Shift Operators

Introduced in VHDL93, shift operators are used for shifting data vectors. They are:

- Shift left logic (SLL): Positions on the right are filled with '0's.
- Shift right logic (SRL): Positions on the left are filled with '0's.
- Shift left arithmetic (SLA): Rightmost bit is replicated on the right.
- Shift right arithmetic (SRA): Leftmost bit is replicated on the left.
- Rotate left (ROL): Circular shift to the left.
- Rotate right (ROR): Circular shift to the right.

As defined in the original packages (see chapter 3 or appendices), the only synthesizable data type that supports shift operators is BIT_VECTOR. If one of the packages for (un)signed types (*numeric_std* or *std_logic_arith*) is declared in the code, then (UN)SIGNED can also be used (though the latter package contains very few of such operators—see appendix K). If the package *std_logic_unsigned*, *std_logic_signed*, or *numeric_std_unsigned* is also declared, then STD_LOGIC_VECTOR can be employed as well (also a very reduced set).

In VHDL 2008, the following new types with support for shift operations were included: BOOLEAN_VECTOR, UFIXED, and SFIXED.

The syntax for shift operators is ⟨left_operand⟩ ⟨shift_operation⟩ ⟨right_operand⟩. The left operand must be of one of the types mentioned above, while the right operand is always an INTEGER (+ or − in front of it is allowed). However, a recommended (more universal) approach for shifting data is with the *concatenation* operator (included in the example below).

Examples Say that x is a BIT_VECTOR signal with value $x =$ "01001". Then the values produced by the assignments below are those indicated in the comments (equivalent expressions, using the *concatenation* operator, are shown between parentheses).

```
-------------------------------------------------------------------------
y <= x SLL 2;   --y<="00100" (y <= x(2 DOWNTO 0) & "00";)
y <= x SLA 2;   --y<="00111" (y <= x(2 DOWNTO 0) & x(0) & x(0);)
y <= x SRL 3;   --y<="00001" (y <= "000" & x(4 DOWNTO 3);)
y <= x SRA 3;   --y<="00001" (y <= x(4) & x(4) & x(4) & x(4 DOWNTO 3);)
y <= x ROL 2;   --y<="00101" (y <= x(2 DOWNTO 0) & x(4 DOWNTO 3);)
y <= x SRL -2;  --same as "x SLL 2"
-------------------------------------------------------------------------
```

Concatenation Operator

Used for grouping objects and values (useful also for shifting data, as shown in the example above), the concatenation operator's representation is &.