The IEEE packages are in /opt/cds/ldv41/tools.sun4v/inca/files/IEEE

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
                       );

    attribute ENUM_ENCODING of std_ulogic : type is "U D 0 1 Z D 0 1
D";

    ------------------------------------------------------------------
    -- 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;
    --synopsys translate_off
    attribute REFLEXIVE of resolved: function is TRUE;
    attribute RESULT_INITIAL_VALUE of resolved: function is
std_ulogic'POS('Z');
    --synopsys translate_on


    ------------------------------------------------------------------
    -- *** 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 "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 "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
    --synopsys synthesis_off
                      ; xmap : BIT := '0'
    --synopsys synthesis_on
                    ) RETURN BIT;

    FUNCTION To_bitvector ( s : std_logic_vector
    --synopsys synthesis_off
                      ; xmap : BIT := '0'
    --synopsys synthesis_on
                    ) RETURN BIT_VECTOR;

    FUNCTION To_bitvector ( s : std_ulogic_vector
    --synopsys synthesis_off
                      ; xmap : BIT := '0'
    --synopsys synthesis_on
                    ) 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
    ------------------------------------------------------------------
    --synopsys synthesis_off
    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;
    --synopsys synthesis_on

END std_logic_1164;


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;
```

```
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;


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;


library std; use std.standard.all;  -- needed for bootstrap mode

package TEXTIO is

  -- Type Definitions for Text I/O

  type LINE is access STRING; -- a line is a pointer to a STRING value

  type TEXT is file of STRING;      -- a file of variable-length ASCII
records
```

```
  type SIDE is (RIGHT, LEFT); -- for justifying output data within
fields

  subtype WIDTH is NATURAL;   -- for specifying widths of output fields

  -- Standard Text Files

--  file INPUT: TEXT is in      "STD_INPUT";
  file INPUT: TEXT open READ_MODE is "STD_INPUT";

--  file OUTPUT:        TEXT is out     "STD_OUTPUT";
  file OUTPUT: TEXT open WRITE_MODE is "STD_OUTPUT";

  -- Input Routines for Standard Types

  procedure READLINE (file F: TEXT; L: inout LINE);
--   procedure READLINE (file F: TEXT; L: out LINE);

  procedure READ (L: inout LINE; VALUE: out BIT; GOOD: out BOOLEAN);
  procedure READ (L: inout LINE; VALUE: out BIT);

  procedure READ (L: inout LINE; VALUE: out BIT_VECTOR; GOOD: out
BOOLEAN);
  procedure READ (L: inout LINE; VALUE: out BIT_VECTOR);

  procedure READ (L: inout LINE; VALUE: out BOOLEAN; GOOD: out
BOOLEAN);
  procedure READ (L: inout LINE; VALUE: out BOOLEAN);

  procedure READ (L: inout LINE; VALUE: out CHARACTER; GOOD: out
BOOLEAN);
  procedure READ (L: inout LINE; VALUE: out CHARACTER);

  procedure READ (L: inout LINE; VALUE: out INTEGER; GOOD: out
BOOLEAN);
  procedure READ (L: inout LINE; VALUE: out INTEGER);

  procedure READ (L: inout LINE; VALUE: out REAL; GOOD: out BOOLEAN);
  procedure READ (L: inout LINE; VALUE: out REAL);

  procedure READ (L: inout LINE; VALUE: out STRING; GOOD: out BOOLEAN);
  procedure READ (L: inout LINE; VALUE: out STRING);

  procedure READ (L: inout LINE; VALUE: out TIME; GOOD: out BOOLEAN);
  procedure READ (L: inout LINE; VALUE: out TIME);

  -- Output Routines for Standard Types

--  procedure WRITELINE (F: out TEXT; L: inout LINE);
  procedure WRITELINE (file F: TEXT; L: inout LINE);

  procedure WRITE (L: inout LINE; VALUE: in BIT;
                   JUSTIFIED: in SIDE := RIGHT; FIELD: in WIDTH := 0);

  procedure WRITE (L: inout LINE; VALUE: in BIT_VECTOR;
                   JUSTIFIED: in SIDE := RIGHT; FIELD: in WIDTH := 0);
```

```
  procedure WRITE (L: inout LINE; VALUE: in BOOLEAN;
                   JUSTIFIED: in SIDE := RIGHT; FIELD: in WIDTH := 0);

  procedure WRITE (L: inout LINE; VALUE: in CHARACTER;
                   JUSTIFIED: in SIDE := RIGHT; FIELD: in WIDTH := 0);

  procedure WRITE (L: inout LINE; VALUE: in INTEGER;
                   JUSTIFIED: in SIDE := RIGHT; FIELD: in WIDTH := 0);

  procedure WRITE (L: inout LINE; VALUE: in REAL;
                   JUSTIFIED: in SIDE := RIGHT; FIELD: in WIDTH := 0;
                   DIGITS: in NATURAL := 0);

  procedure WRITE (L: inout LINE; VALUE: in STRING;
                   JUSTIFIED: in SIDE := RIGHT; FIELD: in WIDTH := 0);

  procedure WRITE (L: inout LINE; VALUE: in TIME;
                   JUSTIFIED: in SIDE := RIGHT; FIELD: in WIDTH := 0;
                   UNIT: in TIME := ns);

  -- File Position Predicates

  function ENDLINE (L: in LINE) return BOOLEAN;

  -- function ENDFILE (F: in TEXT) return BOOLEAN;

end TEXTIO;


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;




library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_MISC.all;


package STD_LOGIC_COMPONENTS is
--synopsys translate_off

    ------------------------------------------------------------------
-----
    --
    -- Logic gates for n inputs
    --
    ------------------------------------------------------------------
-----

    component ANDGATE                       -- N input AND gate
      generic (N:   Positive := 2;          -- number of inputs
             tLH: Time := 0 ns;             -- rise inertial delay
             tHL: Time := 0 ns;             -- fall inertial delay
             strn: STRENGTH := strn_X01); -- output strength
      port (Input: in STD_LOGIC_VECTOR (1 to N);      -- input
```

```
        Output: out STD_LOGIC);       -- output
end component;

component NANDGATE                    -- N input NAND gate
  generic (N:   Positive := 2;        -- number of inputs
         tLH: Time := 0 ns;           -- rise inertial delay
         tHL: Time := 0 ns;           -- fall inertial delay
         strn: STRENGTH := strn_X01); -- output strength
  port (Input: in STD_LOGIC_VECTOR (1 to N);      -- input
        Output: out STD_LOGIC);       -- output
end component;

component ORGATE                      -- N input OR gate
  generic (N:   Positive := 2;        -- number of inputs
         tLH: Time := 0 ns;           -- rise inertial delay
         tHL: Time := 0 ns;           -- fall inertial delay
         strn: STRENGTH := strn_X01); -- output strength
  port (Input: in STD_LOGIC_VECTOR (1 to N);      -- input
        Output: out STD_LOGIC);       -- output
end component;

component NORGATE                     -- N input NOR gate
  generic (N:   Positive := 2;        -- number of inputs
         tLH: Time := 0 ns;           -- rise inertial delay
         tHL: Time := 0 ns;           -- fall inertial delay
         strn: STRENGTH := strn_X01); -- output strength
  port (Input: in STD_LOGIC_VECTOR (1 to N);      -- input
        Output: out STD_LOGIC);       -- output
end component;

component XORGATE                     -- N input XOR gate
  generic (N:   Positive := 2;        -- number of inputs
         tLH: Time := 0 ns;           -- rise inertial delay
         tHL: Time := 0 ns;           -- fall inertial delay
         strn: STRENGTH := strn_X01); -- output strength
  port (Input: in STD_LOGIC_VECTOR (1 to N);      -- input
        Output: out STD_LOGIC);       -- output
end component;

component XNORGATE                    -- N input XNOR gate
  generic (N:   Positive := 2;        -- number of inputs
         tLH: Time := 0 ns;           -- rise inertial delay
         tHL: Time := 0 ns;           -- fall inertial delay
         strn: STRENGTH := strn_X01); -- output strength
  port (Input: in STD_LOGIC_VECTOR (1 to N);      -- input
        Output: out STD_LOGIC);       -- output
end component;

component NXORGATE                    -- N input NXOR gate
  generic (N:   Positive := 2;        -- number of inputs
         tLH: Time := 0 ns;           -- rise inertial delay
         tHL: Time := 0 ns;           -- fall inertial delay
         strn: STRENGTH := strn_X01); -- output strength
  port (Input: in STD_LOGIC_VECTOR (1 to N);      -- input
        Output: out STD_LOGIC);       -- output
end component;
```

```
    ------------------------------------------------------------------
-------
    --
    -- Transfer gates
    --
    ------------------------------------------------------------------
------
    component BUFGATE                     -- buffer with INERTIAL delay
      generic (tLH: Time := 0 ns;         -- rise inertial delay
               tHL: Time := 0 ns;         -- fall inertial delay
               strn: STRENGTH := strn_X01); -- output strength
      port (Input: in STD_LOGIC;               -- input
            Output: out STD_LOGIC);       -- output
    end component;

    component WBUFGATE                    -- buffer with TRANSPORT
delay
      generic (tLH: Time := 0 ns;         -- rise transport delay
               tHL: Time := 0 ns;         -- fall transport delay
               strn: STRENGTH := strn_X01); -- output strength
      port (Input: in STD_LOGIC;               -- input
            Output: out STD_LOGIC);       -- output
    end component;

    component INVGATE                     -- inverter
      generic (tLH: Time := 0 ns;         -- rise inertial delay
               tHL: Time := 0 ns;         -- fall inertial delay
               strn: STRENGTH := strn_X01); -- output strength
      port (Input: in STD_LOGIC;               -- input
            Output: out STD_LOGIC);       -- output
    end component;


    ------------------------------------------------------------------
-------
    --
    -- Tristate transfer gates
    --
    ------------------------------------------------------------------
------
    component BUF3S                       -- tristate buffer
      generic (tLH: Time := 0 ns;         -- rise inertial delay
               tHL: Time := 0 ns;         -- fall inertial delay
               strn: STRENGTH := strn_X01); -- output strength
      port (Input: in STD_LOGIC;               -- input
            Enable: in STD_LOGIC;              -- enable
            Output: out STD_LOGIC);       -- output
    end component;

    component BUF3SL                      -- tristate buffer
      generic (tLH: Time := 0 ns;         -- rise inertial delay
               tHL: Time := 0 ns;         -- fall inertial delay
               strn: STRENGTH := strn_X01); -- output strength
      port (Input: in STD_LOGIC;               -- input
            Enable: in STD_LOGIC;              -- enable
            Output: out STD_LOGIC);       -- output
    end component;
```

```
    component INV3S                       -- tristate inverter
      generic (tLH: Time := 0 ns;         -- rise inertial delay
               tHL: Time := 0 ns;         -- fall inertial delay
               strn: STRENGTH := strn_X01); -- output strength
      port (Input: in STD_LOGIC;               -- input
            Enable: in STD_LOGIC;              -- enable
            Output: out STD_LOGIC);       -- output
    end component;

    component INV3SL                      -- tristate inverter
      generic (tLH: Time := 0 ns;         -- rise inertial delay
               tHL: Time := 0 ns;         -- fall inertial delay
               strn: STRENGTH := strn_X01); -- output strength
      port (Input: in STD_LOGIC;               -- input
            Enable: in STD_LOGIC;              -- enable
            Output: out STD_LOGIC);       -- output
    end component;

    ------------------------------------------------------------------
-------
    --
    -- Other logic components
    --
    ------------------------------------------------------------------
-------

    component MUX2x1                      -- 2 by 1 multiplexer
      generic (tLH: Time :=  0 ns;        -- rise inertial delay
               tHL: Time :=  0 ns;        -- fall inertial delay
               strn: STRENGTH := strn_X01); -- output strength
      port (In0,                     -- data input 0
            In1,                     -- data input 1
            Sel: in STD_LOGIC;                -- select input
(0=>In0)
            Output: out STD_LOGIC);       -- output
    end component;


    ------------------------------------------------------------------
------
    --
    -- State components
    --
    ------------------------------------------------------------------
------

    component DFFREG          -- edge triggered register, reset active
high
      generic (N: Positive :=1;               -- N bit I/O
               tLH: Time :=  0 ns;            -- rise inertial delay
               tHL: Time :=  0 ns;            -- fall inertial delay
               strn: STRENGTH := strn_X01;  -- output strength;
          tHOLD : Time := 0 ns;               -- Hold time
          tSetUp : Time := 0 ns;              -- Setup time
          tPwHighMin : Time := 0 ns;     -- min pulse width when clock
is high
```

```
         tPwLowMin : Time := 0 ns);     -- min pulse width when clock
is low
      port (Data: in STD_LOGIC_VECTOR (N-1 downto 0); -- data input
            Clock,                                          -- clock
input
            Reset: in STD_LOGIC;                       -- reset
input(act.high)
            Output: out STD_LOGIC_VECTOR (N-1 downto 0));   -- output
    end component;

    component DFFREGL           -- edge triggered register, reset
active low
      generic (N: Positive :=1;                 -- N bit I/O
             tLH: Time :=  0 ns;                -- rise inertial delay
             tHL: Time :=  0 ns;                -- fall inertial delay
             strn: STRENGTH := strn_X01;  -- output strength
         tHOLD : Time := 0 ns;                  -- Hold time
         tSetUp : Time := 0 ns;                 -- Setup time
         tPwHighMin : Time := 0 ns;     -- min pulse width when clock
is high
         tPwLowMin : Time := 0 ns);     -- min pulse width when clock
is low
      port (Data: in STD_LOGIC_VECTOR (N-1 downto 0); -- data input
            Clock,                              -- clock input
            Reset: in STD_LOGIC;                       -- reset
input(act. low)
            Output: out STD_LOGIC_VECTOR (N-1 downto 0));   -- output
    end component;

    component DLATREG         -- level sensitive register, reset active
high
      generic (N: Positive :=1;                 -- N bit I/O
             tLH: Time :=  0 ns;                -- rise inertial delay
             tHL: Time :=  0 ns;                -- fall inertial delay
             strn: STRENGTH := strn_X01;  -- output strength
         tHOLD : Time := 0 ns;                  -- Hold time
         tSetUp : Time := 0 ns;                 -- Setup time
         tPwHighMin : Time := 0 ns;     -- min pulse width when enable
is high
         tPwLowMin : Time := 0 ns);     -- min pulse width when enable
is low
      port (Data: in STD_LOGIC_VECTOR (N-1 downto 0); -- data input
            Enable,                             -- enable input
            Reset: in STD_LOGIC;                       -- reset
input(act.high)
            Output: out STD_LOGIC_VECTOR (N-1 downto 0));   -- output
    end component;

    component DLATREGL    -- level sensitive register, reset active low
      generic (N: Positive :=1;                 -- N bit I/O
             tLH: Time :=  0 ns;                -- rise inertial delay
             tHL: Time :=  0 ns;                -- fall inertial delay
             strn: STRENGTH := strn_X01;  -- output strength
         tHOLD : Time := 0 ns;                  -- Hold time
         tSetUp : Time := 0 ns;                 -- Setup time
         tPwHighMin : Time := 0 ns;     -- min pulse width when enable
is high
```

```
         tPwLowMin : Time := 0 ns);     -- min pulse width when enable
is low
      port (Data: in STD_LOGIC_VECTOR (N-1 downto 0); -- data input
            Enable,                              -- enable input
            Reset: in STD_LOGIC;                      -- reset
input(act. low)
            Output: out STD_LOGIC_VECTOR (N-1 downto 0));   -- output
    end component;

    component DFFREGSRH    -- edge triggered register, reset & set
active high
      generic (N: Positive :=1;                 -- N bit register
             tLH: Time :=  0 ns;                -- rise inertial delay
             tHL: Time :=  0 ns;                -- fall inertial delay
             strn: STRENGTH := strn_X01;  -- output strength
         tHOLD : Time := 0 ns;                  -- Hold time
         tSetUp : Time := 0 ns;                 -- Setup time
         tPwHighMin : Time := 0 ns;     -- min pulse width when clock
is high
         tPwLowMin : Time := 0 ns);     -- min pulse width when clock
is low
      port (Data: in STD_LOGIC_VECTOR (N-1 downto 0); -- data input
            Clock: in STD_LOGIC;                    -- clock input
            Reset, Set: in STD_LOGIC := '0';          -- reset, set
input
            Output: out STD_LOGIC_VECTOR (N-1 downto 0));   -- output
    end component;

    component  DFFREGSRL   -- edge triggered register, reset & set
active low
      generic (N: Positive :=1;                 -- N bit register
             tLH: Time :=  0 ns;                -- rise inertial delay
             tHL: Time :=  0 ns;                -- fall inertial delay
             strn: STRENGTH := strn_X01;  -- output strength
         tHOLD : Time := 0 ns;                  -- Hold time
         tSetUp : Time := 0 ns;                 -- Setup time
         tPwHighMin : Time := 0 ns;     -- min pulse width when clock
is high
         tPwLowMin : Time := 0 ns);     -- min pulse width when clock
is low
      port (Data: in STD_LOGIC_VECTOR (N-1 downto 0); -- data input
            Clock: in STD_LOGIC;                    -- clock input
            Reset, Set: in STD_LOGIC := '1';          -- reset, set
input
            Output: out STD_LOGIC_VECTOR (N-1 downto 0));   -- output
    end component;

    component  DLATREGSRH   -- level sensitive register, reset & set
active high
      generic (N: Positive :=1;                 -- N bit register
             tLH: Time :=  0 ns;                -- rise inertial delay
             tHL: Time :=  0 ns;                -- fall inertial delay
             strn: STRENGTH := strn_X01;  -- output strength
         tHOLD : Time := 0 ns;                  -- Hold time
         tSetUp : Time := 0 ns;                 -- Setup time
         tPwHighMin : Time := 0 ns;     -- min pulse width when enable
is high
```

```
         tPwLowMin : Time := 0 ns);     -- min pulse width when enable
is low
      port (Data: in STD_LOGIC_VECTOR (N-1 downto 0); -- data input
            Enable: in STD_LOGIC;                     -- clock input
            Reset, Set: in STD_LOGIC := '0';          -- reset, set
input
            Output: out STD_LOGIC_VECTOR (N-1 downto 0));   -- output
    end component;

    component  DLATREGSRL   -- level sensitive register, reset & set
active low
      generic (N: Positive :=1;                 -- N bit register
             tLH: Time :=  0 ns;                -- rise inertial delay
             tHL: Time :=  0 ns;                -- fall inertial delay
             strn: STRENGTH := strn_X01;  -- output strength
         tHOLD : Time := 0 ns;                  -- Hold time
         tSetUp : Time := 0 ns;                 -- Setup time
         tPwHighMin : Time := 0 ns;     -- min pulse width when enable
is high
         tPwLowMin : Time := 0 ns);     -- min pulse width when enable
is low
      port (Data: in STD_LOGIC_VECTOR (N-1 downto 0); -- data input
            Enable: in STD_LOGIC;                     -- clock input
            Reset, Set: in STD_LOGIC := '1';          -- reset, set
input
            Output: out STD_LOGIC_VECTOR (N-1 downto 0));   -- output
    end component;

    ----------------------------------------------------------------
------
    --
    -- Memory component
    --
    ----------------------------------------------------------------
------

    component DLATRAM     -- level sensitive ram
      generic (Ndata: Positive := 1;                    -- # of data
I/O lines
             Naddr: Positive := 1;               -- # of address lines
             tLH: time := 0 ns;                  -- output rising
delay
             tHL: time := 0 ns;                  -- output fulling
delay
             strn: STRENGTH := strn_X01);        -- output strength
      port (DATAin: in STD_LOGIC_VECTOR(Ndata-1 downto 0);    -- data
in lines
            DATAout: out STD_LOGIC_VECTOR(Ndata-1 downto 0); -- data
out lines
            ADDR: in STD_LOGIC_VECTOR(Naddr-1 downto 0);      --
address lines
            WE: in STD_LOGIC;                   -- write
enable(active high)
            RE: in STD_LOGIC);                        -- read
enable(active high)

    end component;
```

```
    component DLATROM     -- level sensitive rom
      generic (Ndata: Positive := 1;                    -- # of data
I/O lines
               Naddr: Positive := 1;                -- # of address lines
               tLH: time := 0 ns;                   -- output rising
delay
               tHL: time := 0 ns;                   -- output fulling
delay
               strn: STRENGTH := strn_X01);         -- output strength
      port (DATA: out STD_LOGIC_VECTOR(Ndata-1 downto 0);     -- data
out lines
              ADDR: in STD_LOGIC_VECTOR(Naddr-1 downto 0);      --
address lines
              RE: in STD_LOGIC);                            -- read
enable(active high)

    end component;

    ---------------------------------------------------------------------
------
    --
    -- Timing Checker components
    --
    ---------------------------------------------------------------------
------

    component SUHDCK                        -- setup/hold timing checker
      generic (N: Positive := 1;            -- number of data lines
             tSetup,                 -- setup time
             tHold: Time := 0 ns);          -- hold time
      port (Data:  in STD_LOGIC_VECTOR (1 to N);      -- data lines
            Clock: in STD_LOGIC);               -- clock line
    end component;

    component MPWCK                         -- minimum pulse width
checker
      generic (tHigh,                       -- pulse's high time
             tLow: Time := 0 ns);           -- pulse's low time
      port (Clock: in STD_LOGIC);               -- clock line
    end component;

    component RECOVCK                       -- recovery timing checker
      generic (tSetup,                -- setup time
             tHold: Time := 0 ns);          -- hold time
      port (Reset,                          -- Reset line
            Clock: in STD_LOGIC);               -- clock line
    end component;


    ---------------------------------------------------------------------
------
    --
    -- Unidirectional Transistors.
    --
    ---------------------------------------------------------------------
------
```

```
    component NXFERGATE
        generic (tLH, tHL: time := 0 ns);
        port (Input, Enable: in STD_LOGIC; output: Out STD_LOGIC);
    end component;

    component NRXFERGATE
        generic (tLH, tHL: time := 0 ns);
        port (Input, Enable: in STD_LOGIC; output: Out STD_LOGIC);
    end component;

    component PXFERGATE
        generic (tLH, tHL: time := 0 ns);
        port (Input, Enable: in STD_LOGIC; output: Out STD_LOGIC);
    end component;

    component PRXFERGATE
        generic (tLH, tHL: time := 0 ns);
        port (Input, Enable: in STD_LOGIC; output: Out STD_LOGIC);
    end component;

    ------------------------------------------------------------------
------
    --
    -- Resistor Primitive.
    --
    ------------------------------------------------------------------
------
    component RESISTOR
        generic (tLH, tHL: time := 0 ns);
        port (Input: in STD_LOGIC; output: Out STD_LOGIC);
    end component;

--synopsys translate_on
end STD_LOGIC_COMPONENTS;


package NUMERIC_BIT is
  constant CopyRightNotice: STRING
      := "Copyright 1995 IEEE. All rights reserved.";

  --
==========================================================================
=====
  -- Numeric array type definitions
  --
==========================================================================
=====

  type UNSIGNED is array (NATURAL range <> ) of BIT;
  type SIGNED is array (NATURAL range <> ) of BIT;

  --
==========================================================================
=====
  -- Arithmetic Operators:
```

```
  --
================================================================================

  -- Id: A.1
  function "abs" (ARG: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0).
  -- Result: Returns the absolute value of a SIGNED vector ARG.

  -- Id: A.2
  function "-" (ARG: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0).
  -- Result: Returns the value of the unary minus operation on a
  --         SIGNED vector ARG.

  --
================================================================================

  -- Id: A.3
  function "+" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
  -- Result: Adds two UNSIGNED vectors that may be of different
lengths.

  -- Id: A.4
  function "+" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
  -- Result: Adds two SIGNED vectors that may be of different lengths.

  -- Id: A.5
  function "+" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0).
  -- Result: Adds an UNSIGNED vector, L, with a non-negative INTEGER,
R.

  -- Id: A.6
  function "+" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0).
  -- Result: Adds a non-negative INTEGER, L, with an UNSIGNED vector,
R.

  -- Id: A.7
  function "+" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0).
  -- Result: Adds an INTEGER, L(may be positive or negative), to a
SIGNED
  -- vector, R.

  -- Id: A.8
  function "+" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0).
  -- Result: Adds a SIGNED vector, L, to an INTEGER, R.

  --
================================================================================
```

```
  -- Id: A.9
  function "-" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
  -- Result: Subtracts two UNSIGNED vectors that may be of different
lengths.

  -- Id: A.10
  function "-" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
  -- Result: Subtracts a SIGNED vector, R, from another SIGNED vector,
L,
  --         that may possibly be of different lengths.

  -- Id: A.11
  function "-" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0).
  -- Result: Subtracts a non-negative INTEGER, R, from an UNSIGNED
vector, L.

  -- Id: A.12
  function "-" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0).
  -- Result: Subtracts an UNSIGNED vector, R, from a non-negative
INTEGER, L.

  -- Id: A.13
  function "-" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0).
  -- Result: Subtracts an INTEGER, R, from a SIGNED vector, L.

  -- Id: A.14
  function "-" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0).
  -- Result: Subtracts a SIGNED vector, R, from an INTEGER, L.

  --
==========================================================================
=====

  -- Id: A.15
  function "*" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED((L'LENGTH+R'LENGTH-1) downto 0).
  -- Result: Performs the multiplication operation on two UNSIGNED
vectors
  --         that may possibly be of different lengths.

  -- Id: A.16
  function "*" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED((L'LENGTH+R'LENGTH-1) downto 0)
  -- Result: Multiplies two SIGNED vectors that may possibly be of
  --         different lengths.

  -- Id: A.17
  function "*" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED((L'LENGTH+L'LENGTH-1) downto 0).
  -- Result: Multiplies an UNSIGNED vector, L, with a non-negative
```

```
  --         INTEGER, R. R is converted to an UNSIGNED vector of
  --         size L'LENGTH before multiplication.

  -- Id: A.18
  function "*" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED((R'LENGTH+R'LENGTH-1) downto 0).
  -- Result: Multiplies an UNSIGNED vector, R, with a non-negative
  --         INTEGER, L. L is converted to an UNSIGNED vector of
  --         size R'LENGTH before multiplication.

  -- Id: A.19
  function "*" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED((L'LENGTH+L'LENGTH-1) downto 0)
  -- Result: Multiplies a SIGNED vector, L, with an INTEGER, R. R is
  --         converted to a SIGNED vector of size L'LENGTH before
  --         multiplication.

  -- Id: A.20
  function "*" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED((R'LENGTH+R'LENGTH-1) downto 0)
  -- Result: Multiplies a SIGNED vector, R, with an INTEGER, L. L is
  --         converted to a SIGNED vector of size R'LENGTH before
  --         multiplication.

  --
================================================================================
  --
  -- NOTE: If second argument is zero for "/" operator, a severity
level
  --       of ERROR is issued.

  -- Id: A.21
  function "/" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Divides an UNSIGNED vector, L, by another UNSIGNED vector,
R.

  -- Id: A.22
  function "/" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Divides an SIGNED vector, L, by another SIGNED vector, R.

  -- Id: A.23
  function "/" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Divides an UNSIGNED vector, L, by a non-negative INTEGER,
R.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.24
  function "/" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0)
  -- Result: Divides a non-negative INTEGER, L, by an UNSIGNED vector,
R.
```

```
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.

  -- Id: A.25
  function "/" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Divides a SIGNED vector, L, by an INTEGER, R.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.26
  function "/" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0)
  -- Result: Divides an INTEGER, L, by a SIGNED vector, R.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.

  --
================================================================================
=====
  --
  -- NOTE: If second argument is zero for "rem" operator, a severity
level
  --       of ERROR is issued.

  -- Id: A.27
  function "rem" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where L and R are UNSIGNED vectors.

  -- Id: A.28
  function "rem" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where L and R are SIGNED vectors.

  -- Id: A.29
  function "rem" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where L is an UNSIGNED vector and R is
a
  --         non-negative INTEGER.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.30
  function "rem" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where R is an UNSIGNED vector and L is
a
  --         non-negative INTEGER.
  -- If NO_OF_BITS(L) > R'LENGTH, result is truncated to R'LENGTH.

  -- Id: A.31
  function "rem" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where L is SIGNED vector and R is an
INTEGER.
```

```
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.32
  function "rem" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where R is SIGNED vector and L is an
INTEGER.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.

  --
================================================================================
  --
  -- NOTE: If second argument is zero for "mod" operator, a severity
level
  --       of ERROR is issued.

  -- Id: A.33
  function "mod" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L and R are UNSIGNED vectors.

  -- Id: A.34
  function "mod" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L and R are SIGNED vectors.

  -- Id: A.35
  function "mod" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L is an UNSIGNED vector and R
  --         is a non-negative INTEGER.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.36
  function "mod" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where R is an UNSIGNED vector and L
  --         is a non-negative INTEGER.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.

  -- Id: A.37
  function "mod" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L is a SIGNED vector and
  --         R is an INTEGER.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.38
  function "mod" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L is an INTEGER and
```

```
  --         R is a SIGNED vector.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.

  --
=============================================================================
=====
  -- Comparison Operators
  --
=============================================================================
=====

  -- Id: C.1
  function ">" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L and R are UNSIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.2
  function ">" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L and R are SIGNED vectors possibly
  --         of different lengths.

  -- Id: C.3
  function ">" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.4
  function ">" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L is a INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.5
  function ">" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.6
  function ">" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L is a SIGNED vector and
  --         R is a INTEGER.

  --
=============================================================================
=====

  -- Id: C.7
  function "<" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
```

```
  -- Result: Computes "L < R" where L and R are UNSIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.8
  function "<" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L and R are SIGNED vectors possibly
  --         of different lengths.

  -- Id: C.9
  function "<" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.10
  function "<" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L is an INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.11
  function "<" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.12
  function "<" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L is a SIGNED vector and
  --         R is an INTEGER.

  --
==========================================================================
=====

  -- Id: C.13
  function "<=" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L and R are UNSIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.14
  function "<=" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L and R are SIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.15
  function "<=" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.
```

```
  -- Id: C.16
  function "<=" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L is an INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.17
  function "<=" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.18
  function "<=" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L is a SIGNED vector and
  --         R is an INTEGER.

  --
================================================================================

  -- Id: C.19
  function ">=" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L and R are UNSIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.20
  function ">=" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L and R are SIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.21
  function ">=" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.22
  function ">=" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L is an INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.23
  function ">=" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.24
  function ">=" (L: SIGNED; R: INTEGER) return BOOLEAN;
```

```
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L is a SIGNED vector and
  --         R is an INTEGER.

  --
================================================================================

  -- Id: C.25
  function "=" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L and R are UNSIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.26
  function "=" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L and R are SIGNED vectors possibly
  --         of different lengths.

  -- Id: C.27
  function "=" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.28
  function "=" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L is an INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.29
  function "=" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.30
  function "=" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L is a SIGNED vector and
  --         R is an INTEGER.

  --
================================================================================

  -- Id: C.31
  function "/=" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L and R are UNSIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.32
```

```
  function "/=" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L and R are SIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.33
  function "/=" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.34
  function "/=" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L is an INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.35
  function "/=" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.36
  function "/=" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L is a SIGNED vector and
  --         R is an INTEGER.

  --
================================================================================
=====
  -- Shift and Rotate Functions
  --
================================================================================
=====

  -- Id: S.1
  function SHIFT_LEFT (ARG: UNSIGNED; COUNT: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a shift-left on an UNSIGNED vector COUNT times.
  --         The vacated positions are filled with Bit '0'.
  --         The COUNT leftmost bits are lost.

  -- Id: S.2
  function SHIFT_RIGHT (ARG: UNSIGNED; COUNT: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a shift-right on an UNSIGNED vector COUNT times.
  --         The vacated positions are filled with Bit '0'.
  --         The COUNT rightmost bits are lost.

  -- Id: S.3
  function SHIFT_LEFT (ARG: SIGNED; COUNT: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a shift-left on a SIGNED vector COUNT times.
  --         The vacated positions are filled with Bit '0'.
```

```
  --         The COUNT leftmost bits, except ARG'LEFT, are lost.

  -- Id: S.4
  function SHIFT_RIGHT (ARG: SIGNED; COUNT: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a shift-right on a SIGNED vector COUNT times.
  --         The vacated positions are filled with the leftmost bit,
ARG'LEFT.
  --         The COUNT rightmost bits are lost.

  --
================================================================================
=====

  -- Id: S.5
  function ROTATE_LEFT (ARG: UNSIGNED; COUNT: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a rotate-left of an UNSIGNED vector COUNT times.

  -- Id: S.6
  function ROTATE_RIGHT (ARG: UNSIGNED; COUNT: NATURAL) return
UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a rotate-right of an UNSIGNED vector COUNT times.

  -- Id: S.7
  function ROTATE_LEFT (ARG: SIGNED; COUNT: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a logical rotate-left of a SIGNED vector COUNT
times.

  -- Id: S.8
  function ROTATE_RIGHT (ARG: SIGNED; COUNT: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a logical rotate-right of a SIGNED vector COUNT
times.

  --
================================================================================
=====

  ------------------------------------------------------------------------
---------
  -- Note : Function S.9 is not compatible with VHDL 1076-1987. Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------------
---------
  -- Id: S.9
  function "sll" (ARG: UNSIGNED; COUNT: INTEGER) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: SHIFT_LEFT(ARG, COUNT)

  ------------------------------------------------------------------------
---------
  -- Note : Function S.10 is not compatible with VHDL 1076-1987.
Comment
```

```
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------
---------
  -- Id: S.10
  function "sll" (ARG: SIGNED; COUNT: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: SHIFT_LEFT(ARG, COUNT)

  ------------------------------------------------------------------
---------
  -- Note : Function S.11 is not compatible with VHDL 1076-1987.
Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------
---------
  -- Id: S.11
  function "srl" (ARG: UNSIGNED; COUNT: INTEGER) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: SHIFT_RIGHT(ARG, COUNT)

  ------------------------------------------------------------------
---------
  -- Note : Function S.12 is not compatible with VHDL 1076-1987.
Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------
---------
  -- Id: S.12
  function "srl" (ARG: SIGNED; COUNT: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: SIGNED(SHIFT_RIGHT(UNSIGNED(ARG), COUNT))

  ------------------------------------------------------------------
---------
  -- Note : Function S.13 is not compatible with VHDL 1076-1987.
Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------
---------
  -- Id: S.13
  function "rol" (ARG: UNSIGNED; COUNT: INTEGER) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: ROTATE_LEFT(ARG, COUNT)

  ------------------------------------------------------------------
---------
  -- Note : Function S.14 is not compatible with VHDL 1076-1987.
Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------
---------
  -- Id: S.14
```

```
  function "rol" (ARG: SIGNED; COUNT: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: ROTATE_LEFT(ARG, COUNT)

  ------------------------------------------------------------------------
---------
  -- Note : Function S.15 is not compatible with VHDL 1076-1987.
Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------------
---------
  -- Id: S.15
  function "ror" (ARG: UNSIGNED; COUNT: INTEGER) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: ROTATE_RIGHT(ARG, COUNT)

  ------------------------------------------------------------------------
---------
  -- Note : Function S.16 is not compatible with VHDL 1076-1987.
Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------------
---------
  -- Id: S.16
  function "ror" (ARG: SIGNED; COUNT: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: ROTATE_RIGHT(ARG, COUNT)

  --
========================================================================
=====
  -- RESIZE Functions
  --
========================================================================
=====

  -- Id: R.1
  function RESIZE (ARG: SIGNED; NEW_SIZE: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(NEW_SIZE-1 downto 0)
  -- Result: Resizes the SIGNED vector ARG to the specified size.
  --         To create a larger vector, the new [leftmost] bit
positions
  --         are filled with the sign bit (ARG'LEFT). When truncating,
  --         the sign bit is retained along with the rightmost part.

  -- Id: R.2
  function RESIZE (ARG: UNSIGNED; NEW_SIZE: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(NEW_SIZE-1 downto 0)
  -- Result: Resizes the UNSIGNED vector ARG to the specified size.
  --         To create a larger vector, the new [leftmost] bit
positions
  --         are filled with '0'. When truncating, the leftmost bits
  --         are dropped.
```

```
  --
================================================================================
=====
  -- Conversion Functions
  --
================================================================================
=====

  -- Id: D.1
  function TO_INTEGER (ARG: UNSIGNED) return NATURAL;
  -- Result subtype: NATURAL. Value cannot be negative since parameter
is an
  --         UNSIGNED vector.
  -- Result: Converts the UNSIGNED vector to an INTEGER.

  -- Id: D.2
  function TO_INTEGER (ARG: SIGNED) return INTEGER;
  -- Result subtype: INTEGER
  -- Result: Converts a SIGNED vector to an INTEGER.

  -- Id: D.3
  function TO_UNSIGNED (ARG, SIZE: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(SIZE-1 downto 0)
  -- Result: Converts a non-negative INTEGER to an UNSIGNED vector with
  --         the specified size.

  -- Id: D.4
  function TO_SIGNED (ARG: INTEGER; SIZE: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(SIZE-1 downto 0)
  -- Result: Converts an INTEGER to a SIGNED vector of the specified
size.

  --
================================================================================
=====
  -- Logical Operators
  --
================================================================================
=====

  -- Id: L.1
  function "not" (L: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Termwise inversion

  -- Id: L.2
  function "and" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector AND operation

  -- Id: L.3
  function "or" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector OR operation

  -- Id: L.4
  function "nand" (L, R: UNSIGNED) return UNSIGNED;
```

```
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector NAND operation

  -- Id: L.5
  function "nor" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector NOR operation

  -- Id: L.6
  function "xor" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector XOR operation

  ------------------------------------------------------------------------
---------
  -- Note : Function L.7 is not compatible with VHDL 1076-1987. Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------------
---------
  -- Id: L.7
  function "xnor" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector XNOR operation

  -- Id: L.8
  function "not" (L: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Termwise inversion

  -- Id: L.9
  function "and" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector AND operation

  -- Id: L.10
  function "or" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector OR operation

  -- Id: L.11
  function "nand" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector NAND operation

  -- Id: L.12
  function "nor" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector NOR operation

  -- Id: L.13
  function "xor" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector XOR operation

  ------------------------------------------------------------------------
---------
```

```
  -- Note : Function L.14 is not compatible with VHDL 1076-1987.
Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------------
---------
  -- Id: L.14
  function "xnor" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector XNOR operation

  --
============================================================================
=====
  -- Edge Detection Functions
  --
============================================================================
=====

  -- Id: E.1
  function RISING_EDGE (signal S: BIT) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Returns TRUE if an event is detected on signal S and the
  --         value changed from a '0' to a '1'.

  -- Id: E.2
  function FALLING_EDGE (signal S: BIT) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Returns TRUE if an event is detected on signal S and the
  --         value changed from a '1' to a '0'.

end NUMERIC_BIT;


package NUMERIC_STD is
  constant CopyRightNotice: STRING
      := "Copyright 1995 IEEE. All rights reserved.";

  --
============================================================================
=====
  -- Numeric array type definitions
  --
============================================================================
=====

  type UNSIGNED is array (NATURAL range <>) of STD_LOGIC;
  type SIGNED is array (NATURAL range <>) of STD_LOGIC;

  --
============================================================================
=====
  -- Arithmetic Operators:
  --
============================================================================
====
```

```
  -- Id: A.1
  function "abs" (ARG: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0).
  -- Result: Returns the absolute value of a SIGNED vector ARG.

  -- Id: A.2
  function "-" (ARG: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0).
  -- Result: Returns the value of the unary minus operation on a
  --         SIGNED vector ARG.

  --
================================================================================

  -- Id: A.3
  function "+" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
  -- Result: Adds two UNSIGNED vectors that may be of different
lengths.

  -- Id: A.4
  function "+" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
  -- Result: Adds two SIGNED vectors that may be of different lengths.

  -- Id: A.5
  function "+" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0).
  -- Result: Adds an UNSIGNED vector, L, with a non-negative INTEGER,
R.

  -- Id: A.6
  function "+" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0).
  -- Result: Adds a non-negative INTEGER, L, with an UNSIGNED vector,
R.

  -- Id: A.7
  function "+" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0).
  -- Result: Adds an INTEGER, L(may be positive or negative), to a
SIGNED
  --         vector, R.

  -- Id: A.8
  function "+" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0).
  -- Result: Adds a SIGNED vector, L, to an INTEGER, R.

  --
================================================================================

  -- Id: A.9
  function "-" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
```

```
  -- Result: Subtracts two UNSIGNED vectors that may be of different
lengths.

  -- Id: A.10
  function "-" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
  -- Result: Subtracts a SIGNED vector, R, from another SIGNED vector,
L,
  --         that may possibly be of different lengths.

  -- Id: A.11
  function "-" (L: UNSIGNED;R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0).
  -- Result: Subtracts a non-negative INTEGER, R, from an UNSIGNED
vector, L.

  -- Id: A.12
  function "-" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0).
  -- Result: Subtracts an UNSIGNED vector, R, from a non-negative
INTEGER, L.

  -- Id: A.13
  function "-" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0).
  -- Result: Subtracts an INTEGER, R, from a SIGNED vector, L.

  -- Id: A.14
  function "-" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0).
  -- Result: Subtracts a SIGNED vector, R, from an INTEGER, L.

  --
================================================================
=====

  -- Id: A.15
  function "*" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED((L'LENGTH+R'LENGTH-1) downto 0).
  -- Result: Performs the multiplication operation on two UNSIGNED
vectors
  --         that may possibly be of different lengths.

  -- Id: A.16
  function "*" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED((L'LENGTH+R'LENGTH-1) downto 0)
  -- Result: Multiplies two SIGNED vectors that may possibly be of
  --         different lengths.

  -- Id: A.17
  function "*" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED((L'LENGTH+L'LENGTH-1) downto 0).
  -- Result: Multiplies an UNSIGNED vector, L, with a non-negative
  --         INTEGER, R. R is converted to an UNSIGNED vector of
  --         SIZE L'LENGTH before multiplication.

  -- Id: A.18
```

```
  function "*" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED((R'LENGTH+R'LENGTH-1) downto 0).
  -- Result: Multiplies an UNSIGNED vector, R, with a non-negative
  --         INTEGER, L. L is converted to an UNSIGNED vector of
  --         SIZE R'LENGTH before multiplication.

  -- Id: A.19
  function "*" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED((L'LENGTH+L'LENGTH-1) downto 0)
  -- Result: Multiplies a SIGNED vector, L, with an INTEGER, R. R is
  --         converted to a SIGNED vector of SIZE L'LENGTH before
  --         multiplication.

  -- Id: A.20
  function "*" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED((R'LENGTH+R'LENGTH-1) downto 0)
  -- Result: Multiplies a SIGNED vector, R, with an INTEGER, L. L is
  --         converted to a SIGNED vector of SIZE R'LENGTH before
  --         multiplication.

  --
================================================================================
  --
  -- NOTE: If second argument is zero for "/" operator, a severity
level
  --       of ERROR is issued.

  -- Id: A.21
  function "/" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Divides an UNSIGNED vector, L, by another UNSIGNED vector,
R.

  -- Id: A.22
  function "/" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Divides an SIGNED vector, L, by another SIGNED vector, R.

  -- Id: A.23
  function "/" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Divides an UNSIGNED vector, L, by a non-negative INTEGER,
R.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.24
  function "/" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0)
  -- Result: Divides a non-negative INTEGER, L, by an UNSIGNED vector,
R.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.

  -- Id: A.25
  function "/" (L: SIGNED; R: INTEGER) return SIGNED;
```

```
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Divides a SIGNED vector, L, by an INTEGER, R.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.26
  function "/" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0)
  -- Result: Divides an INTEGER, L, by a SIGNED vector, R.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.

  --
================================================================================
=====
  --
  -- NOTE: If second argument is zero for "rem" operator, a severity
level
  --       of ERROR is issued.

  -- Id: A.27
  function "rem" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where L and R are UNSIGNED vectors.

  -- Id: A.28
  function "rem" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where L and R are SIGNED vectors.

  -- Id: A.29
  function "rem" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where L is an UNSIGNED vector and R is
a
  --         non-negative INTEGER.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.30
  function "rem" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where R is an UNSIGNED vector and L is
a
  --         non-negative INTEGER.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.

  -- Id: A.31
  function "rem" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where L is SIGNED vector and R is an
INTEGER.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.32
```

```
  function "rem" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L rem R" where R is SIGNED vector and L is an
INTEGER.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.

  --
================================================================================
  --
  -- NOTE: If second argument is zero for "mod" operator, a severity
level
  --       of ERROR is issued.

  -- Id: A.33
  function "mod" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L and R are UNSIGNED vectors.

  -- Id: A.34
  function "mod" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L and R are SIGNED vectors.

  -- Id: A.35
  function "mod" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L is an UNSIGNED vector and R
  --         is a non-negative INTEGER.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.36
  function "mod" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where R is an UNSIGNED vector and L
  --         is a non-negative INTEGER.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.

  -- Id: A.37
  function "mod" (L: SIGNED; R: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L is a SIGNED vector and
  --         R is an INTEGER.
  --         If NO_OF_BITS(R) > L'LENGTH, result is truncated to
L'LENGTH.

  -- Id: A.38
  function "mod" (L: INTEGER; R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(R'LENGTH-1 downto 0)
  -- Result: Computes "L mod R" where L is an INTEGER and
  --         R is a SIGNED vector.
  --         If NO_OF_BITS(L) > R'LENGTH, result is truncated to
R'LENGTH.
```

```
  --
================================================================================
  -- Comparison Operators
  --
================================================================================

  -- Id: C.1
  function ">" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L and R are UNSIGNED vectors possibly
  --         of different lengths.

  -- Id: C.2
  function ">" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L and R are SIGNED vectors possibly
  --         of different lengths.

  -- Id: C.3
  function ">" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.4
  function ">" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L is a INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.5
  function ">" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.6
  function ">" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L > R" where L is a SIGNED vector and
  --         R is a INTEGER.

  --
================================================================================

  -- Id: C.7
  function "<" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L and R are UNSIGNED vectors possibly
  --         of different lengths.

  -- Id: C.8
```

```
  function "<" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L and R are SIGNED vectors possibly
  --         of different lengths.

  -- Id: C.9
  function "<" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.10
  function "<" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L is an INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.11
  function "<" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.12
  function "<" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L < R" where L is a SIGNED vector and
  --         R is an INTEGER.

  --
==============================================================================
=====

  -- Id: C.13
  function "<=" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L and R are UNSIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.14
  function "<=" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L and R are SIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.15
  function "<=" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.16
  function "<=" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L is an INTEGER and
```

```
  --         R is a SIGNED vector.

  -- Id: C.17
  function "<=" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.18
  function "<=" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L <= R" where L is a SIGNED vector and
  --         R is an INTEGER.

  --
================================================================================

  -- Id: C.19
  function ">=" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L and R are UNSIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.20
  function ">=" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L and R are SIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.21
  function ">=" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.22
  function ">=" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L is an INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.23
  function ">=" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.24
  function ">=" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L >= R" where L is a SIGNED vector and
  --         R is an INTEGER.
```

```
  --
==============================================================================
=====

  -- Id: C.25
  function "=" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L and R are UNSIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.26
  function "=" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L and R are SIGNED vectors possibly
  --         of different lengths.

  -- Id: C.27
  function "=" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.28
  function "=" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L is an INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.29
  function "=" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.30
  function "=" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L = R" where L is a SIGNED vector and
  --         R is an INTEGER.

  --
==============================================================================
=====

  -- Id: C.31
  function "/=" (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L and R are UNSIGNED vectors
possibly
  --         of different lengths.

  -- Id: C.32
  function "/=" (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L and R are SIGNED vectors
possibly
```

```
  --         of different lengths.

  -- Id: C.33
  function "/=" (L: NATURAL; R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L is a non-negative INTEGER and
  --         R is an UNSIGNED vector.

  -- Id: C.34
  function "/=" (L: INTEGER; R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L is an INTEGER and
  --         R is a SIGNED vector.

  -- Id: C.35
  function "/=" (L: UNSIGNED; R: NATURAL) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L is an UNSIGNED vector and
  --         R is a non-negative INTEGER.

  -- Id: C.36
  function "/=" (L: SIGNED; R: INTEGER) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: Computes "L /= R" where L is a SIGNED vector and
  --         R is an INTEGER.

  --
================================================================================
=====
  -- Shift and Rotate Functions
  --
================================================================================
=====

  -- Id: S.1
  function SHIFT_LEFT (ARG: UNSIGNED; COUNT: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a shift-left on an UNSIGNED vector COUNT times.
  --         The vacated positions are filled with '0'.
  --         The COUNT leftmost elements are lost.

  -- Id: S.2
  function SHIFT_RIGHT (ARG: UNSIGNED; COUNT: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a shift-right on an UNSIGNED vector COUNT times.
  --         The vacated positions are filled with '0'.
  --         The COUNT rightmost elements are lost.

  -- Id: S.3
  function SHIFT_LEFT (ARG: SIGNED; COUNT: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a shift-left on a SIGNED vector COUNT times.
  --         The vacated positions are filled with '0'.
  --         The COUNT leftmost elements are lost.

  -- Id: S.4
  function SHIFT_RIGHT (ARG: SIGNED; COUNT: NATURAL) return SIGNED;
```

```
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a shift-right on a SIGNED vector COUNT times.
  --         The vacated positions are filled with the leftmost
  --         element, ARG'LEFT. The COUNT rightmost elements are lost.

  --
================================================================================
=====

  -- Id: S.5
  function ROTATE_LEFT (ARG: UNSIGNED; COUNT: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a rotate-left of an UNSIGNED vector COUNT times.

  -- Id: S.6
  function ROTATE_RIGHT (ARG: UNSIGNED; COUNT: NATURAL) return
UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a rotate-right of an UNSIGNED vector COUNT times.

  -- Id: S.7
  function ROTATE_LEFT (ARG: SIGNED; COUNT: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a logical rotate-left of a SIGNED
  --         vector COUNT times.

  -- Id: S.8
  function ROTATE_RIGHT (ARG: SIGNED; COUNT: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: Performs a logical rotate-right of a SIGNED
  --         vector COUNT times.

  --
================================================================================
=====

  --
================================================================================
=====

  ------------------------------------------------------------------------
---------
  --   Note : Function S.9 is not compatible with VHDL 1076-1987.
Comment
  --   out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------------
---------
  -- Id: S.9
  function "sll" (ARG: UNSIGNED; COUNT: INTEGER) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: SHIFT_LEFT(ARG, COUNT)

  ------------------------------------------------------------------------
---------
  -- Note : Function S.10 is not compatible with VHDL 1076-1987.
Comment
```

```
  --   out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------
---------
  -- Id: S.10
  function "sll" (ARG: SIGNED; COUNT: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: SHIFT_LEFT(ARG, COUNT)

  ------------------------------------------------------------------
---------
  --   Note : Function S.11 is not compatible with VHDL 1076-1987.
Comment
  --   out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------
---------
  -- Id: S.11
  function "srl" (ARG: UNSIGNED; COUNT: INTEGER) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: SHIFT_RIGHT(ARG, COUNT)

  ------------------------------------------------------------------
---------
  --   Note : Function S.12 is not compatible with VHDL 1076-1987.
Comment
  --   out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------
---------
  -- Id: S.12
  function "srl" (ARG: SIGNED; COUNT: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: SIGNED(SHIFT_RIGHT(UNSIGNED(ARG), COUNT))

  ------------------------------------------------------------------
---------
  --   Note : Function S.13 is not compatible with VHDL 1076-1987.
Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------
---------
  -- Id: S.13
  function "rol" (ARG: UNSIGNED; COUNT: INTEGER) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: ROTATE_LEFT(ARG, COUNT)

  ------------------------------------------------------------------
---------
  --   Note : Function S.14 is not compatible with VHDL 1076-1987.
Comment
  --   out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------
---------
  -- Id: S.14
```

```
  function "rol" (ARG: SIGNED; COUNT: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: ROTATE_LEFT(ARG, COUNT)

  ------------------------------------------------------------------------
---------
  -- Note : Function S.15 is not compatible with VHDL 1076-1987.
Comment
  --   out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------------
---------
  -- Id: S.15
  function "ror" (ARG: UNSIGNED; COUNT: INTEGER) return UNSIGNED;
  -- Result subtype: UNSIGNED(ARG'LENGTH-1 downto 0)
  -- Result: ROTATE_RIGHT(ARG, COUNT)

  ------------------------------------------------------------------------
---------
  --   Note : Function S.16 is not compatible with VHDL 1076-1987.
Comment
  --   out the function (declaration and body) for VHDL 1076-1987
compatibility.
  ------------------------------------------------------------------------
---------
  -- Id: S.16
  function "ror" (ARG: SIGNED; COUNT: INTEGER) return SIGNED;
  -- Result subtype: SIGNED(ARG'LENGTH-1 downto 0)
  -- Result: ROTATE_RIGHT(ARG, COUNT)

  --
========================================================================
=====
  --   RESIZE Functions
  --
========================================================================
=====

  -- Id: R.1
  function RESIZE (ARG: SIGNED; NEW_SIZE: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(NEW_SIZE-1 downto 0)
  -- Result: Resizes the SIGNED vector ARG to the specified size.
  --         To create a larger vector, the new [leftmost] bit
positions
  --         are filled with the sign bit (ARG'LEFT). When truncating,
  --         the sign bit is retained along with the rightmost part.

  -- Id: R.2
  function RESIZE (ARG: UNSIGNED; NEW_SIZE: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(NEW_SIZE-1 downto 0)
  -- Result: Resizes the SIGNED vector ARG to the specified size.
  --         To create a larger vector, the new [leftmost] bit
positions
  --         are filled with '0'. When truncating, the leftmost bits
  --         are dropped.
```

```
  --
================================================================================
  -- Conversion Functions
  --
================================================================================

  -- Id: D.1
  function TO_INTEGER (ARG: UNSIGNED) return NATURAL;
  -- Result subtype: NATURAL. Value cannot be negative since parameter
is an
  --             UNSIGNED vector.
  -- Result: Converts the UNSIGNED vector to an INTEGER.

  -- Id: D.2
  function TO_INTEGER (ARG: SIGNED) return INTEGER;
  -- Result subtype: INTEGER
  -- Result: Converts a SIGNED vector to an INTEGER.

  -- Id: D.3
  function TO_UNSIGNED (ARG, SIZE: NATURAL) return UNSIGNED;
  -- Result subtype: UNSIGNED(SIZE-1 downto 0)
  -- Result: Converts a non-negative INTEGER to an UNSIGNED vector with
  --         the specified SIZE.

  -- Id: D.4
  function TO_SIGNED (ARG: INTEGER; SIZE: NATURAL) return SIGNED;
  -- Result subtype: SIGNED(SIZE-1 downto 0)
  -- Result: Converts an INTEGER to a SIGNED vector of the specified
SIZE.

  --
================================================================================
  -- Logical Operators
  --
================================================================================

  -- Id: L.1
  function "not" (L: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Termwise inversion

  -- Id: L.2
  function "and" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector AND operation

  -- Id: L.3
  function "or" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector OR operation

  -- Id: L.4
  function "nand" (L, R: UNSIGNED) return UNSIGNED;
```

```
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector NAND operation

  -- Id: L.5
  function "nor" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector NOR operation

  -- Id: L.6
  function "xor" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector XOR operation

  -- ------------------------------------------------------------------
---------
  -- Note : Function L.7 is not compatible with VHDL 1076-1987. Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  -- ------------------------------------------------------------------
---------
  -- Id: L.7
  function "xnor" (L, R: UNSIGNED) return UNSIGNED;
  -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector XNOR operation

  -- Id: L.8
  function "not" (L: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Termwise inversion

  -- Id: L.9
  function "and" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector AND operation

  -- Id: L.10
  function "or" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector OR operation

  -- Id: L.11
  function "nand" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector NAND operation

  -- Id: L.12
  function "nor" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector NOR operation

  -- Id: L.13
  function "xor" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector XOR operation

  -- ------------------------------------------------------------------
---------
```

```
  -- Note : Function L.14 is not compatible with VHDL 1076-1987.
Comment
  -- out the function (declaration and body) for VHDL 1076-1987
compatibility.
  -- ----------------------------------------------------------------
---------
  -- Id: L.14
  function "xnor" (L, R: SIGNED) return SIGNED;
  -- Result subtype: SIGNED(L'LENGTH-1 downto 0)
  -- Result: Vector XNOR operation

  --
================================================================================
=====
  -- Match Functions
  --
================================================================================
=====

  -- Id: M.1
  function STD_MATCH (L, R: STD_ULOGIC) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: terms compared per STD_LOGIC_1164 intent

  -- Id: M.2
  function STD_MATCH (L, R: UNSIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: terms compared per STD_LOGIC_1164 intent

  -- Id: M.3
  function STD_MATCH (L, R: SIGNED) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: terms compared per STD_LOGIC_1164 intent

  -- Id: M.4
  function STD_MATCH (L, R: STD_LOGIC_VECTOR) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: terms compared per STD_LOGIC_1164 intent

  -- Id: M.5
  function STD_MATCH (L, R: STD_ULOGIC_VECTOR) return BOOLEAN;
  -- Result subtype: BOOLEAN
  -- Result: terms compared per STD_LOGIC_1164 intent

  --
================================================================================
=====
  -- Translation Functions
  --
================================================================================
=====

  -- Id: T.1
  function TO_01 (S: UNSIGNED; XMAP: STD_LOGIC := '0') return UNSIGNED;
  -- Result subtype: UNSIGNED(S'RANGE)
  -- Result: Termwise, 'H' is translated to '1', and 'L' is translated
  --         to '0'. If a value other than '0'|'1'|'H'|'L' is found,
```

```
  --         the array is set to (others => XMAP), and a warning is
  --         issued.

  -- Id: T.2
  function TO_01 (S: SIGNED; XMAP: STD_LOGIC := '0') return SIGNED;
  -- Result subtype: SIGNED(S'RANGE)
  -- Result: Termwise, 'H' is translated to '1', and 'L' is translated
  --         to '0'. If a value other than '0'|'1'|'H'|'L' is found,
  --         the array is set to (others => XMAP), and a warning is
  --         issued.

end NUMERIC_STD;
```