

```
package STD_LOGIC_UNSIGNED is
```

```
function "+"(L: STD_LOGIC_VECTOR; R: STD_LOGIC_VECTOR) return STD_LOGIC_VECT
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_VECT
  -- pragma label_applies_to plus
  constant length: INTEGER := maximum(L'length, R'length);
  variable result : STD_LOGIC_VECTOR (length-1 downto 0);
begin
  result := UNSIGNED(L) + UNSIGNED(R);-- pragma label plus
  return std_logic_vector(result);
end;
```

```
function "+"(L: STD_LOGIC_VECTOR; R: STD_LOGIC) return STD_LOGIC_VECTOR is
  -- pragma label_applies_to plus
  variable result : STD_LOGIC_VECTOR (L'range);
begin
  result := UNSIGNED(L) + R;-- pragma label plus
  return std_logic_vector(result);
end;
```

```
function "+"(L: UNSIGNED; R: UNSIGNED) return STD_LOGIC_VECTOR is
  -- pragma label_applies_to plus
  -- synopsys subpgm_id 260
  constant length: INTEGER := max(L'length, R'length);
begin
  return STD_LOGIC_VECTOR (
    unsigned_plus(-- pragma label plus
      CONV_UNSIGNED(L, length), CONV_UNSIGNED(R, length)));
end;
```

```
function "+"(L: UNSIGNED; R: STD_ULOGIC) return STD_LOGIC_VECTOR is
  -- pragma label_applies_to plus
  -- synopsys subpgm_id 268
  constant length: INTEGER := L'length;
begin
  return STD_LOGIC_VECTOR (
    unsigned_plus(-- pragma label plus
      CONV_UNSIGNED(L, length), CONV_UNSIGNED(R, length))) ;
end;
```

```

-- add two unsigned numbers of the same length
-- both arrays must have range (msb downto 0)
function unsigned_plus(A, B: UNSIGNED) return UNSIGNED is
    variable carry: STD_ULOGIC;
    variable BV, sum: UNSIGNED (A'left downto 0);

    -- pragma map_to_operator ADD_UNNS_OP
    -- pragma type_function LEFT_UNSIGNED_ARG
    -- pragma return_port_name Z

begin
    if (A(A'left) = 'X' or B(B'left) = 'X') then
        sum := (others => 'X');
        return(sum);
    end if;
    carry := '0';
    BV := B;

    for i in 0 to A'left loop
        sum(i) := A(i) xor BV(i) xor carry;
        carry := (A(i) and BV(i)) or
            (A(i) and carry) or
            (carry and BV(i));
    end loop;
    return sum;
end;

```

```

function CONV_UNSIGNED(ARG: UNSIGNED; SIZE: INTEGER) return UNSIGNED is
    constant msb: INTEGER := min(ARG'length, SIZE) - 1;
    subtype rtype is UNSIGNED (SIZE-1 downto 0);
    variable new_bounds: UNSIGNED (ARG'length-1 downto 0);
    variable result: rtype;
    -- synopsys built_in SYN_ZERO_EXTEND
    -- synopsys subpgm_id 372
begin
    -- synopsys synthesis_off
    new_bounds := MAKE_BINARY(ARG);
    if (new_bounds(0) = 'X') then
        result := rtype'(others => 'X');
        return result;
    end if;
    result := rtype'(others => '0');
    result(msb downto 0) := new_bounds(msb downto 0);
    return result;
    -- synopsys synthesis_on
end;

```

```
function MAKE_BINARY(A : UNSIGNED) return UNSIGNED is
  -- synopsis built_in SYN_FEED_THRU
  variable one_bit : STD_ULOGIC;
  variable result : UNSIGNED (A'range);
begin
  -- synopsis synthesis_off
  for i in A'range loop
    if (IS_X(A(i))) then
      assert false
      report "There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic ope
      severity warning;
      result := (others => 'X');
      return result;
    end if;
    result(i) := tbl_BINARY(A(i));
  end loop;
  return result;
  -- synopsis synthesis_on
end;
```

```
-- synopsis synthesis_off
type tbl_type is array (STD_ULOGIC) of STD_ULOGIC;
constant tbl_BINARY : tbl_type :=
  ('X', 'X', '0', '1', 'X', 'X', '0', '1', 'X');
-- synopsis synthesis_on
```