Identifiers, Data Types, and Operators

Identifiers, Data Types, and Operators

- Identifiers
  - Basic identifiers: start with a letter, do not end with "_"
  - Case insensitive
- Data Objects
  - Signals
  - Constants
  - Variables
  - Files

VHDL Standard Data Types

Type	Range of values	Example declaration
integer	implementation defined	**signal** index: **integer**:= 0;
real	implementation defined	**variable** val: **real**:= 1.0;
boolean	(TRUE, FALSE)	**variable** test: **boolean**:=TRUE;
character	defined in package STANDARD	**variable** term: **character**:= '@';
bit	0, 1	**signal** In1: **bit**:= '0';
bit_vector	array with each element of type bit	**variable** PC: **bit_vector**(31 **downto** 0)
time	implementation defined	**variable** delay: **time**:= 25 **ns**;
string	array with each element of type character	**variable** name : **string**(1 **to** 10) := "model name";
natural	0 to the maximum integer value in the implementation	**variable** index: **natural**:= 0;
positive	1 to the maximum integer value in the implementation	**variable** index: **positive**:= 1;

Data Types (cont.)

- Enumerated data types are particularly useful for constructing models of computing systems
  - Examples

    type instr_opcode **is** ('add', 'sub', 'xor', 'nor', 'beq', 'lw', 'sw');
    type state **is** ('empty', 'half_full', 'half_empty', 'empty');
- Array types

```
type byte is array (7 downto 0) of std_logic;
type word is array (31 downto 0) of std_logic;
type memory is array (0 to 4095) of word;
```

Physical Types

```
type time is range   <implementation dependent>
units
fs;                  -- femtoseconds
ps = 1000 fs;        -- picoseconds
ns = 1000 ps;        -- nanoseconds
us = 1000 ns;        -- microseconds
ms = 1000 us;        -- milliseconds
s = 1000 ms;         -- seconds
min = 60 s;          -- minutes
hr = 60 min;         -- hours
end units;

type power is range 1 to 1000000        In terms of base units
units
uw;                  -- base unit is microwatts
mw = 1000 uw;        -- milliwatts
w = 1000 mw;         -- watts
kw = 1000000 mw      -- kilowatts
mw = 1000 kw;        -- megawatts
end units;
```


Physical Types: Example

```
entity inv_rc is
generic (c_load: real:= 0.066E-12); -- farads
port (i1 : in std_logic;
      o1: out: std_logic);
constant rpu: real:= 25000.0; --ohms
constant rpd: real :=15000.0; -- ohms
end inv_rc;

                                  explicit type casting and range management
architecture delay of inv_rc is
constant tplh: time := integer (rpu*c_load*1.0E15)*3 fs;
constant tpll: time := integer (rpu*c_load*1.0E15)*3 fs;
begin
o1 <= '1' after tplh when i1 = '0' else
    '0' after tpll when i1- = '1' or i1 = 'Z' else
    'X' after tplh;
end delay;
```

Example adapted from "VHDL: Analysis and Modeling of Digital Systems," Z. Navabi, McGraw Hill, 1998.

Physical Types: Example (cont.)

```
type capacitance is range 0 to
    1E16
units
ffr; -- femtofarads
pfr = 1000 ffr;
nfr = 1000 pfr;
ufr = 1000 nfr
mfr = 1000 ufr
far = 1000 mfr;
kfr = 1000 far;
end units;
```

```
type resistance is range 0 to 1E16
units
l_o; -- milli-ohms
ohms = 1000 l_o;
k_o= 1000 ohms;
m_o = 1000 k_o;
g_o = 1000 m_o;
end units;
```

- Rather than mapping the values to the real numbers, create new physical types

Example adapted from "VHDL: Analysis and Modeling of Digital Systems," Z. Navabi, McGraw Hill, 1998.

Physical Types: Example (cont.)

```
entity inv_rc is
generic (c_load: capacitance := 66 ffr); -- farads
port (i1 : in std_logic;
      o1: out: std_logic);
constant rpu: resistance:= 25000 ohms;
constant rpd : resistance := 15000 ohms;
end inv_rc;

architecture delay of inv_rc is

constant tplh: time := (rpu/ 1 l_o)* (c_load/1 ffr) *3 fs/1000;
constant tpll: time := (rpu/ 1 l_o)* (c_load/1 ffr) *3 fs/1000;
begin
o1 <= '1' after tplh when i1 = '0' else
    '0' after tpll when i1 = '1' or i1 = 'Z' else
    'X' after tplh;
end delay;
```

Define a new overloaded multiplication operator

This expression now becomes

rpu * c_load * 3

Example adapted from "VHDL: Analysis and Modeling of Digital Systems," Z. Navabi, McGraw Hill, 1998.

Modeling with Physical Types

- Use packages to encapsulate type definitions, type conversions functions and arithmetic functions for new types
- Examples
  - Modeling power
  - Modeling silicon area
  - Modeling physical resources that are “cumulative”

Operators

•VHDL ‘93 vs. VHDL ‘87 operators

logical operators	**and**	**or**	**nand**	**nor**	**xor**	**xnor**
relational operators	**=**	**/=**	**<**	**<=**	**>**	**>=**
shift operators	**sll**	**srl**	**sla**	**sra**	**rol**	**ror**
addition operators	**+**	–	**&**			
unary operators	**+**	–				
multiplying operators	*	/	**mod**	**rem**		
miscellaneous operators	**	**abs**	**not**	**&**		

- VHDL text or language reference manual for less commonly used operators and types