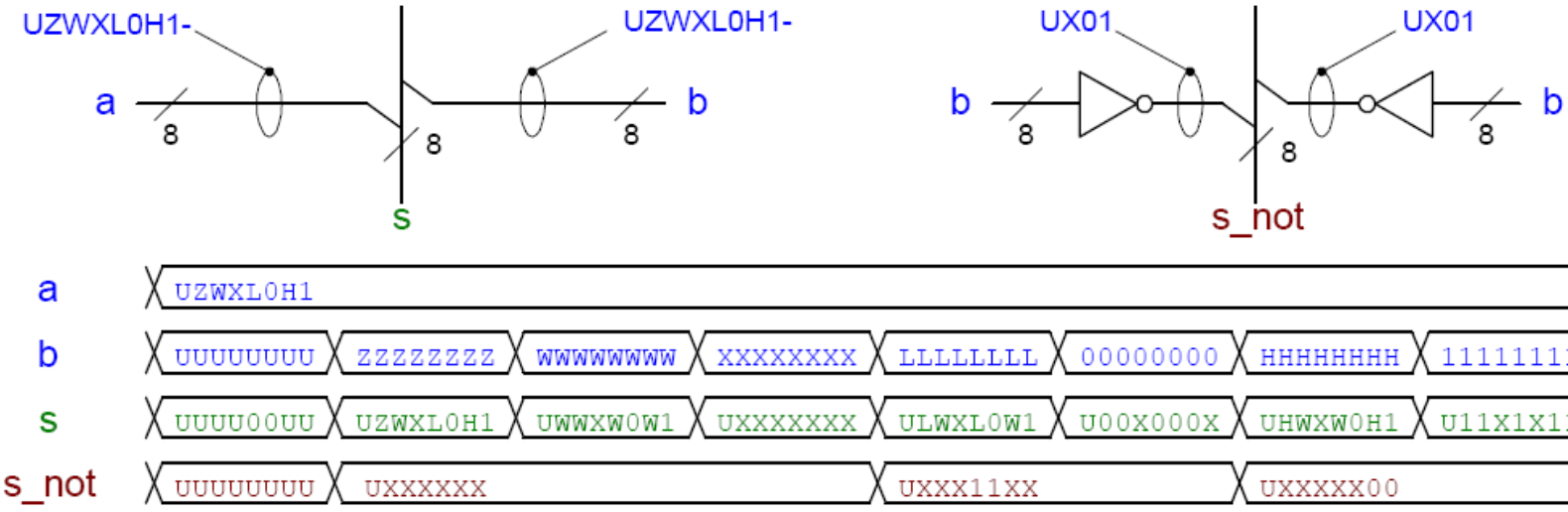
Exemplo empregando o pacote std_logic_1164

- **Sinais s s_not**: vetores de 8 bits
- **s controlado por**: a e b
- **s_not controlado por**: a e b

```
 1 LIBRARY ieee;
 2 USE ieee.std_logic_1164.ALL;
 3
 4 ENTITY std1164c IS
 5   PORT( a, b     : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
 6         s, s_not : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
 7   END std1164c;
 8
 9 ARCHITECTURE exemplo OF std1164c IS
10 BEGIN
11   s     <= a;         s     <= b;
12   s_not <= NOT a;   s_not  <= NOT b;
13 END exemplo;
```

Exemplo empregando o pacote std_logic_1164

- **Valores no sinal s**:
  - decididos segundo a função de resolução
- **Valores no sinal s_not**:
  - s_not <= NOT a; função Not retorna sub-tipo UX01
  - s_not <= NOT b; função Not retorna sub-tipo UX01
  - decisão segundo função de resolução aplicada entre valores UX01



Operações com o valor não importa " - "

- **Valor** " - ":
  - pela definição do padrão: condição não importa
- **Simulador**:
  - não incorpora o significado desta condição
- **Exemplo**: teste da condição y = '-'
  - deveria retornar sempre valor verdadeiro
  - retorna TRUE unicamente se y contiver o valor não importa
- **Significado do valor " - "**
  - considerado na função std_match pacote ieee.numeric_std
  - assunto do capítulo 11

Operações com o valor não importa " - "

- **Linha 13**: '1' = '-' falso (não incorpora significado)
- **Linha 14**: '1' = 'X' falso (esperado)
- **Linha 15**: '-' = '-' verdadeiro (esperado)

```
 1 LIBRARY ieee;
 2 USE ieee.std_logic_1164.ALL;
 3
 4 ENTITY std1164j IS
 5   PORT (f, v     : IN  STD_LOGIC;
 6         s1,s2,s3 : OUT STD_LOGIC);
 7 END std1164j;
 8
 9 ARCHITECTURE teste OF std1164j IS
10   CONSTANT nao_importa : STD_LOGIC := '-';
11   CONSTANT valor_1     : STD_LOGIC := '1';
12 BEGIN
13  s1 <= v WHEN valor_1     = '-' ELSE f;  -- equivale a: s1 <= f; (falso)
14  s2 <= v WHEN valor_1     = 'X' ELSE f;  -- equivale a: s2 <= f; (falso)
15  s3 <= v WHEN nao_importa = '-' ELSE f;  -- equivale a: s3 <= v; (verdadeiro)
16  END teste;
```

Síntese de empregando tipos do pacote std_logic_1164

- **Síntese de um tipo enumerado:**
  - necessário estabelecer uma codificação binária dos valores contidos

- **Tipo enumerado std_ulogic**: possui 9 valores
  - 4 linhas para sua codificação ???
    - não é este o objetivo
  - objetivo do padrão:
    - modelar o comportamento de uma única interligação digital

- **Ferramentas de síntese**:
  - estabelecem uma correspondência entre:

    valores ↔ comportamento desejado para a interligação

Síntese de empregando tipos do pacote std_logic_1164

- **Valor -** : sem correspondência com um nível lógico válido
  - ferramenta de síntese: emprega p/ minimização lógica

- **Valor Z**: sem correspondência com um nível lógico válido
  - ferramenta de síntese:
    - infere porta com saída em estado de alta impedância

Síntese de empregando tipos do pacote std_logic_1164

- **Valores U X W** : sem correspondência com nível lógico válido
  - empregados na simulação:
    - detectar conflitos entre sinais acionados
    - definir a condição "não inicializado" para pontos do circuito
  - ferramenta de síntese: equivalência com o valor " - ".

- **Nota**: descrições visando a síntese
  - contendo operações com valores U X W
    - corretas para síntese ???
    - provavelmente não, lembre-se
      - circuito digital interpreta apenas nível lógicos válidos
      - para qualquer outra condição: qual o comportamento do circuito?

Síntese de empregando tipos do pacote std_logic_1164

- **Valores 0 1 L H :** únicos com correspondência a níveis lógicos válidos
  - ferramenta de síntese:
    - operação com valores: (IF y = 'L' THEN ...)
      - equivalência entre: L ↔ 0 e H ↔ 1
      - porta lógica não distingue a força do valor, apenas o nível lógico
    - atribuição de valores: (y <= 'H';)
      - 0 e 1: infere porta convencional
      - H: deveria inferir porta do tipo dreno aberto
        - não ocorre nas ferramentas:
          - seguem o modelagem da funções:
            - elas retornam um tipo UX01 (níveis lógicos com força)

Síntese de empregando tipos do pacote std_logic_1164

- **Exemplo de codificação adotado por uma ferramenta de síntese:**

valores std_ulogic	U	X	0	1	Z	W	L	H	-
valores codificados	X	X	0	1	Z	X	0	1	X

- **Valor X** :
  - semelhante ao valor " - ",
    - dependendo do contexto, a ferramenta escolhe o valor 0 ou 1

```
-------------------------------------------------------------------
-- Directives for synthesis of std_ulogic.
-- Declare the type encoding attribute and set the value for std_ulogic
-- Directives for the resolution function and all other function are
-- in the package body
-------------------------------------------------------------------
ATTRIBUTE logic_type_encoding : string ;
ATTRIBUTE logic_type_encoding of std_ulogic:type is
            -- ('U','X','0','1','Z','W','L','H','-')
               ('X','X','0','1','Z','X','0','1','X') ;
```

Saídas com estado em alta impedância - região de código concorrente

- **Descrição:** região de código concorrente
- **hab = 1 saída habilitada**
- **hab = 0 saída alta impedância**

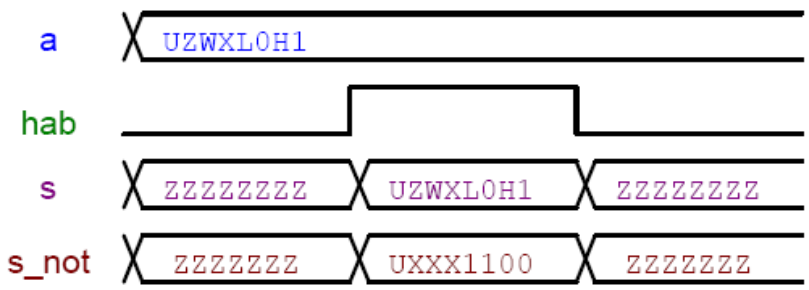
```
 1 LIBRARY ieee;
 2 USE ieee.std_logic_1164.ALL;
 3
 4 ENTITY std1164d IS
 5   PORT( hab      : IN  BIT;
 6         a        : IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
 7         s, s_not : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
 8   END std1164d;
 9
10 ARCHITECTURE exemplo OF std1164d IS
11 BEGIN
12   s     <=     a WHEN hab = '1' ELSE "ZZZZZZZZ";
13   s_not <= NOT a WHEN hab = '1' ELSE "ZZZZZZZZ";
14 END exemplo;
```

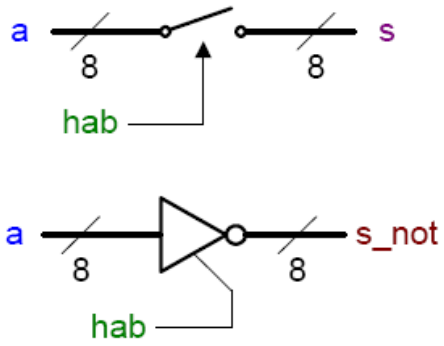
Saídas com estado em alta impedância - região de código concorrente

- **Simulação**

- **Sinal s**
  - hab = 1 s segue a
  - hab = 0 s alta impedância

- **Sinal s_not**
  - hab = 1 s_not apenas valores UX01 (função Not retorna o sub-tipo UX01)
  - hab = 0 s_not alta impedância





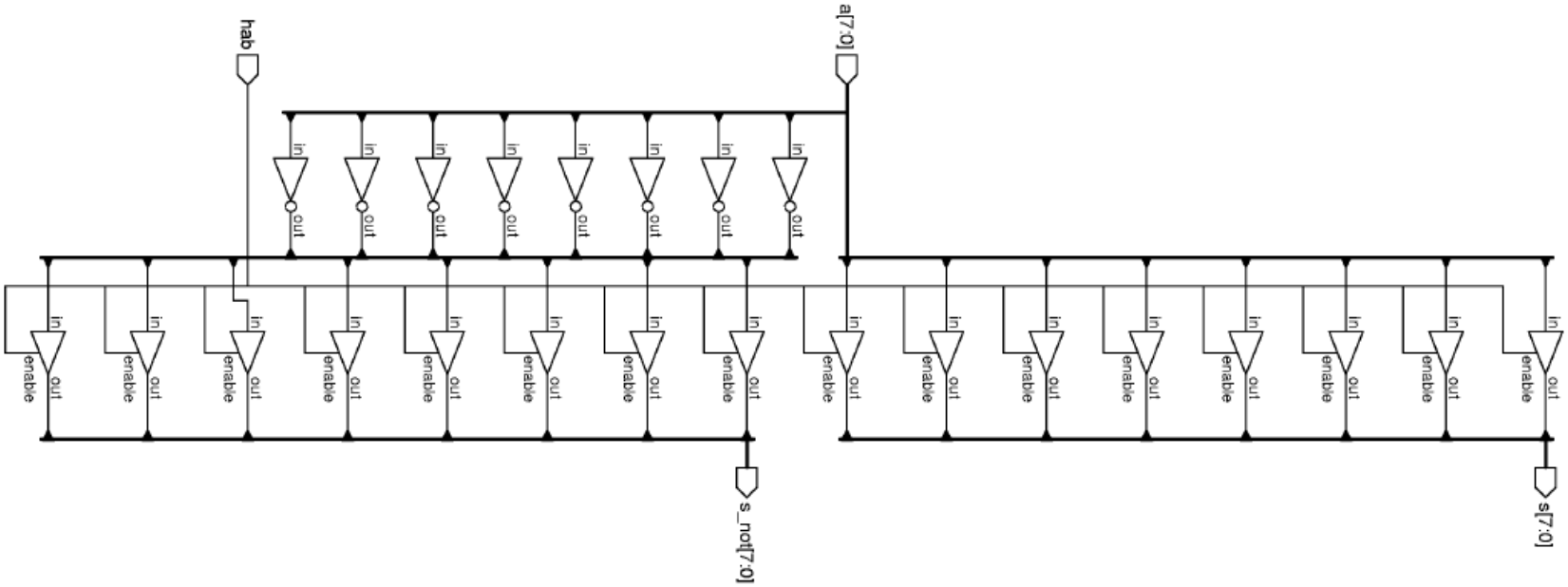
Saídas com estado em alta impedância - região de código seqüencial

- **Descrição**: região de código seqüencial
- **hab = 1 saída habilitada**
- **hab = 0 saída alta impedância**
- **Sinais hab a** → lista de sensibilidade do processo

```
 1 LIBRARY ieee;
 2 USE ieee.std_logic_1164.ALL;
 3
 4 ENTITY std1164e IS
 5   PORT( hab      : IN  BIT;
 6         a        : IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
 7         s, s_not : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
 8   END std1164e;
 9
10 ARCHITECTURE exemplo OF std1164e IS
11 BEGIN
12   abc: PROCESS (a, hab)
13     BEGIN
14     IF (hab = '1') THEN
15       s     <=  a;
16       s_not <= NOT a;
17     ELSE
18       s <= (OTHERS => 'Z');
19       s_not <= "ZZZZZZZZ";
20     END IF;
21   END PROCESS abc;
22 END exemplo;
```

Saídas com estado em alta impedância - circuito sintetizado RTL (2 descrições)

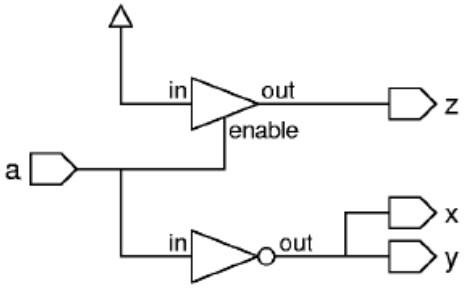
- **Saídas s s_not** → empregam portas com saída em alta impedância
  - unicamente as condições correspondentes aos valores Z 0 1



Exemplo: síntese de uma descrição empregando valores 0 1 L H

- **Conforme visto, a ferramenta estabelece uma codificação entre**:
  - (U X 0 1 Z W L H -) ↔ níveis lógicos alto e baixo
- **Linhas 11 a 14**: o sinal x :
  - os dois nível lógicos válidos (H L) definem o circuito
  - condição OTHERS da linha 14 ignorada
- **O mesmo ocorre nas linhas 15 - 18, sinal y**
  - os dois nível lógicos válidos (1 0) definem o circuito
- **Linhas 19 a 21** : somente um nível lógico → condição OTHERS considerada

```
 9 ARCHITECTURE teste OF std1164i IS
10 BEGIN
11   WITH a SELECT
12     x <= 'H' WHEN '0',     -- H, 1o nivel logico valido
13          'L' WHEN '1',     -- L, 2o nivel logico valido
14          'Z' WHEN OTHERS;
15   WITH a SELECT
16     y <= '1' WHEN 'L',     -- 1, 1o nivel logico valido
17          '0' WHEN 'H',     -- 0, 2o nivel logico valido
18          'Z' WHEN OTHERS;
19   WITH a SELECT
20     z <= '1' WHEN '1',     -- 1, 1o nivel logico valido
21          'Z' WHEN OTHERS;
22 END teste;
```

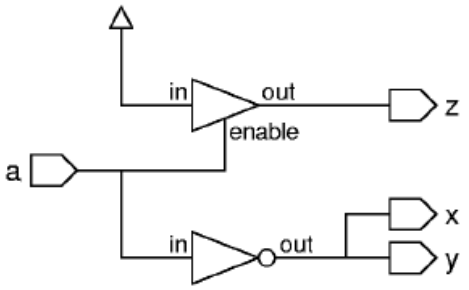


circuito sintetizado

Exemplo: síntese de uma descrição empregando valores 0 1 L H

- **Mensagem da ferramenta de síntese**

```
 1 LIBRARY ieee;
 2 USE ieee.std_logic_1164.ALL;
 3
 4 ENTITY std1164i IS
 5   PORT (a       : IN  STD_LOGIC;
 6         x,y,z   : OUT STD_LOGIC);
 7 END std1164i;
 8
 9 ARCHITECTURE teste OF std1164i IS
10 BEGIN
11   WITH a SELECT
12     x <= 'H' WHEN '0',
13          'L' WHEN '1',
14          'Z' WHEN OTHERS;
15   WITH a SELECT
16     y <= '1' WHEN 'L',
17          '0' WHEN 'H',
18          'Z' WHEN OTHERS;
19   WITH a SELECT
20     z <= '1' WHEN '1',
21          'Z' WHEN OTHERS;
22 END teste;
```



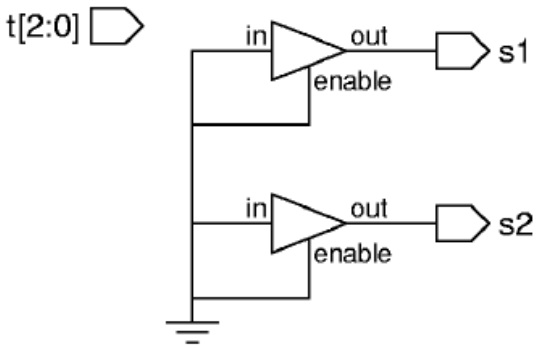
circuito sintetizado

```
-- Compiling root entity std1164i(teste)
"\vhdl\std1164i.vhd",line 11: Info, others clause is never selected for synthesis.
"\vhdl\std1164i.vhd",line 15: Info, others clause is never selected for synthesis.
--
```

Síntese com o valor não importa " - " (comparação com este valor)

- **Simulação**: - comparação " - " não retorna sempre um valor verdadeiro
- **Síntese**: - mesmo comportamento
- **Na descrição**: caso o significado do valor fosse incorporado:
  - Linha 11 " s1 = '1' na condição " t(2), t(0) =0,1 "
  - Linha 12, " s2 = '0' em qualquer condição
- **Ferramenta assume que o valor " - " nunca pode ser imposto**:
  - comparação sempre falsa

```
 1 LIBRARY ieee;
 2 USE ieee.std_logic_1164.ALL;
 3
 4 ENTITY std1164h IS
 5   PORT (t     : IN  STD_LOGIC_VECTOR (2 DOWNTO 0);
 6         s1,s2 : OUT STD_LOGIC);
 7 END std1164h;
 8
 9 ARCHITECTURE teste OF std1164h IS
10 BEGIN
11   s1 <= '1' WHEN t= "0-1"  ELSE 'Z';
12   s2 <= '0' WHEN t(1)= '-' ELSE 'Z';
13 END teste;
```



circuito sintetizado

Detecção de bordas de subida ou descida

- **Simulação**: sinais empregando tipos std_logic
  - deve considerar a existência de outros valores, além de " 0 " e " 1 ".

- **Transição válida**:
  - valores antes e depois do evento:
    - níveis lógicos validos ?
    - exemplo:
      transição entre (Z → 1) não é uma borda de subida válida!

- **Teste de borda de subida**:
  - ocorrência de um evento?
  - valor atual (H ou 1) ? (nível lógico alto)
  - valor anterior (L ou 0) ? (nível lógico baixo)

```
IF ck'EVENT AND (ck='1' OR ck='H') AND (ck'LAST_VALUE='0' OR ck'LAST_VALUE='L') THEN

-- (evento?) e (nivel atual alto?)  e  (       nivel anterior baixo?          )
```

Detecção de bordas de subida ou descida

- **Do ponto de vista de síntese:**
  - atributo LAST_VALUE não é possível de ser sintetizado
  - comportamento da ferramenta nestes casos nem sempre é definido.

- **Conveniente empregar funções disponíveis no disponíveis no pacote**
  - rising_edge : retorna verdadeiro borda de subida
  - falling_edge : retorna verdadeiro borda de descida
  - mantém compatibilidade: simulação ↔ síntese
  - aceito pela ferramenta de síntese

Detecção de bordas de subida ou descida

- **Exemplo de uma descrição**

```
 1 LIBRARY ieee;
 2 USE ieee.std_logic_1164.ALL;
 3
 4 ENTITY std1164g IS
 5   PORT (ck, da, db, dc : IN  STD_LOGIC;
 6         qa, qb, qc     : OUT STD_LOGIC);
 7 END std1164g;
 8
 9 ARCHITECTURE teste OF std1164g IS
10 BEGIN
11  abc: PROCESS (ck)
12  BEGIN
13    IF ck'EVENT AND (ck='1' OR ck='H') AND (ck'LAST_VALUE='0' OR ck'LAST_VALUE='L')
14      THEN qa <= da;
15    END IF;
16
17    IF rising_edge(ck) THEN qb <= db;
18    END IF;
19
20    IF ck'EVENT AND ck='1' THEN qc <= dc;
21    END IF;
22   END PROCESS abc;
23 END teste;
```

Detecção de bordas de subida ou descida

- **Resultado da síntese da descrição**
  - as 3 estruturas foram inferidas corretamente
  - mesmo o teste incompleto → `IF ck'EVENT AND ck='1' THEN qc <= dc;`