Attributes

Young Won Lim
10/22/2025

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

This document was produced by using OpenOffice and Octave.

Young Won Lim
10/22/2025

Attributes

Attributes (1)

A VHDL attribute is a predefined or user-defined property
that provides extra information about a type or object,
such as array lengths (A'LENGTH) or type ranges (S'Range).

Attributes are applied using an apostrophe
(e.g., signal_name'attribute) and

are categorized as returning
- a value (like A'LENGTH)
- a signal (like A'Delayed),

and can be used for
- simulation,
- synthesis, or
- reporting purposes.

Google AI Overview : VHDL attribute

Attributes (2)

Types of VHDL Attributes

Predefined Attributes:

These are built into the VHDL language and
include information about types, signals, and other entities.

Type Attributes:

A'Length (length of an array),
S'Range (range of a signal),
S'Ascending (if a type has an ascending range).

Signal Attributes:

'Last_value (returns the last value of a signal),
'Stable (returns true if a signal hasn't changed),
'Quiet' (returns true if a signal has no transactions).

Google AI Overview : VHDL attribute

Attributes (3)

Synthesis Attributes:

Special directives for synthesis tools,
often used to control how the hardware is implemented.

'Keep' (Directs the synthesis tool to keep a specific wire or logic).

User-Defined Attributes:

These are attributes you create and assign
to specific objects in your design to add custom information.

Google AI Overview : VHDL attribute

Attributes (4)

How to Use Attributes

Syntax:
You use an apostrophe to apply an attribute to an object.

Example: variable_name'attribute_name.

Applications:

Simulation: Attributes like s'Driving can be used
within processes to understand
when a process is actively driving a signal.

Synthesis: Synthesis attributes guide
the hardware synthesis process,
ensuring certain logic is preserved.

Reporting: Attributes such as 'Instance_name
provide detailed paths for reports and assertions.

Google AI Overview : VHDL attribute

‘Image Attribute

the 'IMAGE attribute, when applied
to a type or object of a discrete type (like INTEGER),
returns a STRING representation of its value.

This is particularly useful for displaying numerical values
in a human-readable format during simulation or
for generating output messages.

Type_Name'IMAGE(Expression)

Object_Name'IMAGE

Type_Name is the name of a discrete type (e.g., INTEGER, NATURAL, POSITIVE).
Object_Name is the name of a signal, variable, or constant of a discrete type.
Expression is an expression that evaluates to a value of the specified discrete type.

Google AI Overview

‘Image Attribute

```
-- Declare a variable
variable my_integer_var : integer := 42;

-- Use 'IMAGE with a variable
report "The value is: " & integer'image(my_integer_var);

-- Use 'IMAGE with a literal
report "Another value: " & integer'image(100);

-- Use 'IMAGE with an expression
report "Result of calculation: " & integer'image(5 * 8);
```

Google AI Overview

‘Image Attribute

The 'IMAGE attribute converts the numerical value
into its decimal string representation.

For example, integer'image(42) would return the string "42".

This allows for easy integration of numerical data
into REPORT statements, ASSERT statements,
or when writing to files during simulation.

Note: The 'IMAGE attribute is primarily used
for simulation and verification purposes.

It is generally not synthesizable,
meaning it cannot be directly implemented in hardware.

For hardware implementations, numerical values are typically represented in binary,
and conversion to string form is handled in simulation or external tools.

Google AI Overview

Qualified Expression

Qualified expression (1)

a qualified expression explicitly specifies
the type of an expression,

This is achieved by preceding the expression or aggregate
with the desired type name followed by an apostrophe.

TypeName'(Expression_or_Aggregate)

Where TypeName is the desired type and
Expression_or_Aggregate is the expression or aggregate
whose type is being explicitly stated.

Google AI overview

Qualified expression (2)

particularly when the type might otherwise be ambiguous
or when an aggregate needs a specific type context.

When to use qualified expressions:

- **Aggregates**:
- **Overloaded Subprograms**:
- **Clarity** and **Readability**:

Google AI overview

Qualified expression (2)

Qualified Expression in Aggregates:

When creating an aggregate
(e.g., an array or record aggregate),
the type of the aggregate
might not be immediately clear from the context,
especially if there are multiple types
that could potentially match the elements.

A qualified expression resolves this ambiguity.

string vs. std_ulogic_vector

```
type T_2 is std_ulogic_vector(1 downto 0);
signal A, B : std_ulogic;
-- …
-- Qualified expression used
– for the aggregate (A & B)
case T_2'(A & B) is
   when "00" => -- ...
   when others => -- ...
end case;
```

Google AI overview

Qualified expression (3)

Qualified Expression in Overloaded Subprograms:

When calling an overloaded function or procedure
(where multiple subprograms share the same name
but have different parameter types),
a qualified expression can be used
to explicitly specify the type of an argument,
ensuring the correct version of the subprogram is called.

```
function PARITY ( X : bit_vector ) return bit;
function PARITY ( X : std_logic_vector ) return std_logic;
-- ...
P_BIT <= PARITY(bit_vector'("00100"));
-- Qualifies the argument for the bit_vector version
```

Google AI overview

Qualified expression (4)

Qualified Expression in Allocators: Qualified expressions can be used with allocators to define the initial value of a newly created object.

```
new T_rec'(field1 => value1, field2 => value2);
-- Qualifies the record aggregate
```

Clarity and Readability:
Even when not strictly required by the compiler,
qualified expressions can improve code readability
by making the intended type of an expression explicit.

Google AI overview

Qualified expression string’(“hello”) (1)

The most common scenario for using string'("hello") is
when calling overloaded subprograms,
such as the WRITE procedure in the TEXTIO package.

The WRITE procedure is overloaded
to accept both STRING and BIT_VECTOR types.

Without the qualified expression,
a literal like "hello"
could be ambiguous to the compiler,
as it could potentially be interpreted
as either a string or a bit vector.

By using string'("hello"),
you explicitly tell the compiler to treat "hello"
as a STRING, resolving the ambiguity.

Google AI overview

Qualified expression string’(“hello”) (2)

string'("hello") is a qualified expression
used to explicitly specify the type of a literal.

This is particularly useful in situations
where a literal could be interpreted as multiple types,
leading to ambiguity for the compiler.

string: specifies the target type,
indicating that the literal should be treated
as a VHDL STRING.
' (tick or apostrophe) is the attribute separator,
indicating that the following parenthesized expression
is an attribute or a qualified expression.
("hello") is the literal value,
enclosed in double quotes
to denote a string literal in VHDL.

Google AI overview

Data Category

Value Kind

Function Kind

Signal Kind

Type Kind

Range Kind

Value Kind Attributes

Value type atributes	'LEFT
	'RIGHT
	'HIGH
	'LOW
Value array attributes	'LENGTH
Value block attributes	'STRUCTURE
	'BEHAVIOR

Function Kind Attributes

Function type atributes	'POS(value)
	'VAL(value)
	'SUCC(value)
	'PRED(value)
	'LEFTOF(value)
	'LENGTH(value)
Function array attributes	array'LEFT(n)
	array'RIGHT(n)
	array'HIGH(n)
	array'LOW(n)
Function signal attributes	S'EVENT
	S'ACTIVE
	S'LAST_EVENT
	S'LAST_ACTIVE

Signal Kind Attributes

s'DELAYED[(time)]

s'STABLE[(time)]

s'QUIET[(time)]

s'TRANSACTION

Type Kind Attributes

t'BASE

Range Kind Attributes

a'RANGE[(n)]

a'REVERSE_RANGE[(n)]

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/