**Chapter 8: Names**

Chapter 8 focuses on the concept of names in VHDL, which are identifiers used to refer to objects, types, subprograms, and other design elements. Names ensure clarity, organization, and accessibility in a VHDL design.

**8.1 General**

* Purpose: Names uniquely identify design elements and enable referencing them in expressions, statements, and configurations.
* Types of Names: Includes simple names, selected names, indexed names, slice names, attribute names, and external names.

**8.2 Simple Names**

* Refer to declared entities within a specific scope**.**



**8.3 Selected Names**

* Refer to elements within a hierarchical design or package.
* Syntax: <prefix>.<suffix>

A close-up of a standard

Description automatically generated

**8.4 Indexed Names**

* Access individual elements in arrays or vectors using an index.
* Syntax: <array\_name>(<index>)



**8.5 Slice Names**

* Refer to a subset of an array or vector.
* Syntax: <array\_name>(<range>)

A screenshot of a computer error

Description automatically generated

**8.6 Attribute Names**

* Refer to attributes of objects, such as predefined attributes ('event, 'length, etc.) or user-defined attributes.
* Syntax: <object>'<attribute>

A white background with black text

Description automatically generated

**8.7 External Names**

* Refer to objects or entities external to the current design unit, useful for modular and hierarchical designs.



**From :** <https://adaptivesupport.amd.com/s/question/0D54U00007hX4mrSAC/vhdl-external-name-alias-for-internal-array-signal-not-being-resolved?language=en_US>

**Chapter 9: Expressions**

Chapter 9 focuses on expressions in VHDL, which are combinations of operators and operands used to compute values, control behavior, or specify conditions. Expressions are fundamental for describing hardware behavior.

**9.1 General**

Expressions: Formed by combining operands (constants, variables, signals, or literals) with operators.

Usage: Used in assignments, conditions, and computations throughout a design.

**9.2 Operators**

Operators define operations on operands. They are categorized as:

1. **Arithmetic Operators**:
   * Perform mathematical operations.
   * Examples: +, -, \*, /, mod, rem.
2. **Relational Operators**:
   * Compare two operands and return a boolean.
   * Examples: =, /=, <, <=, >, >=.
3. **Logical Operators**:
   * Perform bitwise or boolean operations.
   * Examples: and, or, nand, nor, xor, not.
4. **Shift Operators**:
   * Shift the bits of operands.
   * Examples: sll, srl, sla, sra, rol, ror.
5. **Concatenation Operator**:
   * Combines elements into an array or string.
   * Example: &.
6. **Miscellaneous Operators**:
   * Include unary operators like + and - for sign operations.

**9.3 Operands**

Operands are the entities acted upon by operators:

* **Literals**: Constant values like 10, '1', or "1010".
* **Names**: References to signals, variables, or constants.
* **Function Calls**: Results of subprogram executions.
* **Qualified Expressions**: Specify the type of a literal or expression.

**9.4 Static Expressions**

* Computed entirely at compile-time.
* Used in constant declarations or constraints.

A white rectangular object with black text

Description automatically generated

**9.5 Universal Expressions**

* Expressions involving universal types like integers or real numbers.
* Automatically converted to specific types when necessary.