VHDL Data types and Operators available in the IEEE Standard Packages

Silvia Chiusano

Politecnico di Torino

<Silvia.Chiusano@polito.it>

Index

1. A short summary

1.1. Declarations

The following libraries must be included:

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.std_logic_arith.ALL;
use ieee.std_logic_unsigned.ALL;
----or use ieee.std_logic_signed.ALL
```

We will not use the operators included in ieee.std_logic_misc.ALL.

Focusing shortly on the single libraries:

- `ieee.std_logic_arith.ALL`:
  - includes operators working on signed and unsigned data types.
- `ieee.std_logic_unsigned.ALL` (or `ieee.std_logic_signed.ALL`):
  - include operators working on `std_logic` and `std_logic_vector` data types.
  - including the `ieee.std_logic_unsigned` library, the `std_logic_vector` data type is considered as a pure binary number.
  - including the `ieee.std_logic_signed` library, the `std_logic_vector` data type is considered as a 2 complement binary number.
  - Just one out of the two libraries must be included.

1.2. Data types

The following data types are available:

- `type integer is range -(2**31-1) to (2**31)`
- `type boolean is (false, true);`
- `type std_logic 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
    );
```

- `type std_logic_vector is array (integer range <>) of std_logic;`
  - It is a bit vector, represented MSB first. According to the included ieee library (`std_logic_unsigned or std_logic_signed`) it is considered as a pure binary number
- `type UNSIGNED is array (NATURAL range <>) of STD_LOGIC;`
  - It is a bit vector, represented MSB first. It is considered as a number encoded as absolute binary.
- `type SIGNED is array (NATURAL range <>) of STD_LOGIC;`
  - It is a bit vector, represented MSB first. It is considered as a 2 complement binary number.

1.3. Operators

Remarks:

- Arithmetic and comparison operators can operate on arguments of a same type, only. Ad-hoc conversion functions (See Table 4) allow type conversions.

  Example:

  signal B, C: std_logic_vector (3 downto 0);

  signal D:unsigned (3 downto 0);

  C <= B+D; **Not possible!**

  C <= B+conv_std_logic_vector(D, 4); **Correct!**

1.3.1. Arithmetic Operators

1.3.1.1. Operators +,-

arg1 operator arg2 ® result, operator: +, -	
arg1, arg2	**result**
both std_logic_vector std_logic_vector, std_logic (or reverse) std_logic_vector, integer (or reverse)	std_logic_vector
both unsigned unsigned, std_logic (or reverse) unsigned, integer (or reverse)	unsigned or std_logic_vector
both signed signed, std_logic (or reverse) signed, integer (or reverse) signed, unsigned (or reverse)	signed or std_logic_vector

Table 1: *Operators +,-*

1.3.1.2. Operators *

arg1 operator arg2 ® result, operator: *	
arg1, arg2	**result**
both std_logic_vector	std_logic_vector
both unsigned	unsigned o std_logic_vector
both signed signed, unsigned (or reverse)	signed o std_logic_vector

Table 2: *Operators **

1.3.2. Comparison Operators <,<=,>,>=,=,/=

arg1 operator arg2 ® result, operator: <,<=,>,>=,=,/=	
arg1, arg2	**result**
both std_logic_vector std_logic_vector, integer (or reverse)	boolean
both unsigned both signed unsigned, integer (or reverse) signed, integer (or reverse) signed, unsigned (or reverse)	boolean

Table 3: *Comparison Operators*

1.3.3. Conversion Operators

Operator	Argument (arg)	Result
conv_integer(arg)	std_logic std_logic_vector signed unsigned	integer
conv_unsigned(arg, size: integer) size: number of bits of the final result	std_logic signed unsigned integer	unsigned
conv_signed(arg, size: integer) size: number of bits of the final result	std_logic signed unsigned integer	signed
conv_std_logic_vector(arg, size: integer) size: number of bits of the final result	std_logic signed unsigned integer	std_logic_vector

Table 4: *Conversion Operators*

1.3.4. Shift Operators

Operator	Argument (arg)
SHL(arg, size: std_logic_vector) → result SHR(arg, size: std_logic_vector) → result size: number of bits to shift	std_logic_vector
SHL(arg, size: unsigned) → result SHR(arg, size: unsigned) → result size: number of bits to shift	signed unsigned

Table 5: *Shift Operators*

2. IEEE Standard Packages

2.1. Package Standard

```
------------------------------------------------------------------------
------------------------------------------------------------------------
--                                                                    --
--                  package standard                                  --
--                                                                    --
-- Contents:            Standard type and subprogram declarations.    --
--                  See the VHDL reference manual for a complete      --
--                  explanation of these types and functions.         --
--                                                                    --
-- Exported types:      boolean                                       --
--                  bit                                               --
--                  vlbit                                             --
--                  character                                         --
--                  boolean_1d                                        --
--                  boolean_2d                                        --
--                  vlbit_1d                                          --
--                  vlbit_2d                                          --
--                  character_1d                                      --
--                  character_2d                                      --
--                  integer_1d                                        --
--                  integer_2d                                        --
--                  time_1d                                           --
--                  time_2d                                           --
--                  string                                            --
--                  vlbit_vector                                      --
--                                                                    --
-- Exported functions:  addum                                         --
--                  add2c                                             --
--                  comp2c                                            --
--                  divum                                             --
--                  div2c                                             --
--                  extendum                                          --
--                  extend2c                                          --
--                  mulum                                             --
--                  mul2c                                             --
--                  shiftlum                                          --
--                  shiftl2c                                          --
--                  shiftrum                                          --
--                  shiftr2c                                          --
--                  subum                                             --
--                  sub2c                                             --
--                  int2vlb                                           --
--                  boo2vlb                                           --
--                  vlb2boo                                           --
--                  int2boo                                           --
--                  vlb2int                                           --
--                  boo2int                                           --
```

```
--                  int2v1d                          --
--                  int2v1d                          --
--                  v1d2int                          --
--                                                --
-- Revision:          1.1                             --
--                                                --
-- Copyright (C) 1988, 1989 Viewlogic Systems, Inc.                --
--                                                --
------------------------------------------------------------------------
------------------------------------------------------------------------
package standard is

  type boolean     is (false, true);
  type bit         is ('0', '1');
  type vlbit       is ('X', 'Z', '0', '1');
  type character   is (
      NUL,  SOH,  STX,  ETX,  EOT,  ENQ,  ACK,  BEL,
      BS,   HT,   LF,   VT,   FF,   CR,   SO,   SI,
      DLE,  DC1,  DC2,  DC3,  DC4,  NAK,  SYN,  ETB,
      CAN,  EM,   SUB,  ESC,  FSP,  GSP,  RSP,  USP,
      ' ',  '!',  '"',  '#',  '$',  '%',  '&',  ''',
      '(',  ')',  '*',  '+',  ',',  '-',  '.',  '/',
      '0',  '1',  '2',  '3',  '4',  '5',  '6',  '7',
      '8',  '9',  ':',  ';',  '<',  '=',  '>',  '?',
      '@',  'A',  'B',  'C',  'D',  'E',  'F',  'G',
      'H',  'I',  'J',  'K',  'L',  'M',  'N',  'O',
      'P',  'Q',  'R',  'S',  'T',  'U',  'V',  'W',
      'X',  'Y',  'Z',  '[',  '\',  ']',  '^',  '_',
      '`',  'a',  'b',  'c',  'd',  'e',  'f',  'g',
      'h',  'i',  'j',  'k',  'l',  'm',  'n',  'o',
      'p',  'q',  'r',  's',  't',  'u',  'v',  'w',
      'x',  'y',  'z',  '{',  '|',  '}',  '~',  DEL);

  type boolean_1d  is array (integer range <>)
                   of boolean;
  type boolean_2d  is array (integer range <>, integer range <>)
                   of boolean;
  type vlbit_1d    is array (integer range <>)
                   of vlbit;
  type vlbit_2d    is array (integer range <>, integer range <>)
                   of vlbit;
  type character_1d is array (integer range <>)
                   of character;
  type character_2d is array (integer range <>, integer range <>)
                   of character;
  type integer_1d  is array (integer range <>)
                   of integer;
  type integer_2d  is array (integer range <>, integer range <>)
                   of integer;
  type time_1d     is array (integer range <>)
                   of time;
  type time_2d     is array (integer range <>, integer range <>)
                   of time;
  type string      is array (integer range <>)
                   of character;
  type bit_vector  is array (integer range <>)
```

```
                of bit;
    type vlbit_vector is array (integer range <>)
                of vlbit;

    function addum          (v1, v2: vlbit_1d)          return vlbit_1d;
    function add2c          (v1, v2: vlbit_1d)          return vlbit_1d;
    function comp2c         (v:      vlbit_1d)          return vlbit_1d;
    function divum          (v1, v2: vlbit_1d)          return vlbit_1d;
    function div2c          (v1, v2: vlbit_1d)          return vlbit_1d;
    function extendum       (v: vlbit_1d; i: integer)   return vlbit_1d;
    function extend2c       (v: vlbit_1d; i: integer)   return vlbit_1d;
    function mulum          (v1, v2: vlbit_1d)          return vlbit_1d;
    function mul2c          (v1, v2: vlbit_1d)          return vlbit_1d;
    function shiftlum       (v:      vlbit_1d; i: integer) return vlbit_1d;
    function shiftl2c       (v:      vlbit_1d; i: integer) return vlbit_1d;
    function shiftrum       (v:      vlbit_1d; i: integer) return vlbit_1d;
    function shiftr2c       (v:      vlbit_1d; i: integer) return vlbit_1d;
    function subum          (v1, v2: vlbit_1d)          return vlbit_1d;
    function sub2c          (v1, v2: vlbit_1d)          return vlbit_1d;

    function int2vlb        (i: integer) return vlbit;
    function boo2vlb        (b: boolean) return vlbit;
    function vlb2boo        (v: vlbit)   return boolean;
    function int2boo        (i: integer) return boolean;
    function vlb2int        (v: vlbit)   return integer;
    function boo2int        (b: boolean) return integer;

    function int2vec        (i: integer)  return vlbit_vector;
    function int2v1d        (i: integer)  return vlbit_1d;
    function v1d2int        (v: vlbit_1d) return integer;

end standard;
```

2.2. Package Std_logic_1164

```
-- ----------------------------------------------------------------------
--
--  Title     :  std_logic_1164 multi-value logic system
--  Library   :  This package shall be compiled into a library
--            :  symbolically named IEEE.
--            :
--  Developers:  IEEE model standards group (par 1164)
--  Purpose   :  This packages defines a standard for designers
--            :  to use in describing the interconnection data types
--            :  used in vhdl modeling.
--            :
--  Limitation:  The logic system defined in this package may
--            :  be insufficient for modeling switched transistors,
--            :  since such a requirement is out of the scope of this
--            :  effort. Furthermore, mathematics, primitives,
--            :  timing standards, etc. are considered orthogonal
--            :  issues as it relates to this package and are therefore
--            :  beyond the scope of this effort.
--            :
--  Note      :  No declarations or definitions shall be included in,
```

```
--            :  or excluded from this package. The "package declaration"
--            :  defines the types, subtypes and declarations of
--            :  std_logic_1164. The std_logic_1164 package body shall be
--            :  considered the formal definition of the semantics of
--            :  this package. Tool developers may choose to implement
--            :  the package body in the most efficient manner available
--            :  to them.
--            :
-- ---------------------------------------------------------------------
--   modification history :
-- ---------------------------------------------------------------------
--  version | mod. date:|
--   v4.200 | 01/02/92  |
-- ---------------------------------------------------------------------

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
                      );
   ---------------------------------------------------------------------
   -- 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;

   ---------------------------------------------------------------------
   -- common subtypes
   ---------------------------------------------------------------------
   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')

    ------------------------------------------------------------------
    -- overloaded logical operators
    ------------------------------------------------------------------

    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;

    ------------------------------------------------------------------
    -- vectorized overloaded logical operators
    ------------------------------------------------------------------
    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;

--  ------------------------------------------------------------------------
--  Note : The declaration and implementation of the "xnor" function is
--  specifically commented until at which time the VHDL language has been
--  officially adopted as containing such a function. At such a point,
--  the following comments may be removed along with this notice without
--  further "official" ballotting of this std_logic_1164 package. It is
--  the intent of this effort to provide such a function once it becomes
--  available in the VHDL standard.
--  ------------------------------------------------------------------------
--  function "xnor" ( l, r : std_logic_vector  ) return std_logic_vector;
--  function "xnor" ( l, r : std_ulogic_vector ) return std_ulogic_vector;

    FUNCTION "not"  ( l : std_logic_vector  ) RETURN std_logic_vector;
    FUNCTION "not"  ( l : std_ulogic_vector ) RETURN std_ulogic_vector;

    ------------------------------------------------------------------
    -- conversion functions
    ------------------------------------------------------------------
    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;

    FUNCTION To_StdULogic       ( b : BIT             ) RETURN std_ulogic;
```

```
    FUNCTION To_StdLogicVector  ( b : BIT_VECTOR        ) RETURN std_logic_vector;
    FUNCTION To_StdLogicVector  ( s : std_ulogic_vector ) RETURN std_logic_vector;
    FUNCTION To_StdULogicVector ( b : BIT_VECTOR        ) RETURN std_ulogic_vector;
    FUNCTION To_StdULogicVector ( s : std_logic_vector  ) RETURN std_ulogic_vector;

    --------------------------------------------------------------------
    -- strength strippers and type convertors
    --------------------------------------------------------------------

    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;

    --------------------------------------------------------------------
    -- edge detection
    --------------------------------------------------------------------
    FUNCTION rising_edge  (SIGNAL s : std_ulogic) RETURN BOOLEAN;
    FUNCTION falling_edge (SIGNAL s : std_ulogic) RETURN BOOLEAN;

    --------------------------------------------------------------------
    -- object contains an unknown
    --------------------------------------------------------------------
    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;

END std_logic_1164;
```

2.3. Package Std_logic_arith

```
-------------------------------------------------------------------------
--                                                                     --
-- Copyright (c) 1990,1991,1992 by Synopsys, Inc.  All rights reserved. --
--                                                                     --
-- This source file may be used and distributed without restriction    --
-- provided that this copyright statement is not removed from the file --
```

```
-- and that any derivative work contains this copyright notice.        --
--                                                                     --
--      Package name: STD_LOGIC_ARITH                                  --
--                                                                     --
--      Purpose:                                                       --
--       A set of arithemtic, conversion, and comparison functions     --
--       for SIGNED, UNSIGNED, SMALL_INT, INTEGER,                     --
--       STD_ULOGIC, STD_ LOGIC, and STD_LOGIC_VECTOR.                 --
--                                                                     --
-------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

package std_logic_arith is

   type UNSIGNED is array (NATURAL range <>) of STD_LOGIC;
   type SIGNED is array (NATURAL range <>) of STD_LOGIC;
   subtype SMALL_INT is INTEGER range 0 to 1;


   function "+"(L: UNSIGNED; R: UNSIGNED) return UNSIGNED;
   function "+"(L: SIGNED; R: SIGNED) return SIGNED;
   function "+"(L: UNSIGNED; R: SIGNED) return SIGNED;
   function "+"(L: SIGNED; R: UNSIGNED) return SIGNED;
   function "+"(L: UNSIGNED; R: INTEGER) return UNSIGNED;
   function "+"(L: INTEGER; R: UNSIGNED) return UNSIGNED;
   function "+"(L: SIGNED; R: INTEGER) return SIGNED;
   function "+"(L: INTEGER; R: SIGNED) return SIGNED;
   function "+"(L: UNSIGNED; R: STD_ULOGIC) return UNSIGNED;
   function "+"(L: STD_ULOGIC; R: UNSIGNED) return UNSIGNED;
   function "+"(L: SIGNED; R: STD_ULOGIC) return SIGNED;
   function "+"(L: STD_ULOGIC; R: SIGNED) return SIGNED;

   function "+"(L: UNSIGNED; R: UNSIGNED) return STD_LOGIC_VECTOR;
   function "+"(L: SIGNED; R: SIGNED) return STD_LOGIC_VECTOR;
   function "+"(L: UNSIGNED; R: SIGNED) return STD_LOGIC_VECTOR;
   function "+"(L: SIGNED; R: UNSIGNED) return STD_LOGIC_VECTOR;
   function "+"(L: UNSIGNED; R: INTEGER) return STD_LOGIC_VECTOR;
   function "+"(L: INTEGER; R: UNSIGNED) return STD_LOGIC_VECTOR;
   function "+"(L: SIGNED; R: INTEGER) return STD_LOGIC_VECTOR;
   function "+"(L: INTEGER; R: SIGNED) return STD_LOGIC_VECTOR;
   function "+"(L: UNSIGNED; R: STD_ULOGIC) return STD_LOGIC_VECTOR;
   function "+"(L: STD_ULOGIC; R: UNSIGNED) return STD_LOGIC_VECTOR;
   function "+"(L: SIGNED; R: STD_ULOGIC) return STD_LOGIC_VECTOR;
   function "+"(L: STD_ULOGIC; R: SIGNED) return STD_LOGIC_VECTOR;

   function "-"(L: UNSIGNED; R: UNSIGNED) return UNSIGNED;
   function "-"(L: SIGNED; R: SIGNED) return SIGNED;
   function "-"(L: UNSIGNED; R: SIGNED) return SIGNED;
   function "-"(L: SIGNED; R: UNSIGNED) return SIGNED;
   function "-"(L: UNSIGNED; R: INTEGER) return UNSIGNED;
   function "-"(L: INTEGER; R: UNSIGNED) return UNSIGNED;
   function "-"(L: SIGNED; R: INTEGER) return SIGNED;
   function "-"(L: INTEGER; R: SIGNED) return SIGNED;
   function "-"(L: UNSIGNED; R: STD_ULOGIC) return UNSIGNED;
```

```
function "-"(L: STD_ULOGIC; R: UNSIGNED) return UNSIGNED;
function "-"(L: SIGNED; R: STD_ULOGIC) return SIGNED;
function "-"(L: STD_ULOGIC; R: SIGNED) return SIGNED;

function "-"(L: UNSIGNED; R: UNSIGNED) return STD_LOGIC_VECTOR;
function "-"(L: SIGNED; R: SIGNED) return STD_LOGIC_VECTOR;
function "-"(L: UNSIGNED; R: SIGNED) return STD_LOGIC_VECTOR;
function "-"(L: SIGNED; R: UNSIGNED) return STD_LOGIC_VECTOR;
function "-"(L: UNSIGNED; R: INTEGER) return STD_LOGIC_VECTOR;
function "-"(L: INTEGER; R: UNSIGNED) return STD_LOGIC_VECTOR;
function "-"(L: SIGNED; R: INTEGER) return STD_LOGIC_VECTOR;
function "-"(L: INTEGER; R: SIGNED) return STD_ LOGIC_VECTOR;
function "-"(L: UNSIGNED; R: STD_ULOGIC) return STD_LOGIC_VECTOR;
function "-"(L: STD_ULOGIC; R: UNSIGNED) return STD_LOGIC_VECTOR;
function "-"(L: SIGNED; R: STD_ULOGIC) return STD_LOGIC_VECTOR;
function "-"(L: STD_ULOGIC; R: SIGNED) return STD_LOGIC_VECTOR;

function "+"(L: UNSIGNED) return UNSIGNED;
function "+"(L: SIGNED) return SIGNED;
function "-"(L: SIGNED) return SIGNED;
function "ABS"(L: SIGNED) return SIGNED;

function "+"(L: UNSIGNED) return STD_LOGIC_VECTOR;
function "+"(L: SIGNED) return STD_LOGIC_VECTOR;
function "-"(L: SIGNED) return STD_LOGIC_VECTOR;
function "ABS"(L: SIGNED) return STD_LOGIC_VECTOR;

function "*"(L: UNSIGNED; R: UNSIGNED) return UNSIGNED;
function "*"(L: SIGNED; R: SIGNED) return SIGNED;
function "*"(L: SIGNED; R: UNSIGNED) return SIGNED;
function "*"(L: UNSIGNED; R: SIGNED) return SIGNED;

function "*"(L: UNSIGNED; R: UNSIGNED) return STD_LOGIC_VECTOR;
function "*"(L: SIGNED; R: SIGNED) return STD_LOGIC_VECTOR;
function "*"(L: SIGNED; R: UNSIGNED) return STD_LOGIC_VECTOR;
function "*"(L: UNSIGNED; R: SIGNED) return STD_LOGIC_VECTOR;

function "<"(L: UNSIGNED; R: UNSIGNED) return BOOLEAN;
function "<"(L: SIGNED; R: SIGNED) return BOOLEAN;
function "<"(L: UNSIGNED; R: SIGNED) return BOOLEAN;
function "<"(L: SIGNED; R: UNSIGNED) return BOOLEAN;
function "<"(L: UNSIGNED; R: INTEGER) return BOOLEAN;
function "<"(L: INTEGER; R: UNSIGNED) return BOOLEAN;
function "<"(L: SIGNED; R: INTEGER) return BOOLEAN;
function "<"(L: INTEGER; R: SIGNED) return BOOLEAN;

function "<="(L: UNSIGNED; R: UNSIGNED) return BOOLEAN;
function "<="(L: SIGNED; R: SIGNED) return BOOLEAN;
function "<="(L: UNSIGNED; R: SIGNED) return BOOLEAN;
function "<="(L: SIGNED; R: UNSIGNED) return BOOLEAN;
function "<="(L: UNSIGNED; R: INTEGER) return BOOLEAN;
function "<="(L: INTEGER; R: UNSIGNED) return BOOLEAN;
function "<="(L: SIGNED; R: INTEGER) return BOOLEAN;
function "<="(L: INTEGER; R: SIGNED) return BOOLEAN;

function ">"(L: UNSIGNED; R: UNSIGNED) return BOOLEAN;
```

```
function ">"(L: SIGNED; R: SIGNED) return BOOLEAN;
function ">"(L: UNSIGNED; R: SIGNED) return BOOLEAN;
function ">"(L: SIGNED; R: UNSIGNED) return BOOLEAN;
function ">"(L: UNSIGNED; R: INTEGER) return BOOLEAN;
function ">"(L: INTEGER; R: UNSIGNED) return BOOLEAN;
function ">"(L: SIGNED; R: INTEGER) return BOOLEAN;
function ">"(L: INTEGER; R: SIGNED) return BOOLEAN;

function ">="(L: UNSIGNED; R: UNSIGNED) return BOOLEAN;
function ">="(L: SIGNED; R: SIGNED) return BOOLEAN;
function ">="(L: UNSIGNED; R: SIGNED) return BOOLEAN;
function ">="(L: SIGNED; R: UNSIGNED) return BOOLEAN;
function ">="(L: UNSIGNED; R: INTEGER) return BOOLEAN;
function ">="(L: INTEGER; R: UNSIGNED) return BOOLEAN;
function ">="(L: SIGNED; R: INTEGER) return BOOLEAN;
function ">="(L: INTEGER; R: SIGNED) return BOOLEAN;

function "="(L: UNSIGNED; R: UNSIGNED) return BOOLEAN;
function "="(L: SIGNED; R: SIGNED) return BOOLEAN;
function "="(L: UNSIGNED; R: SIGNED) return BOOLEAN;
function "="(L: SIGNED; R: UNSIGNED) return BOOLEAN;
function "="(L: UNSIGNED; R: INTEGER) return BOOLEAN;
function "="(L: INTEGER; R: UNSIGNED) return BOOLEAN;
function "="(L: SIGNED; R: INTEGER) return BOOLEAN;
function "="(L: INTEGER; R: SIGNED) return BOOLEAN;

function "/="(L: UNSIGNED; R: UNSIGNED) return BOOLEAN;
function "/="(L: SIGNED; R: SIGNED) return BOOLEAN;
function "/="(L: UNSIGNED; R: SIGNED) return BOOLEAN;
function "/="(L: SIGNED; R: UNSIGNED) return BOOLEAN;
function "/="(L: UNSIGNED; R: INTEGER) return BOOLEAN;
function "/="(L: INTEGER; R: UNSIGNED) return BOOLEAN;
function "/="(L: SIGNED; R: INTEGER) return BOOLEAN;
function "/="(L: INTEGER; R: SIGNED) return BOOLEAN;

function SHL(ARG: UNSIGNED; COUNT: UNSIGNED) return UNSIGNED;
function SHL(ARG: SIGNED; COUNT: UNSIGNED) return SIGNED;
function SHR(ARG: UNSIGNED; COUNT: UNSIGNED) return UNSIGNED;
function SHR(ARG: SIGNED; COUNT: UNSIGNED) return SIGNED;

function CONV_INTEGER(ARG: INTEGER) return INTEGER;
function CONV_INTEGER(ARG: UNSIGNED) return INTEGER;
function CONV_INTEGER(ARG: SIGNED) return INTEGER;
function CONV_INTEGER(ARG: STD_ULOGIC) return SMALL_INT;

function CONV_UNSIGNED(ARG: INTEGER; SIZE: INTEGER) return UNSIGNED;
function CONV_UNSIGNED(ARG: UNSIGNED; SIZE: INTEGER) return UNSIGNED;
function CONV_UNSIGNED(ARG: SIGNED; SIZE: INTEGER) return UNSIGNED;
function CONV_UNSIGNED(ARG: STD_ULOGIC; SIZE: INTEGER) return UNSIGNED;

function CONV_SIGNED(ARG: INTEGER; SIZE: INTEGER) return SIGNED;
function CONV_SIGNED(ARG: UNSIGNED; SIZE: INTEGER) return SIGNED;
function CONV_SIGNED(ARG: SIGNED; SIZE: INTEGER) return SIGNED;
function CONV_SIGNED(ARG: STD_ULOGIC; SIZE: INTEGER) return SIGNED;

function CONV_STD_LOGIC_VECTOR(ARG: INTEGER; SIZE: INTEGER)
```

```
                                                    return STD_LOGIC_VECTOR;
    function CONV_STD_LOGIC_VECTOR(ARG: UNSIGNED; SIZE: INTEGER)
                                                    return STD_LOGIC_VECTOR;
    function CONV_STD_LOGIC_VECTOR(ARG: SIGNED; SIZE: INTEGER)
                                                    return STD_LOGIC_VECTOR;
    function CONV_STD_LOGIC_VECTOR(ARG: STD_ULOGIC; SIZE: INTEGER)
                                                    return STD_LOGIC_VECTOR;
    -- zero extend STD_LOGIC_VECTOR (ARG) to SIZE,
    -- SIZE < 0 is same as SIZE = 0
    -- returns STD_LOGIC_VECTOR(SIZE-1 downto 0)
    function EXT(ARG: STD_LOGIC_VECTOR; SIZE: INTEGER) return
STD_LOGIC_ VECTOR;

    -- sign extend STD_LOGIC_VECTOR (ARG) to SIZE,
    -- SIZE < 0 is same as SIZE = 0
    -- return STD_LOGIC_VECTOR(SIZE-1 downto 0)
    function SXT(ARG: STD_LOGIC_VECTOR; SIZE: INTEGER) return
STD_LOGIC_VECTOR;

end Std_logic_arith;
```

2.4. Package Std_logic_unsigned

```
------------------------------------------------------------------------
--                                                                    --
-- Copyright (c) 1990, 1991, 1992 by Synopsys, Inc.                   --
--                         All rights reserved.                       --
--                                                                    --
-- This source file may be used and distributed without restriction   --
-- provided that this copyright statement is not removed from the file --
-- and that any derivative work contains this copyright notice.       --
--                                                                    --
--     Package name: STD_LOGIC_UNSIGNED                               --
--                                                                    --
--                                                                    --
--     Date:           09/11/92 KN                                    --
--                     10/08/92        AMT                            --
--                                                                    --
--     Purpose:                                                       --
--      A set of unsigned arithemtic, conversion,                     --
--        and comparision functions for STD_LOGIC_VECTOR.             --
--                                                                    --
--     Note:  comparision of same length discrete arrays is defined   --
--            by the LRM.  This package will "overload" those         --
--            definitions                                             --
--                                                                    --
------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
```

```
    package STD_LOGIC_UNSIGNED is

        function "+"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR;
        function "+"(L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
        function "+"(L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
        function "+"(L: STD_LOGIC_VECTOR; R: STD_LOGIC) return STD_LOGIC_VECTOR;
        function "+"(L: STD_LOGIC; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;

        function "-"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR;
        function "-"(L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
        function "-"(L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
        function "-"(L: STD_LOGIC_VECTOR; R: STD_LOGIC) return STD_LOGIC_VECTOR;
        function "-"(L: STD_LOGIC; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;

        function "+"(L: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;

        function "*"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR;

        function "<"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function "<"(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function "<"(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;

        function "<="(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function "<="(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function "<="(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;

        function ">"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function ">"(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function ">"(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;

        function ">="(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function ">="(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function ">="(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;

        function "="(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function "="(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function "="(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;

        function "/="(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function "/="(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function "/="(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function SHL(ARG:STD_LOGIC_VECTOR;COUNT: STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR;
        function SHR(ARG:STD_LOGIC_VECTOR;COUNT: STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR;

        function CONV_INTEGER(ARG: STD_LOGIC_VECTOR) return INTEGER;
    -- remove this since it is already in std_logic_arith
    --    function CONV_STD_LOGIC_VECTOR(ARG: INTEGER; SIZE: INTEGER) return
STD_LOGIC_VECTOR;

    end STD_LOGIC_UNSIGNED;
```

2.5. Package Std_logic_signed

```
-------------------------------------------------------------------------
--                                                                     --
-- Copyright (c) 1990, 1991, 1992 by Synopsys, Inc.                    --
--                         All rights reserved.                        --
--                                                                     --
-- This source file may be used and distributed without restriction    --
-- provided that this copyright statement is not removed from the file --
-- and that any derivative work contains this copyright notice.        --
--                                                                     --
--     Package name: STD_LOGIC_SIGNED                                  --
--                                                                     --
--                                                                     --
--     Date:       09/11/91 KN                                         --
--                 10/08/92 AMT change std_ulogic to signed std_logic  --
--                    10/28/92 AMT added signed functions, -, ABS      --
--                                                                     --
--     Purpose:                                                        --
--      A set of signed arithemtic, conversion,                        --
--        and comparision functions for STD_LOGIC_VECTOR.              --
--                                                                     --
--     Note:    Comparision of same length std_logic_vector is defined --
--              in the LRM.  The interpretation is for unsigned vectors --
--              This package will "overload" that definition.          --
--                                                                     --
-------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

package STD_LOGIC_SIGNED is

    function "+"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR;
    function "+"(L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
    function "+"(L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
    function "+"(L: STD_LOGIC_VECTOR; R: STD_LOGIC) return STD_LOGIC_VECTOR;
    function "+"(L: STD_LOGIC; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;

    function "-"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR;
    function "-"(L: STD_LOGIC_VECTOR; R: INTEGER) return STD_LOGIC_VECTOR;
    function "-"(L: INTEGER; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
    function "-"(L: STD_LOGIC_VECTOR; R: STD_LOGIC) return STD_LOGIC_VECTOR;
    function "-"(L: STD_LOGIC; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;

    function "+"(L: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
    function "-"(L: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;
    function "ABS"(L: STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR;


    function "*"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR;
```

```
        function "<"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function "<"(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function "<"(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;

        function "<="(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function "<="(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function "<="(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;

        function ">"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function ">"(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function ">"(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;

        function ">="(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function ">="(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function ">="(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;

        function "="(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function "="(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function "="(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;

        function "/="(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function "/="(L: STD_LOGIC_VECTOR; R: INTEGER) return BOOLEAN;
        function "/="(L: INTEGER; R: STD_LOGIC_VECTOR) return BOOLEAN;
        function SHL(ARG:STD_LOGIC_VECTOR;COUNT: STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR;
        function SHR(ARG:STD_LOGIC_VECTOR;COUNT: STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR;

        function CONV_INTEGER(ARG: STD_LOGIC_VECTOR) return INTEGER;

    -- remove this since it is already in std_logic_arith
    --    function CONV_STD_LOGIC_VECTOR(ARG: INTEGER; SIZE: INTEGER) return
STD_LOGIC_VECTOR;

    end STD_LOGIC_SIGNED;
```

2.6. Package Std_logic_misc

```
    ------------------------------------------------------------------------
    --
    -- Copyright (c) 1990, 1991, 1992 by Synopsys, Inc.  All rights reserved.
    --
    -- This source file may be used and distributed without restriction
    -- provided that this copyright statement is not removed from the file
    -- and that any derivative work contains this copyright notice.
    --
    --      Package name: std_logic_misc
    --
    --      Purpose: This package defines supplemental types, subtypes,
    --                   constants, and functions for the Std_logic_1164 Package.
    --
    --      Author:  GWH
    --
    ------------------------------------------------------------------------
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
library SYNOPSYS;
use SYNOPSYS.attributes.all;


package std_logic_misc is

   -- output-strength types

   type STRENGTH is (strn_X01, strn_X0H, strn_XL1, strn_X0Z, strn_XZ1,
                     strn_WLH, strn_WLZ, strn_WZH, strn_W0H, strn_WL1);


--synopsys synthesis_off

   type MINOMAX is array (1 to 3) of TIME;


   ------------------------------------------------------------------
   --
   -- functions for mapping the STD_(U)LOGIC according to STRENGTH
   --
   ------------------------------------------------------------------

   function strength_map(input: STD_ULOGIC; strn: STRENGTH) return STD_LOGIC;

   function strength_map_z(input:STD_ULOGIC; strn:STRENGTH) return STD_LOGIC;

   ------------------------------------------------------------------
   --
   -- conversion functions for STD_ULOGIC_VECTOR and STD_LOGIC_VECTOR
   --
   ------------------------------------------------------------------

--synopsys synthesis_on
   function Drive (V: STD_ULOGIC_VECTOR) return STD_LOGIC_VECTOR;

   function Drive (V: STD_LOGIC_VECTOR) return STD_ULOGIC_VECTOR;
--synopsys synthesis_off

   attribute CLOSELY_RELATED_TCF of Drive: function is TRUE;

   ------------------------------------------------------------------
   --
   -- conversion functions for sensing various types
   -- (the second argument allows the user to specify the value to
   --  be returned when the network is undriven)
   --
   ---------------------------------------------------------------------

   function Sense (V: STD_ULOGIC; vZ, vU, vDC: STD_ULOGIC) return STD_LOGIC;

   function Sense (V: STD_ULOGIC_VECTOR; vZ, vU, vDC: STD_ULOGIC)
                                          return STD_LOGIC_VECTOR;
   function Sense (V: STD_ULOGIC_VECTOR; vZ, vU, vDC: STD_ULOGIC)
```

```
                                        return STD_ULOGIC_VECTOR;

    function Sense (V: STD_LOGIC_VECTOR; vZ, vU, vDC: STD_ULOGIC)
                                        return STD_LOGIC_VECTOR;
    function Sense (V: STD_LOGIC_VECTOR; vZ, vU, vDC: STD_ULOGIC)
                                        return STD_ULOGIC_VECTOR;

--synopsys synthesis_on


    ------------------------------------------------------------------
    --
    -- Function: STD_LOGIC_VECTORtoBIT_VECTOR
STD_ULOGIC_VECTORtoBIT_VECTOR
    --
    -- Purpose: Conversion fun. from STD_(U)LOGIC_VECTOR to BIT_VECTOR
    --
    -- Mapping:     0, L --> 0
    --              1, H --> 1
    --              X, W --> vX if Xflag is TRUE
    --              X, W --> 0  if Xflag is FALSE
    --              Z --> vZ if Zflag is TRUE
    --              Z --> 0  if Zflag is FALSE
    --              U --> vU if Uflag is TRUE
    --              U --> 0  if Uflag is FALSE
    --              - --> vDC if DCflag is TRUE
    --              - --> 0  if DCflag is FALSE
    --
    ------------------------------------------------------------------

    function STD_LOGIC_VECTORtoBIT_VECTOR (V: STD_LOGIC_VECTOR
--synopsys synthesis_off
        ; vX, vZ, vU, vDC: BIT := '0';
          Xflag, Zflag, Uflag, DCflag: BOOLEAN := FALSE
--synopsys synthesis_on
        ) return BIT_VECTOR;

    function STD_ULOGIC_VECTORtoBIT_VECTOR (V: STD_ULOGIC_VECTOR
--synopsys synthesis_off
        ; vX, vZ, vU, vDC: BIT := '0';
          Xflag, Zflag, Uflag, DCflag: BOOLEAN := FALSE
--synopsys synthesis_on
        ) return BIT_VECTOR;


    ------------------------------------------------------------------
    --
    -- Function: STD_ULOGICtoBIT
    --
    -- Purpose: Conversion function from STD_(U)LOGIC to BIT
    --
    -- Mapping:     0, L --> 0
    --              1, H --> 1
    --              X, W --> vX if Xflag is TRUE
    --              X, W --> 0  if Xflag is FALSE
    --              Z --> vZ if Zflag is TRUE
```

```
    --                      Z --> 0  if Zflag is FALSE
    --                      U --> vU if Uflag is TRUE
    --                      U --> 0  if Uflag is FALSE
    --                      - --> vDC if DCflag is TRUE
    --                      - --> 0  if DCflag is FALSE
    --
    ----------------------------------------------------------------------

    function STD_ULOGICtoBIT (V: STD_ULOGIC
--synopsys synthesis_off
        ; vX, vZ, vU, vDC: BIT := '0';
          Xflag, Zflag, Uflag, DCflag: BOOLEAN := FALSE
--synopsys synthesis_on
        ) return BIT;

        ----------------------------------------------------------------
        function AND_REDUCE(ARG: STD_LOGIC_VECTOR) return UX01;
        function NAND_REDUCE(ARG: STD_LOGIC_VECTOR) return UX01;
        function OR_REDUCE(ARG: STD_LOGIC_VECTOR) return UX01;
        function NOR_REDUCE(ARG: STD_LOGIC_VECTOR) return UX01;
        function XOR_REDUCE(ARG: STD_LOGIC_VECTOR) return UX01;
        function XNOR_REDUCE(ARG: STD_LOGIC_VECTOR) return UX01;

        function AND_REDUCE(ARG: STD_ULOGIC_VECTOR) return UX01;
        function NAND_REDUCE(ARG: STD_ULOGIC_VECTOR) return UX01;
        function OR_REDUCE(ARG: STD_ULOGIC_VECTOR) return UX01;
        function NOR_REDUCE(ARG: STD_ULOGIC_VECTOR) return UX01;
        function XOR_REDUCE(ARG: STD_ULOGIC_VECTOR) return UX01;
        function XNOR_REDUCE(ARG: STD_ULOGIC_VECTOR) return UX01;

--synopsys synthesis_off

        function fun_BUF3S(Input, Enable: UX01; Strn: STRENGTH) return STD_LOGIC;
        function fun_BUF3SL(Input, Enable: UX01; Strn: STRENGTH) return STD_LOGIC;
        function fun_MUX2x1(Input0, Input1, Sel: UX01) return UX01;

        function fun_MAJ23(Input0, Input1, Input2: UX01) return UX01;
        function fun_WiredX(Input0, Input1: std_ulogic) return STD_LOGIC;

--synopsys synthesis_on

end;
```