Identifiers, Data Types, and Operators

- **Identifiers**
  - basic identifiers: start with a letter, do not end with "_"
  - case insensitive

- **Data Objects**
  - signals
  - constants
  - variables
  - files

<table>
<thead>
<tr>
<th>Type</th>
<th>Range of values</th>
<th>Example declaration</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>implementation defined</td>
<td>signal index: integer:= 0;</td>
</tr>
<tr>
<td>real</td>
<td>implementation defined</td>
<td>variable val: real:= 1.0;</td>
</tr>
<tr>
<td>boolean</td>
<td>(TRUE, FALSE)</td>
<td>variable test: boolean:=TRUE;</td>
</tr>
<tr>
<td>character</td>
<td>defined in package STANDARD</td>
<td>variable term: character:= `'@';</td>
</tr>
<tr>
<td>bit</td>
<td>0, 1</td>
<td>signal In1: bit:= '0';</td>
</tr>
<tr>
<td>bit_vector</td>
<td>array with each element of type bit</td>
<td>variable PC: bit_vector(31 downto 0)</td>
</tr>
<tr>
<td>time</td>
<td>implementation defined</td>
<td>variable delay: time:= 25 ns;</td>
</tr>
<tr>
<td>string</td>
<td>array with each element of type character</td>
<td>variable name : string(1 to 10):= “model name”;</td>
</tr>
<tr>
<td>natural</td>
<td>0 to the maximum integer value in the implementation</td>
<td>variable index: natural:= 0;</td>
</tr>
<tr>
<td>positive</td>
<td>1 to the maximum integer value in the implementation</td>
<td>variable index: positive:= 1;</td>
</tr>
</tbody>
</table>
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 100000
  units
  uw;-- base unit is microwatts
  mw = 1000 uw;-- milliwatts
  w = 1000 mw;-- watts
  kw = 1000000 mw-- kilowatts
  mw = 1000 kw;-- megawatts
end units;
```
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

<table>
<thead>
<tr>
<th>logical operators</th>
<th>and</th>
<th>or</th>
<th>nand</th>
<th>nor</th>
<th>xor</th>
<th>xnor</th>
</tr>
</thead>
<tbody>
<tr>
<td>relational operators</td>
<td>:=</td>
<td>/=</td>
<td>&lt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&gt;=</td>
</tr>
<tr>
<td>shift operators</td>
<td>sll</td>
<td>srl</td>
<td>sla</td>
<td>sra</td>
<td>rol</td>
<td>ror</td>
</tr>
<tr>
<td>addition operators</td>
<td>+</td>
<td>–</td>
<td>&amp;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unary operators</td>
<td>+</td>
<td>–</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>multiplying operators</td>
<td>*</td>
<td>/</td>
<td>mod</td>
<td>rem</td>
<td></td>
<td></td>
</tr>
<tr>
<td>miscellaneous operators</td>
<td>**</td>
<td>abs</td>
<td>not</td>
<td>&amp;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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