

## VHDL OPERATORS

**Logic operators** are the heart of logic equations and conditional statements

AND	OR	NOT
NAND	NOR	
XOR	XNOR	

*there is NO order of precedence so use lots of parentheses*  
*XNOR was not in original VHDL (added in 1993)*

### Relational Operators:

Used in conditional statements

=	equal to
/=	not equal to
<	less than
<=	less then or equal to
>	greater than
>=	greater than or equal to

### Adding Operators

+	addition
-	subtraction
&	concatenation

puts two bits or bit\_vectors into a bit\_vector

example:

```
signal A: bit_vector(5 downto 0);
signal B,C: bit_vector(2 downto 0);
B <= '0' & '1' & '0';
C <= '1' & '1' & '0';
A <= B & C; -- A now has "010110"
```

Note: you should use std\_logic\_vector and unsigned or arith packages as follows:

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

### Multiplying Operators

*	multiplication
/	division
mod	modulus
rem	remainder

*mod* & *rem* operate on integers & result is integer

*rem* has sign of 1<sup>st</sup> operand and is defined as:

$$A \text{ rem } B = A - (A/B) * B$$

*mod* has sign of 2<sup>nd</sup> operand and is defined as:

$$A \text{ mod } B = A - B * N \quad \text{-- for an integer } N$$

examples:

```
7 mod 4           -- has value 3
-7 mod 4          -- has value -3
7 mod (-4)       -- has value -1
-7 mod (-4)      -- has value -3
```

# VHDL OPERATORS

## Misc. Operators

**	exponentiation
	left operand = integer or floating point
	right operand = integer only
abs	absolute value
not	inversion

## Shift Operators

sll	shift left logical	(fill value is '0')
srl	shift right logical	(fill value is '0')
sla	shift left arithmetic	(fill value is right-hand bit)
sra	shift right arithmetic	(fill value is left-hand bit)
rol	rotate left	
ror	rotate right	

all operators have two operands:

left operand is bit\_vector to shift/rotate

right operand is integer for # shifts/rotates

- integer same as opposite operator with + integer

examples:

"1100"	sll 1	yields "1000"
"1100"	srl 2	yields "0011"
"1100"	sla 1	yields "1000"
"1100"	sra 2	yields "1111"
"1100"	rol 1	yields "1001"
"1100"	ror 2	yields "0011"
"1100"	ror -1	same as "1100" rol 1

Highest	Order of Precedence for Operators				Lowest
Misc.	Multiplying	Adding	Shift	Relational	Logic

## Evaluation Rules:

1. Operators evaluated in order of precedence highest are evaluated first
2. Operators of equal precedence are evaluated from left to right
3. Deepest nested parentheses are evaluated first

Because of #2 you should use lots of parentheses