Toppersnotes
Unleash the topper in you

UGC-NET

Paper - 2

NATIONAL TESTING AGENCY (NTA)

ELECTRONIC SCIENCE

Paper 2 – Volume 5

SIERRA INNOVATIONS

Index

Unit – 5

Unit - 6

Microprocessor 8086

Unit – 5

Introduction to VHDL

Introduction

It is impossible to directly design any complex hardware therefore according to the specification and functionality of the hardware, programming languages are needed. HDL (Hardware description language) can be used to model a digital system at many level, ranging from the algorithmic level to the gate level.

(VHDL, which stand for VHSIC (Very high speed integrated circuit) hardware description language was developed in the early 1980s as a spin-off of a high speed integrated circuit research project sponsored by the U.S. department of defence./In 1986. he IEEE was presented with a proposal to standardize the language, which did in 1987 after substantial. The language was enhanced and update in later years, resulting it become standard language in IC industry. (HDL is programming language that have been designed and optimized for digital circuit design and modeling. This is a language for describing the structural, physical and behavioral characteristics of digital systems.

Feature of VHDL language-

1. VHDL language is publicly available, human readable, machine readable and not proprietary.

2. VHDL is an IEEE and ANSI standard therefore it become as a standard package This standard effort was make it easier for chip vendor.

3. Various digital modeling techniques, such as finite state machine description, algorithm mic descriptions and Boolean equation, can be modeled using the language It support both synchronous and asynchronous timing model. VHDL combines features of software programming language, text language and netlist language.
4. VHDL include many features appropriate for describing the behaviour of electronic components ranging from simple logic gates to complete microprocessors and custom chips.
5. VHDL allow electrical aspects of circuit behaviour (such as rise and fall times of signals, delays through gates and functional operation) to be precisely described. The resulting HDL simulation model can be used as building in larger circuit for the purpose of simulation.
6. One of most important aspects of VHDL are their ability to describe the performance specification for a circuit in the form of test bench. Test bench are higher level description of circuit stimulus and corresponding expected output that verify the behaviour of the hardware circuit over time.

7. VHDL is powerful language with which enter to new designs at a high level, they are also useful as a low level form of communication.
8. VHDL allow to validate the design prior fabrication and provide a range of features that support simulation of digital system. 1y. VHDL supports both structural and behavioural descriptions of a system at multiple levels of abstraction.

VHDL Language Basic

VHDL is used to describe a model for a digital hardware device. This model specifies the external view of the device and one or more internal view. The internal view specifies the interface of the device through which it communicates with other models in theenvironments. VHDL has set pattern for writing a code called VHDL code model. This model consist three parts and can be shown by following block diagram. To understand VHDL, consider a very simple examples so, we can see what con- stitute the minimum VHDL source file. We will start by a very simple combinational circuit, a 8-bit comparator. This comparator will accept two 8 bit inputs, compare them and produce a 1 bit cult result (either 1, indicating a similar or 0 indicating a difference between the two inpur values).

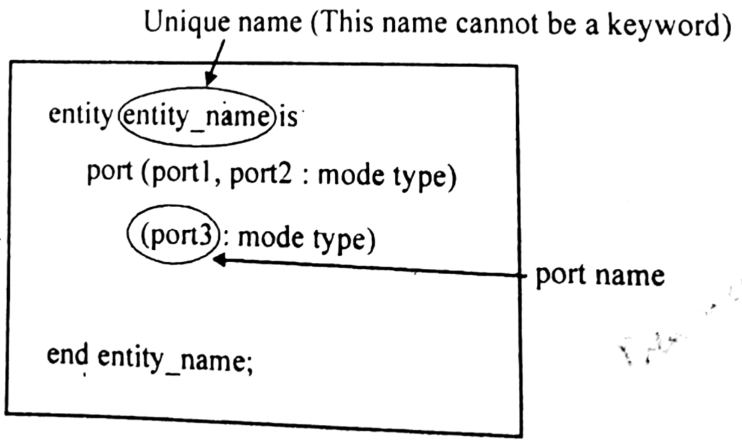
Library

Library in VHDL code depends upon the complier of VHDL. This library consist all the syntax of the language and all the logic gates as primitives. Syntax to include the library in VHDL code is as in use icee.std_logic_1 164.all; Where icee is the IEEE standard, "std_logic" is the data type which will be discuss later and this chapter and "1164" is the version of the language derived by IEEE.

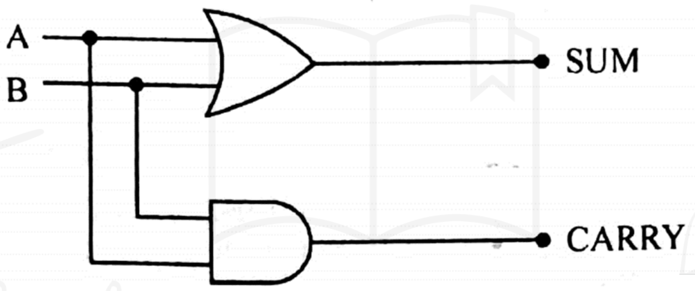
Entity Declaration

A VHDL entity is a statement (indicated by the entity keyword) that define the external specification of a circuit or sub-circuit. The minimum VHDL design description must include at least one entity and one corresponding architecture. When we write an entity declaration, you must provide a unique name for that entity and a port list define the input and output ports of the circuit. Each port in the port list must be given a name, direction (or mode, in VHDL jargon) and a type. Optionally, you may also include a special type of parameter list (called a generic list) that allows you to past additional information into an entity.

Therefore in brief, an entity declaration provides the complete interface for a circuit. Using the information provide in an entity declaration (the name, data type and mode or direction of each port), we have all the information need to connect that portion of a circuit into other or higher level circuit etc.


Unique name (This name cannot be a keyword)
entity entity_name is
port (port1, port2 : mode type)
(port3): mode type)
port name
end entity_name;

Consider an example of half adder circuit which has two inputs and two output as shown in fig.



The entity declaration of the half adder is

entity half_adder is port (A, B : in bit); port (sum, carry; : out. bit) ; end half_adder ;

The entity declaration include a name, half_adder and a part declaration statement defining all the inputs and outputs of the entity. The port list include definition of four ports

A, B, sum and carry. Each of these four ports is given a mode (either in, out or inout) and a type (in this case bit is a predefined type of the language, which mean this entity have been specified the port that can take the values '0' or '1'). Another example of an entity declaration for the 8 bit comparator.

library ieee use ieee. std_logic_1164 all; entity compare is port (A, B : in std_logic_vector (0 to 7); EQ : out std_logic); end compare

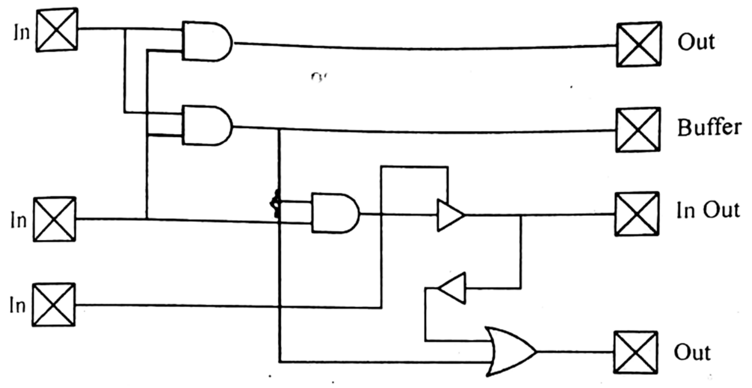
The entity declaration includes a name, compare, and a port declaration statement defining all the inputs and outputs of the entity. The port list includes definitions of three ports A, B, and EQ. Each of these three ports is given a direction (either in, out or inout), and a type (in this case either std_logic_vector (0 to 7), which specifies an 8-bit array, or std_logic, which represents a single-bit value). From above two examples of entity declaration, it is clear that entity declaration does not specify anything about the internal of the entity. It only specifies the name of the entity and the interface ports. However the actual operation of the circuit is not included in the entity declaration.

Port

All the pins of an integrated circuit are corresponds as the port in an entity of a VHDL code for that particular IC. Integrated circuits have unidirectional and bidirectional pins and according to that pin, ports are define in the entity. Port declaration is primary content of the entity declaration. Each port has port name mode and type. The interface port are the signals through which the entity passes information to and form its external environment through mode. There are four type of modes.

1. In
2. Inout
3. Out
4. Buffer

1. In: It is an input port which can be only read the assign value to that port within the entity model. Consider an entity declaration. AIB : Ir b. entity entity_name is port (A, B : in bit); end entity_name In above declaration A and B are input type mode of the port.
2. Out: It is an output port which can be update with in the entity model. It is very clear that it cannot be read. It means data flows only out from this port. Consider an entity declaration entity entity_name is port (Z: out bit) end entity_name; In above declaration z is output type mode of the port.
3. Inout: It is a bidirectional port which can be used for in and out. It-means we can assign the data to that port as well as received the data from the port with in the entity model. The entity declaration is entity entity name is port (z:input bit); end entity name; Buffer: It is also a bidirectional port which can be read and update the data within
4. The entity model. But it differs from the inout mode that it cannot have more than one source. This port is not synthesizeble so VHDL designer avoid to use this port. The entity declaration is entity entity_name is port (z : buffer bit); end entity_name;



8Identifier

Identifier are user defined word used to name objects in VHDL models. We have seen examples of identifiers for input and output signals as well as the name of the design entity and architecture body. Basically, there are two types of identifiers, a basic identifier and an extended identifier. Basic identifiers as define by the VHDL 87 standard. The basic identifiers follow basic rules. That is Contain only alpha-numeric characters (A to Z, a to z, 0-9) and the underscore (_) character.

The first character must be a letter and a last one cannot be an underscore. An identifier cannot include two consecutive underscores. An identifiers, is case insensitive (for example And2 and AND2 or and2 refer to the same object) An identifier can be of any length. VHDL Keyword not allowed

Examples of valid identifiers are : X10, x_10, My_gate 1.

Some invalid identifiers are : _X10, my_gate@input, gate-input.

The rules for these basic identifiers are often too restrictive to indicate signals. For examples, if one wants to indicate an active low signal such as an active low RESET one cannot call it "/RESET". In order to overcome these limitations in the VHDL 93 standard, a new type of identifiers is defined. They are called extended identifiers. There is a set of extended identifier rules which allow identifiers with any sequence of char- acters.

The basic rules of extended identifier is An extended identifier is enclosed by the back slash, "\" character (e.g. \2 FOR $, idle_state\).

An extended identifier is case sensitive (e.g.\ rdy \ , \ RDY \ , \ RDy \ are refer to the different object).

An extended identifier allow graphical characters (e.g. \ Vector_$_vector!). An extended identifier contain space and consecutive underscore (e.g. Vlast of zone \ , \idle _ _state\).

An extended identifier allow the VHDL Keywords.

VHDL Object

Anything that stores information for intermediate operation is called an "object" in VHDL. In VHDL, a data object holds a value of some specified type. A object is create by an object declaration and has a value and type associate with it. An object can be constant, variable, signal. Following is a brief discussion of each class of objects.

1. Constant

A constant holds a value that cannot be changed within the design description or any executable code. The declaration syntax is constant identifier type : = value; An example of constant declaration is constant weight. This value is usually assigned upon declaration. Constant are generally used to improve the readability of code and me also make it easier to modify code rather than changing a value each place where if used, you need to change only the value of the constant. For example. Constant width : integer : = 8; constant must be declared in package entity, architecture or process declarative regions. Duction Ullo stant defined in a package can be referenced by any entity or architecture To the package is used; one defined in an entity declaration is visible only within the Way ; one define in an architecture is visible only to that architecture; and one define the process declarative region is visible only to that process. Variable

An object of class variable holds a single value of given type. A variable can be updated wing a variable assignment statement. The variable is updated without any delay as soon the statement is executed. Variables assignment is always sequential and within a wincess. Variable can only be used within sequential areas, within a process and in cubprograms (function and procedures). The declaration syntax is variable identifier [, identifier] : type := initial values]; An example of a variable declaration and initialization is Variable sum : integer := '10'; In above example variable is object class, sum is object name and integer are data type and initialized by '10'. Therefore the variable assignment and initialization symbol : = dicates immediate assignment. If multiple variable assignments are made then the order of the assignment decides the value of variable.

Objects of the variable class only have a current value and no time history. They are sed as intermediate storage locations and for passing parameters to functions in thequential bodies. Variables can not be defined in the concurrent portion of the code. Although 1993 standard has introduced shared variables, most tools don't support them. cope of variable: the

scope of the variable is only limited to the process, procedure or a function where it is declared. Hence variables can not be used to transmit infor- mation from one process to another.

Assignment to a variable can be made only with in a sequen tial portion of the code. The assignment operator ":=" is used for variables. The target variable immediately assume the value of the RHS expression.

Declaration of variable: Variable are declared before the "begin" keyword of the sequential body. E.g. process variable a, b, c : BIT ; Begin end process;

3. Signals

An object belonging to signal class hold a list of value of a given type. The signals can be declared in an entity, architecture package or subprogram. Signal can hold or pass logic values, while variables cannot. A signal is a path way along which information passes from one component in the VHDL description to another. Signals are objects whose value may be changed with respect to a time. It can be any type, the signal assignment symbol is **<=** and initialized by :=. The declaration syntax is

signal identifier **[**, identifier..**]** ; type **[**:= initial value**]** ;

Example

signal clock : bit ; signal gate_delay : time := 10 n sec ; In first example signal is the object class, clock is the object name and bit is the data type In second example signal is the object class, gate_delay is the object name time is the data type and 10ns is the initialization of the signal.

Signal assignment is concurrent outside the process and sequential within a process. Signal take the last value assigned to it in a process. Its is assigned the value only after the completion of the simulation cycle run. Signal can have delay specified in their assignment. E.g. signal waveform : std_logic ; waveform **<=** '0', '1' after 5ns, '0' after 10ns, '1' after 20ns, Difference between variable and signal how it relates to, when their value changes. A variable change instantaneously when the It is important to understand the difference between variables and signals, particularly variable assignment is executed. On the other hand, a signal changes the value a dela after the assignment expression is evaluated, If not delay is-specified, the signal wit variables and signals. Lets compare the two files in which a process is used to calculairchange after a delta delay. This has important consequences for the updated values of the signal RESULT **[**7**]**.

sample of a process using Variables

architecture VAR of EXAMPLES is

signal TRIGGER, RESULT : integer := 0 ;

begin

process

variable variable 1 : integer :=1;

variable variable 2 : integer :=2;

variable variable 3 : integer :=3;

```
begin
wait on TRIGGER ;
variable1 := variable 2;
variable2 := variable 1 + variable 3 ;
variable3 := variable 2;
RESULT <= variable 1 + variable 2 + variable 3;
end process ;
end VAR;
```

Example of a process using Signals

```
architecture SIGN of EXAMPLE is
signal TRIGGER, RESULT: integer := 0;
signal signall : integer :=1;
signal signal2 : integer :=2;
signal signal3 : integer :=3;
begin
process
begin
wait on TRIGGER;
signall <= signal2;
signal2 <= signall + signal3;
signal3 <= signal2;
RESULT <= signall + signal2 + signal3;
end- process;
end SIGN;
```

in the first case, the variables "variablel, variable2 and variables" are computed sequen nally and their values updated instantaneously after the TRIGGER signal arrives. Next the RESULT is computed using the new values of the variables. This results in the following values (after a time 'TRIGGER): variablel = 2, variable? = $ (-2 + 3), variables 5. Since RESULT is a signal it will be computed at the time TRIGGER and updated at the time TRIGGER + delta. Its value will be RESULT = 12 (i.e. 2+5+5). On the other hand, in the second example, the signals will be computed at the time

TRIGGER. All of these signals are computed at the same time, using the old values of signal 1, 2 and 3. All the signals will be updated at delta time after the TRIGGER has arrived. Thus the signals will have these values: signall = 2, signal 2 = 4 (=1 + 3), signal3 = 2 and RESULT = 7.

Data Type

Data type in VHDL is define a set of value and a defined set of valid operations. Like a high level software programming language, VHDL allows data to be represented in terms of high-level data types. A data type is an abstract representation of stored data, such as you might encounter in software languages. These data types might represent individual wires in a circuit, or they might represent collection of wires. The data types in VHDL define in four types are -

i. Scalar
ii. Composite
iii. Access
iv. File

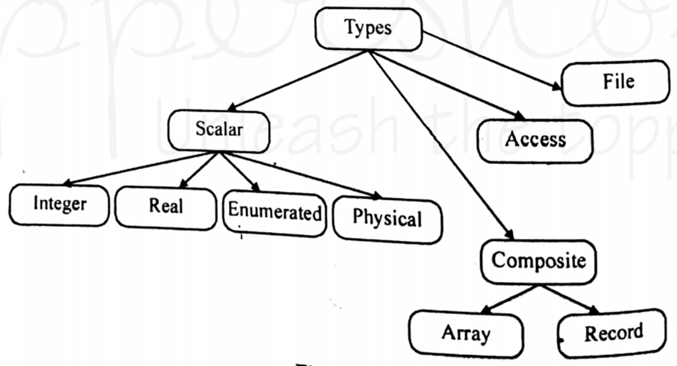


Fig. 8.5

Scalar types include an numeric, enumeration, and physical object types. Types which areSome up of real numbers, integer, quantities with associated physical units such as times, object which are made up of character literals or identifiers are all scalar types. guler type have an order, which allows relational operator to be used with them.

There are four different kind of scalar types. These types are :

Integer Type: It is a set of whole number with a range of value from - (2"-1) to + (231 - 1).

Example assignment to a variable of type integer is architecture test_int of test is begin

process (X) variable a: integer;

begin

a := 1;

-- legal

-- legal

a := -1;

a := 1.0;

-- illegal

end process;

end test_int ;

In the above example, the first two variable assignments are valid since they assign integers to variables of type integer. The last variable assignment is illegal because it attempts to assign a real number value to a variable of type integer. ") Real Type: This type consist real number or floating point numbers. The minimum range for any implementation is defined by -1.0E38 to +1.0 E38. It consist the space of 32 bit and these data types are not synthesizeable.

Example assignments to a variable of real type is' architecture test_real of test is

begin

Process (X)

variable a: real;

begin

-- legal

a := 1.3;

-7.5; -- legal

-- illegal

a :=1 ;

a :=1.7 813 ; -- legal

a := 5.3 ns; illegal

end process ;

end test_real;

In above example first two variable are legal but third and fifth are illegal due to integer and physical type respectively.

(iii) Enumerated Type: An enumeration type declaration defines a type that has a set of user defined value consisting of identifier and character literals. It is used to increase the readability of the code. These are character one of the ASCII set), Boolean (can be false or .true), Bit (can be 0 or 1), IEEE 1164 standard define array of std_logic_1164. This consist nine legal values 'U' (uninitialized), 'X' (unknown), '0 (strong '0'), '1' (strong '1'), 'Z' (high impedance), 'W' (weak unknown), 'L' (weak '0), H' (weak '1') '- (Don't care). The designer first declares the members of the enumerated type. In below example, the designer declares a new type binary with two legal values, ON and OFF.

type binary is (On, Off) ;

... some statements ... architecture test_enum of test is begin

process (X)

variable a : binary;

begin

a := ON;

-- legal

a := OFF;

-- legal

... more statements ...

.. more statements ...

end Process ;

end test_enum;

(iv) Physical Type: A numeric type used for representation of physical quantities of measurement units such as mass, time, voltage, length, current etc.

2. The physical date

type is used for values which have associated units. The resigner first declares the name and range of the data type and then specifies the units of the types. Example of physical type declaration is type capacitance is

range 0 to 1 E 10

units

pf

nf := 1000 pf;

lf := 1000 nf;

mf := 1000 uf;

end units;

Notice there is no semicolon separating the end of the type statement and the units statements. The only predefined physical type is time.

3. Composite Type

The scalar types can hold one value at the current simulation time while composite types can hold multiple value at a time. Composite types consist of array type and record type. () Afray Type: An array type consist of multiple elements of the same type and one or more dimensions (e.g. two dimensions array). the array is declared in a type statement. These are numerous items in an array declaration.

type word is array (15 down to 0) of bit;

type word is array (0 to 7) of bit;

In the above examples first item is the name of the array. Second the range of the array declared. The keyword 'to' and 'down to' designated ascending or descending indices respectively, with in specified range. The third item in the array declaration is the specification of the data type in each element of the array.

ype bit_vector is array (natural range **<>**) of bit;

ype std_logic_vector is array (natural range **<>**) of std_logic;

Above examples are declaration for one dimensional array (vector) and as uncon- trained array. The number of bits, sid_logic in them are not specified (range **<>**). example of The a two dimensional array is ype table 6 x 2 is array (0 to 5, 1 down to 0) of bit; " means table 6 x 2 is a two dimensional array which consist all the elements of bit Pe and one dimension consists 6 values 0 to 5 and second dimension consist 2 value down to 0. Of Record Type: An object of record type has multiple elements of same or different yes. A type declaration is used to define a record. Note that the types of records lements must be defined before the record is defined. Also notice that there is not semi-colon after the word record. The record and end record keywords bracket the field names. After the Record keyword the record's field names are assigned and their data types are specified type binary is (on, off); type switch_info is record.

status : binary;

IDnumber : integer;

end record

variable switch : switch_info;

switch.status : = On;

-- status of the switch

switch.IDnumber := 30; -- e.g. number of the switch.

In the above example a record type, switch_info is declared. This example makes use of the binary enumerated type. Note that value are assigned to record elements by use of the field names.

4. Access Type

The VHDL access type will not be discussed in detail in this module; it will be covered more thoroughly in the 'Advanced Concepts in Level VHDL' module appearing in this collection of modules. In brief, the access type is similar to a pointer in other programming languages in that it dynamically allocates and deallocates storage space to the object. This capability is useful for implementing abstract data structures (such as queues and first-in first-out buffers) where the size of the structure may not be known at compile time.

File Type

A file type is a special type of variable that contains sequential data. File types are very useful for writing test benches.

5. Sub Type

A subtype declaration is used to declare a, subtype, which is a type with a constraint the type from which the subtype is derived is called the base form of range constraints or index constraints. However, a subtype may include the entire type. Constraints take the range of the base type. The constraint for the subtype may be empty, in which case " subtype is simply a different name for the type. Since