Arithmetic Packages- Introduction

- It would be very painful if when building a counter we had to think about all the internals
  - We need an adder etc.
    » How do we build the adder?
  - Would be much better if we could simply say
    » Add 1 to the current counter value
    » count <= count + 1
  - And an (optimal) circuit was automatically generated for us
  - Furthermore, if we work in this way the design is still architecturally independent- A good thing!!
Arithmetic Package Overview - I

• VHDL offers a number of packages which provide common arithmetical functions
  – Addition (+)
  – Subtraction (-)
  – Multiplication (*)
  – Division (/)
  – Comparison (=, >, <, <=, >=, /=)

• By simply adding in a ‘use’ clause at the top of your code these become available
  – Synthesis automatically generates the logic required !!
Arithmetic Package Overview- II

• There are a number of different packages that exist for historical reasons
  – STD_LOGIC_ARITH, std_logic_signed, std_logic_unsigned
  – NUMERIC_STD (IEEE)

• We will only consider ‘NUMERIC_STD’ as it is the only standard package which is defined on all commercial synthesis and simulation tools
  – Tools must provide a common set of arithmetical functions
  – Synthesis result (gates and how they are connected) will change with synthesis tool, but functionality will not
Declaring Arithmetic Signals & Variables

• NUMERIC_STD offers 2 data types
  – SIGNED, UNSIGNED
  – These are declared in a similar method to ‘std_logic_vector’
  – Can be used to ‘declare’ signals, variables, even ports in an entity

• UNSIGNED
  – Assumes that only positive values are going to be used
  – Example declaration
    ```
    signal count: unsigned (3 downto 0)
    ```
  – This creates a signal used for storing values 0 -> 15
Declaring Arithmetic Signals & Variables

- **SIGNED**
  - 2’s complement form, with MSB used as a sign bit
  - Example declaration
    ```vhd
    signal count: signed (3 downto 0)
    ```
  - Creates a signal used for storing the values -8 -> +7

<table>
<thead>
<tr>
<th>Integer</th>
<th>Signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>-8</td>
<td>1000</td>
</tr>
<tr>
<td>-1</td>
<td>1111</td>
</tr>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>+7</td>
<td>0111</td>
</tr>
</tbody>
</table>
Representation of Signed/Unsigned

• Signed/Unsigned values are represented using a subset of std_logic_vector
  – I.e. ‘0’, ‘1’ in each bit
• However, cannot perform comparisons, assignments etc. directly with std_logic
  – We need to use conversion functions (see later)
Arithmetic Package Functions- I

• For a detailed list of functions (and their operations) see the program listing from ‘NUMERIC_STD.VHD’ this is the official IEEE package
• How do I read the package header?
• Consider the function Id: A.6

```
function “+” (L: UNSIGNED; R: NATURAL) return UNSIGNED
UNSIGNED + NATURAL = UNSIGNED
```
## Arithmetic Package Functions - II

- **Signed Arithmetic Functions:**

<table>
<thead>
<tr>
<th>Function</th>
<th>Argument 1</th>
<th>Argument 2</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>signed</td>
<td>signed</td>
<td>signed</td>
</tr>
<tr>
<td></td>
<td>signed</td>
<td>integer</td>
<td>signed</td>
</tr>
<tr>
<td>-</td>
<td>signed</td>
<td>signed</td>
<td>signed</td>
</tr>
<tr>
<td></td>
<td>signed</td>
<td>integer</td>
<td>signed</td>
</tr>
</tbody>
</table>

- i.e. functions to add/ subtract signed numbers
  - Note integer as a +ve/ -ve argument for signed op’s
Arithmetic Package Functions - III

- Unsigned Arithmetic Functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Argument 1</th>
<th>Argument 2</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>unsigned</td>
<td>unsigned</td>
<td>unsigned</td>
</tr>
<tr>
<td></td>
<td>unsigned</td>
<td>natural</td>
<td>unsigned</td>
</tr>
<tr>
<td>-</td>
<td>unsigned</td>
<td>unsigned</td>
<td>unsigned</td>
</tr>
<tr>
<td></td>
<td>unsigned</td>
<td>natural</td>
<td>unsigned</td>
</tr>
</tbody>
</table>

- i.e. functions to add/ subtract unsigned numbers
  - Note natural as a +ve only argument since no notion of sign
### Arithmetic Package Functions - IV

- **Comparison functions:**

<table>
<thead>
<tr>
<th>Function</th>
<th>Name</th>
<th>Argument 1</th>
<th>Argument 2</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Equal</td>
<td><strong>unsigned</strong></td>
<td><strong>unsigned</strong></td>
<td></td>
</tr>
<tr>
<td>/=</td>
<td>Not equal</td>
<td><strong>signed</strong></td>
<td><strong>signed</strong></td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td><strong>natural</strong></td>
<td><strong>unsigned</strong></td>
<td>boolean</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td><strong>integer</strong></td>
<td><strong>signed</strong></td>
<td></td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than/equal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than/equal</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Arithmetic Package Functions - V

- **Resize functions**
  - Used for resizing a signed/unsigned value
  - Useful if we want to extract carry bit etc.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Argument 1</th>
<th>Argument 2</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>resize</td>
<td>Resize argument</td>
<td>unsigned</td>
<td>natural (new size)</td>
<td>unsigned (new size)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>signed</td>
<td>natural (new size)</td>
<td>signed (new size)</td>
</tr>
</tbody>
</table>

- **Example**

  \[
  \text{newvalue} = \text{resize}(\text{oldvalue}, 5)
  \]
Arithmetic Package Functions- VI

• Simple conversion functions
  – Used to convert integer/natural numbers to and from signed/unsigned

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Argument 1</th>
<th>Argument 2</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>to_integer</td>
<td>Convert to integer</td>
<td>unsigned</td>
<td>signed</td>
<td>natural</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>integer</td>
</tr>
<tr>
<td>to_unsigned</td>
<td>Convert to unsigned</td>
<td>natural</td>
<td>natural</td>
<td>unsigned</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(size)</td>
<td>(size)</td>
</tr>
<tr>
<td>to_signed</td>
<td>Convert to signed</td>
<td>integer</td>
<td>natural</td>
<td>signed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(size)</td>
<td>(size)</td>
</tr>
</tbody>
</table>
Arithmetic Package Functions- VII

• Conversion to/ from std_logic_vector
  – Used to convert signed and unsigned values to and from std_logic_vector
    » Really just copies each bit

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Argument</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned</td>
<td>Convert to unsigned</td>
<td>std_logic_vector</td>
<td>unsigned</td>
</tr>
<tr>
<td>signed</td>
<td>Convert to signed</td>
<td>std_logic_vector</td>
<td>signed</td>
</tr>
<tr>
<td>std_logic_vector</td>
<td>Convert to std_logic_vector</td>
<td>unsigned</td>
<td>std_logic_vector</td>
</tr>
<tr>
<td>std_logic_vector</td>
<td>Convert to std_logic_vector</td>
<td>signed</td>
<td>std_logic_vector</td>
</tr>
</tbody>
</table>
Making the Package Visible

• At the top of the VHDL source file, the line

```vhdl
use ieee.numeric_std.all
```

• is added

• This makes the package (functions, data types, etc.) ‘visible’
Arithmetic Package Example- I

• Some concrete examples of the use of the arithmetic package!
• Simple counter
  – Counts up to certain value then resets back
  – Only increments if input ‘inc’ is set to ‘1’

CLK

INC

COUNT

0000 0001 0001 0010 0011 0100 0000

No change when inc = ‘0’
• **VHDL process**

```vhdl
process (CLK, RESET)
begin
    if RESET='1' then -- reset (asynchronous)
        internal_count <= "0000";
    elsif CLK'EVENT and CLK='1' then -- clock
        if inc = '1' then
            if internal_count = "0100" then
                internal_count <= "0000";
            else
                internal_count <= internal_count + 1;
            end if;
        else
            null;
        end if;
    end if;
end process;
```
• Architecture definition

```vhdl
use ieee.numeric_std.all;

architecture BEHAVIOR of UPCOUNTER is
  signal internal_count : unsigned (3 downto 0);
begin
  -- see previous slide for process
  count <= std_logic_vector(internal_count);
end BEHAVIOR;
```
Arithmetic Packages- Conclusion

• Have given an overview of the numeric package and the functions which it contains
• These are widely use for counters etc.
• Can be used to generate many different arithmetic circuits (e.g. multipliers etc.) though it may be better to design these for the given application as the target after synthesis may be inefficient