Lesson 7

Libraries and Data Types

Mohd Saufy Rohmad
Technical Trainer and Consultant

Libraries and Data Types

- VHDL signals, such as inputs and outputs, must have a type declaration.
- Those types are defined in different libraries.
- This is one of the most confusing parts of VHDL that scares the programmers.

Library

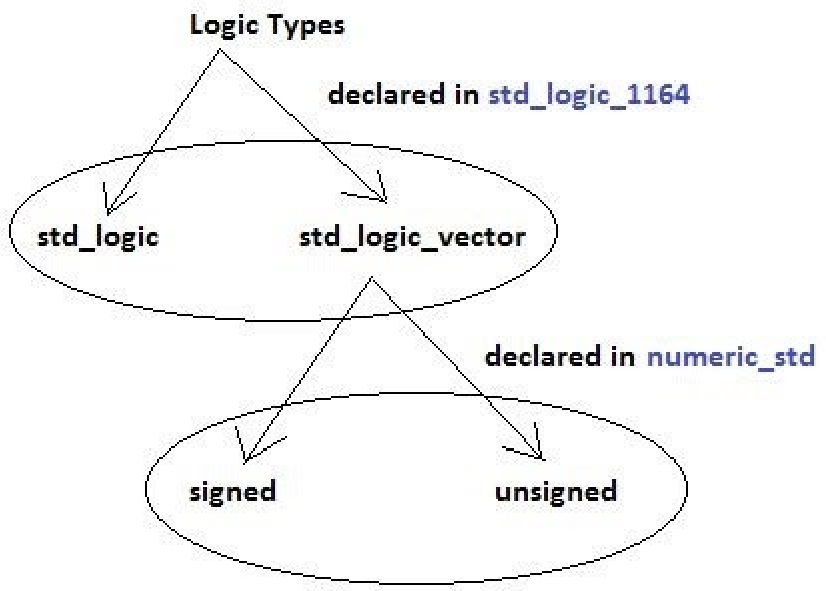
- A library is a directory and each package is a file in that directory.
- The package file is a database containing information about the components in the package (the component inputs, outputs, types, etc).
- To use a component in a design, we use library statements to specify the libraries to be searched and a use statement for each package we need to use.
- The two most commonly used libraries are called IEEE and WORK.
- The WORK library is always available without having to use a library statement.
- USAGE : library ieee;

Library

- If you create your custom data types in a package, it will come under the work library.

  STD_LOGIC_1164 package:

  STD_LOGIC_1164 package:



Library

- The IEEE library contains one of the most basic package, std_logic_1164.
- It contains the types std_logic (similar to bit) and std_logic_vector (similar to bit_vector) data types.
- Digital signals are different in the sense that it can have values other than '0' or '1'.
- It can have undefined or high impedance conditions as shown in the flip flop structure in Hour 3.
- The advantage of the std_logic types is that they can have values other than ’0’ and ’1’.
- For example, an std_logic signal can also have undefined (’X’) and high-impedance values (’Z’).
- The std_logic_1164 package also redefines (“overloads”) the standard boolean operators (and, or, not, etc.) so that they work with std_logic signals.
- USAGE: IEEE.STD_LOGIC_1164.ALL

Why only STD_LOGIC_1164 package is not enough?

- IEEE.STD_LOGIC_1164 lacks basic operations such as addition, comparison, shifts, and conversion to integers for the STD_LOGIC_VECTOR data.
- It is possible to use those basic operations with signed, unsigned or integer type of datas.
- However, if you only use std_logic_1164, then you have to convert the signals every time to singed/unsigned, which will make the code very messy.
- Thats why we use additional packages like NUMERIC_STD.

NUMERIC_STD package:

- numeric_std is also defined in the IEEE library. It supports signed and unsigned data types.
- These are sub types of std_logic_vector with overloaded operators that allow them to be used both as vectors of logic values and as as binary numbers (in signed two’s complement or unsigned representations).
- The hierarchy of these logic types could be drawn as follows:
- The standard arithmetic operators (+, -, *, /, **, >, <, <=, >=, =, /=) can be applied to signals of type signed or unsigned.
- Note that it may not be practical or possible to synthesize complex operators such as multiplication, division or exponentiation.

NUMERIC_STD package:

- List of useful functions under NUMERIC_STD:
  - signed() -- converts a std_logic_vector/unsigned to signed
  - unsigned() -- converts a std_logic_vector/signed to unsigned
  - std_logic_vector() - converts a singed/unsigned to std_logic_vector
  - Note that, you cant directly convert a std_logic_vector to integer
  - to_integer() -- converts a signed/unsigned to integer
  - to_signed() -- converts an integer to signed
  - to_unsigned() -- converts an integer to unsigned

STD_LOGIC_ARITH package:

- std_logic_arith is also defined in the IEEE library and it also does similar operations like numeric_std package.
- However, it is recommended to use numeric_std package over std_logic_arith package.
- Because std_logic_arith, etc. were packages written by Synopsis, a leading EDA company and included in their tools' version of the ieee library, without the approval of the ieee standardization body.
- IEEE decided to write and approve a standard package for numerical operations on SL(std_logic) and BIT based vectors.
- Those packages are called numeric_std and numeric_bit.
- These are ieee standard packages, located in the official version of the ieee library, and their behavior is governed by the standard, so compatibility is assured.

Code

Any VHDL code begins with..........

library IEEE;

use ieee.std_logic_1164.all;

use ieee.numeric_std.all

uppercase or lowercase doesnt matter.

- We recommend using unsigned or signed data types instead of std_logic_vector whenever possible

Library and Data Types

- We can declare a vector signal, that contains more than one bit with std_logic_vector that comes with the std_logic_1164 package.
- As we have discussed already in the last hour, the std_logic_vector cant be added, subtracted, multiplied etc directly.
- So we always have to convert them to unsigned or signed using the package numeric_std.
- For simplifying the VHDL, we recommend using unsigned or signed directly instead of std_logic_vector.
- It wont have any effect in the synthesis of the circuit.
- Here we will demonstrate how using unsigned/signed makes your life easier.

Library and Data Types

- Lets say, we want to make an adder that will add two 8-bit vectors A and B respectively.
- We have to use the numeric_std package to do the addition because std_logic_1164 doesnt support addition.
- Now if we declare the inputs A and B as 8-bit std_logic_vector(7 downto 0), we need to convert them to unsigned or signed first to add the numbers.
- Besides, we have to resize them to a 9-bit output (to avoid overflow).
- The way is the concatenate a bit '0' in the MSB of each input A and B. And then add them.

Library and Data Types

- Lets say, we want to make an adder that will add two 8-bit vectors A and B respectively.
- We have to use the numeric_std package to do the addition because std_logic_1164 doesnt support addition.
- Now if we declare the inputs A and B as 8-bit std_logic_vector(7 downto 0), we need to convert them to unsigned or signed first to add the numbers.
- Besides, we have to resize them to a 9-bit output (to avoid overflow).
- The way is the concatenate a bit '0' in the MSB of each input A and B. And then add them.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity adder_8bit is
port (
A,B: in std_logic_vector (7 downto 0);
C: out std_logic_vector (8 downto 0));
end  adder_8bit;

architecture  adder_8bit_arch of  adder_8bit is
begin
C <= std_logic_vector(unsigned(('0' & A)) + unsigned(('0' & B)));
end  adder_8bit_arch;
```

But...

- However, we have to convert each input and the concatenation first to unsigned, so that it is possible to add them.
- Afterwards, we have to convert them back to std_logic_vector because our output is std_logic_vector type.
- We can make life simpler with declaring the input as unsigned type directly.
- We dont need any kind of conversion in that way.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use ieee.numeric_std.all;
entity adder_8bitb is
port (
A,B: in unsigned (7 downto 0);
C: out unsigned (8 downto 0));
end  adder_8bitb;

architecture  adder_8bitb_arch of  adder_8bitb is
begin
     C <= ('0' & A) + ('0' & B);
end  adder_8bitb_arch;
```

But...

- Even the concatenation of a bit in the MSB is not a very good idea.
- We know in this case that the output needs to have 1 bit extra to avoid overflow.
- But lets think, we need to do multiplication, the output bits will be about double the input size.
- For example, the output should be 16-bit if we multiply two 8-bit vectors to avoid overflow.
- We can do some other operations like division or multiply-and-accumulate where the output size could be different.
- How many bits are we gonna concatenate in each case? It would be nearly impossible to keep track of all those concatenations.
- Thats why numeric_std comes with a very useful function called resize.
- This function can take the first argument as the logic operations, for example A+B and the second argument as the size of the desired output.
- In this case, we want to have an output of 9 bits.
- So the second argument is going to be 9.
- The overall VHDL code would be resize(A+B,9);

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity adder_8bitc is
port (
A,B: in unsigned (7 downto 0);
C: out unsigned (8 downto 0));
end  adder_8bitc;

architecture  adder_8bitc_arch of  adder_8bitc is
begin
C <= resize(A+B, 9);
end  adder_8bitc_arch;
```

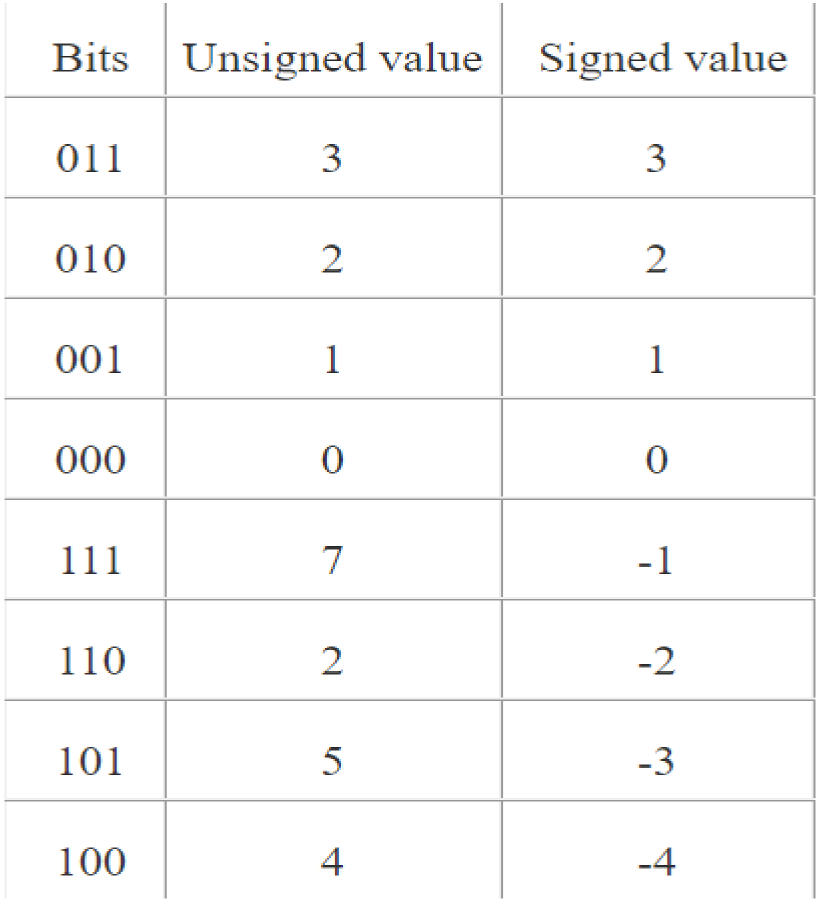
Signed unsigned

- You might have noticed already that the second and third VHDL codes look much simpler and cleaner than the first one.
- If the first code was not so simple, it would make things more messy and it would be impossible to keep track for a big design.
- That's why we recommend using unsigned and signed instead of the std_logic_vector.

Signed vs unsigned in VHDL

- Signed and unsigned types exist in the numeric_std package, which is part of the ieee library.
- It should be noted that there is another package file that is used frequently to perform mathematical operations: std_logic_arith.
- However, std_logic_arith is not an official ieee supported package file and it is not recommended for use in digital designs.
- A signal that is defined as type signed means that the tools interpret this signal to be either positive or negative.
- A signal that is defined as type unsigned means that the signal will be only positive.
- Internally, the FPGA will use Two's Complement representation.
- For example, a 3-bit signal can be interpreted according to the table below:

Signed vs unsigned in VHDL



Bits	Unsigned value	Signed value
011	3	3
010	2	2
001	1	1
000	0	0
111	7	-1
110	2	-2
101	5	-3
100	4	-4

Thank You