Data Types

Young Won Lim
06/19/2025

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

4 Classes of Data Types

- Scalar Type
  - Enumeration Type
  - Numerical Data Types
    - Integer
    - Real
  - Physical Data Types
- Composite Type
  - Array
  - Record
- Access Type
  - dynamic memory allocation
- File Type
  - test vectors

4 Classes of Data Types

```
entity FULLADDER is
  port(A, B, CARRY_IN : in bit;
       SUM, CARRY     : out bit);
end FULLADDER;
```

```
architecture MIX of FULLADDER is
  component HALFADDER
    port(A, B       : in bit;
         SUM, CARRY : out bit);

  signal W_SUM, W_CARRY1, W_CARRY2 : bit;

begin
  HA1: HALFADDER
    port map(A, B, W_SUM, W_CARRY1);

  HA2: HALFADDER
    port map(CARRY_IN, W_SUM,
             SUM, W_CARRY2);

  CARRY <= W_CARRY1 or W_CARRY2;

end MIX;
```

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

4 Classes of Data Types

Every signal has a type

Type specifies possible values

Types has to be defined at signal declaration …

… either in

entity: port declaration,

or in

architecture: signal declaration

Types have to match

```
port(A, B, CARRY_IN : in bit;
     SUM, CARRY     : out bit);

port(A, B       : in bit;
     SUM, CARRY : out bit);

signal W_SUM, W_CARRY1, W_CARRY2 : bit;
```

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

4 Classes of Data Types

In VHDL, signals must have
a data type associated with them
that limits the number of possible values.

This type has to be fixed when the signal is declared,
either as entity port or an internal architecture signal,

and can not be changed during runtime.

Whenever signal values are updated,
the data types on both sides of
the assignment operator '<=' have to match.

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Standard Data Types

```
package STANDARD is
   type BOOLEAN is (FALSE,TRUE);
   type BIT is ('0','1');
   type CHARACTER is (-- ascii set);
   type INTEGER is range
        -- implementation_defined
   type REAL is range
        -- implementation_defined
   -- BIT_VECTOR, STRING, TIME
end STANDARD;
```

Every type has a number of possible values

Standard types are defined by the language

User can define his own types

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Standard Data Types

A number of data types are already defined
in the standard package
which is always implicitly referenced.

'boolean' is usually used to control the flow of the VHDL execution
while 'bit' uses level values ('0', '1') instead of truth values ('false', 'true')
and is therefore better suited to model wires.
Number values can be communicated
via signals of type 'integer' or 'real'.

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Standard Data Types

The actual range and accuracy
depends on the platform implementation
and only lower bounds are defined,
e.g. integers are guaranteed to be at least 32 bits wide.
Floating point operations can not be synthesized automatically,
yet, i.e. the use of 'real' data types is restricted to testbench applications.

The same applies to 'character' and 'time'.

Real types are not synthesizeable:

You have to decide how many bits
will be used for the digits pre and after the decimal point!
you have to use synthesizeable division algorithms
to calculate them

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Data Type ‘time”

```
architecture EXAMPLE of TIME_TYPE is
  signal CLK : bit := '0';
  constant PERIOD : time := 50 ns;
begin
  process
  begin
    wait for 50 ns;
    ...
    wait for PERIOD;
    ...
    wait for 5 * PERIOD;
    ...
    wait for PERIOD * 5.5;
  end process;
...
  -- concurrent signal assignment
  CLK <= not CLK after 0.025 us;

  -- or with constant time
  -- CLK <= not CLK after PERIOD/2;

end EXAMPLE;
```

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Data Type ‘time”

Usage
- Testbenches
- Gate delays

Multiplication / division
- Multiplied / divided by integer / real
- Returns TIME type
- Internally in smallest unit (fs)

Available time units fs, ps, ns, us, ms, sec, min, hr

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Data Type ‘time”

’time’ is a special data type
as it consists out of a numerical value
and a physical unit.

It is used to delay the execution of statements
for a certain amount of time, e.g. in testbenches or
to model gate and propagation delays.

Signals of data type ’time’ can be multiplied
or divided by ’integer’ and ’real’ values.

The result of these operations remains
of data type ’time’.

The internal resolution of VHDL simulators
is set to femto-seconds (fs).

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Definition of Arrays

Signal A_BUS, Z_BUS : bit_vector (3 downto 0);

Definition of Arrays

Collection of signals of the same type

Predefined arrays
bit_vector (array of bit)
string (array of character)

Unconstrained arrays:
definition of actual size during signal/port declaration

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Definition of Arrays

Arrays are useful to group
signals of the same type and meaning.

Two unconstrained array data types,
i.e. whose range is not limited,
are predefined in VHDL:

'bit_vector' and 'string'
are arrays of 'bit' and 'character' values, respectively.

Please note that the array boundaries
have to be fixed during signal declarations,
e.g. 'bit_vector(3 downto 0)'.

Only constrained arrays may be used
as entity ports or architecture signals.

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

‘integer’ and ‘bit’ types

```
architecture EXAMPLE_1 of DATATYPES
 is
  signal SEL :  bit;
  signal A, B, Z :
        integer range 0 to 3;

begin
  A <= 2;
  B <= 3;
  process(SEL,A,B)
  begin
    if SEL = ‘1’ then
      Z <= A;
    else
      Z <= B;
    end if;
  end process;
end EXAMPLE_1;
```

```
architecture EXAMPLE_2 of DATATYPES
 is
  signal SEL : bit;
  signal A, B, Z :
        bit_vector(1 downto 0);

begin
    A <= "10";
    B <= "11";
    process(SEL,A,B)
    begin
      if SEL = ‘1’ then
        Z <= A;
      else
        Z <= B;
      end if;
    end process;
end EXAMPLE_2;
```

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Definition of Arrays

Integer signals will be mapped
to a number of wires during synthesis.

These wires could be modelled via bit vectors as well,
yet 'bit_vector' signals do not have
a numerical interpretation associated with them.

Therefore the synthesis result
for the two example architectures would be the same.

The process models a simple multiplexer
which selects the input A as source for its output Z
when the select signal SEL is '1' and the input B otherwise.

Please note that the multiplexer process
is exactly the same for both data types!

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Assignment with Array Types

```
architecture EXAMPLE of ARRAYS is
  signal Z_BUS : bit_vector (3 downto 0);
  signal C_BUS : bit_vector (0 to 3);
begin
  Z_BUS <= C_BUS;
end EXAMPLE;
```

Z_BUS(3) ⇐ C_BUS(0)
Z_BUS(2) ⇐ C_BUS(1)
Z_BUS(1) ⇐ C_BUS(2)
Z_BUS(0) ⇐ C_BUS(3)

Elements are assigned according to their position, not their number
The direction of arrays should always be defined the same way

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Assignment with Array Types

Notes

Special care is necessary when signal assignments with arrays are carried out.

Although the data type and the width of the signals have to match, this is not true for the order of the array elements.

The values are assigned according to their position within the array, not according to their index.

Therefore it is highly recommended to use only one direction (usually ’downto’ in hardware applications) throughout your designs.

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Bit String Literals

```
architecture EXAMPLE of ASSIGNMENT is

  signal Z_BUS : bit_vector (3 downto 0);
  signal BIG_BUS :bit_vector (15 downto 0);

begin
  -- legal assignments:
  Z_BUS(3)   <= '1';
  Z_BUS      <= "1100";

  Z_BUS      <= b"1100";
  Z_BUS      <= x"c";
  Z_BUS      <= X"C";
  BIG_BUS    <= B"0000_0001_0010_0011";

end EXAMPLE;
```

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Bit String Literals

- Single bit values are enclosed in '.'
- Vector values are enclosed in “…”
  - optional base specification (default: binary)
  - Values may be separated by underscores to improve readability

Different specification of single bits an bit vectors

VHDL’93: Valid assignments for the data type ‘bit’ are also valid for all character arrays, e.g. ‘std_(u)logic_vector

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Bit String Literals

Notes

The specification of signal values is different for the base types ’character’ and ’bit’ and their corresponding array types ’string’ and ’bit_vector’. Single values are always enclosed in single quotation marks (’), while double quotation marks (“) are used to specify array values.

As bit vectors are often used to represent numerical values, VHDL offers several possibilities to increase the readability of bit vector assignments.

First, a base for the following number may be specified. Per default binary data consisting of ’0’s and ’1’s is assumed. Please note that the values have to enclosed in double quotation marks even though only a single symbol might be necessary when another base is used! Additionally, underscores (_) may be inserted at will to split long chains of numbers into smaller groups in order to improve readability.

Since VHDL’93, the same rules apply to the enhanced bit vector types ’std_(u)logic_vector’, which will be discussed later on, as well.

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Concatenation

```
architecture EXAMPLE_1 of CONCATENATION is
  signal BYTE : bit_vector (7 downto 0);
  signal A_BUS, B_BUS :bit_vector (3 downto 0);

begin

  BYTE   <= A_BUS & B_BUS;

end EXAMPLE_1;
```

Resulting signal assignment:

BYTE(7) ⇐ A_BUS(3)
BYTE(6) ⇐ A_BUS(2)
BYTE(5) ⇐ A_BUS(1)
BYTE(4) ⇐ A_BUS(0)
BYTE(3) ⇐ B_BUS(3)
BYTE(2) ⇐ B_BUS(2)
BYTE(1) ⇐ B_BUS(1)
BYTE(0) ⇐ B_BUS(0)

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Concatenation

```
architecture EXAMPLE_2 of CONCATENATION is
  signal Z_BUS : bit_vector (3 downto 0);
  signal A_BIT, B_BIT, C_BIT, D_BIT :bit;

begin

  Z_BUS   <= A_BIT & B_BIT & C_BIT & D_BIT;

end EXAMPLE_2;
```

Resulting signal assignment:

Z_BUS(3) ⇐ A_BIT
Z_BUS(2) ⇐ B_BIT
Z_BUS(1) ⇐ C_BIT
Z_BUS(0) ⇐ D_BIT

The Concatenation operator '&' is allowed on the right side of the signal assignment operator '⇐' , only

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Concatenation

Notes

As signal assignments require matching data types on both sides of the operator it is sometimes necessary to assemble an array in the VHDL code.

The concatenation operator ’&’ groups together the elements on its sides which have to be of the same data type, only.

Again, the array indices are ignored and only the position of the elements within the arrays is used. The concatenation operator may be used on the right side of signal assignments, only!

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Aggregate

```
architecture EXAMPLE of AGGREGATES is
  signal BYTE         : bit_vector (7 downto 0);
  signal Z_BUS        : bit_vector (3 downto 0);
  signal A_BIT, B_BIT : bit;
  signal C_BIT, D_BIT : bit;

begin
  Z_BUS <= (A_BIT, B_BIT, C_BIT, D_BIT);
  (A_BIT, B_BIT, C_BIT, D_BIT) <= bit_vector'("1011");
  (A_BIT, B_BIT, C_BIT, D_BIT) <= BYTE(3 downto 0);

  BYTE <= (7 => '1', 5 downto 1 => '1', 6 => B_BIT,
           others => '0');
end EXAMPLE;
```

Aggregates bundle signals together

May be used on both sides of an assignment

Keyword ‘other’ selects all remaining elements

Some aggregate constructs may not be supported by your synthesis tool

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Assignment of ‘0’ to all bits of a vector regardless of the width: VECTOR ⇐ (others⇒ ‘0’);

Aggregate

Notes

Another way of assigning signals which does not suffer from this limitation is via the aggregate construct.

Here, the signals that are to build the final array are enclosed in a ’(’ ’)’ pair and separated by ’,’.

Instead of a simple concatenation, it is also possible to address the array elements explicitly by their corresponding index, as shown in the last signal assignment statement of the aggregate example.

The keyword ’others’ may be used to select those indices that have not been addressed, yet.

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Slices of Arrays

```
architecture EXAMPLE of SLICES is
  signal BYTE  : bit_vector (7 downto 0);
  signal A_BUS : bit_vector (3 downto 0);
  signal Z_BUS : bit_vector (3 downto 0);
  signal A_BIT : bit;
begin

  BYTE (5 downto 2) <= A_BUS;
  BYTE (5 downto 0) <= A_BUS;      -- wrong

  Z_BUS (1 downto 0) <= '0' & A_BIT;
  Z_BUS <= BYTE (6 downto 3);
  Z_BUS (0 to 1) <= '0' & A_BIT;   -- wrong

  A_BIT   <= A_BUS (0);

end EXAMPLE;
```

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

Slices of Arrays

Slices select elements of arrays

Arrays: Size must match on both sides of an assignment

The direction of the ‘slice” and of the array must match

Notes

The inverse operation of concatenation and aggregation is the selection of slices of arrays, i.e. only a part of an array is to be used.

The range of the desired array slice is specified in brackets and must match the range declaration of the signal!

Of course, it is possible to select only single array elements.

https://www.vhdl-online.de/courses/system_design/vhdl_language_and_syntax/data_types

References

[1] http://en.wikipedia.org/
[2] J. V. Spiegel, VHDL Tutorial,
http://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html
[3] J. R. Armstrong, F. G. Gray, Structured Logic Design with VHDL
[4] Z. Navabi, VHDL Analysis and Modeling of Digital Systems
[5] D. Smith, HDL Chip Design
[6] http://www.csee.umbc.edu/portal/help/VHDL/stdpkg.html
[7] VHDL Tutorial - VHDL onlinewww.vhdl-online.de/tutorial/