UNIT I
Introduction to VHDL

VHDL stands for *very high-speed integrated circuit* hardware description language. Which is one of the programming languages used to model a digital system by dataflow, behavioral and structural style of modeling.

VHDL: - V -VHSIC, H - Hardware, D - Description, L – Language

Sequential language like C, C++ can design any hardware but not able to synthesis that hardware or may not able to simulated. Hence like HDL is there.

VHDL is a case intensive language but it is a strongly type. Strongly type means the input and outputs of the design components (entity) should be of the same type i.e. they should be either bit type, real, integer, natural, boolean, std_logic type etc.

Fundamental section of a basic VHDL code

Library :

Syntax :

Library library_name;

Eg. library ieee;

Library is the collection of compile VHDL design units i.e. entity, architecture, configuration package and package.

Use statement can access different component inside the library

Syntax of use statement

Use library_name.package_name.item_name;

Eg. Use ieee.std_logic_1164.all;

History of VHDL

1. This language was generated in 1981 under VHSIC program. But reuse of this language thus a need of modification.
2. In July 1983, three companies (IBM, Texas instrument and Intermatrix) got the order to develop the new language.
3. The first version of VHDL (version 7.2) was released in August 1985. After released version 7.2, the language was transfer to IEEE for standardization in 1986. after enhancement to the language, the language was standardized by IEEE in December 1987.
4. This version of language is known as std 1076-1987
5. In September 1993, VHDL was restandarized to clarify and enhanced the language, this language is called IEEE std 1076-1993.

Describing a design:

For learning VHDL , we will start with basic element of the language. Any vhdl code is a combination of design units, objects, type & operators linked together in a logical manner to produce the desired output.

In VHDL an entity is used to describe a hardware module.

An entity can be described using,

- 1. Entity declaration.
- 2. Architecture.
- 3. Configuration
- 4. Package declaration.
- 5. Package body.

Let's see what are these?

Entity declaration:

It defines the names, input output signals and modes of a hardware module.

Syntax:

entity entity_name **is**
Port declaration;
end entity_name;

It describe the interface of the design to its external environment. It can be use as a component in other entity after being compile into a default library work

- Ports are interface through which an entity can communicates with its environments.
- Port declaration defines the name, type, direction and possible defaults value for the signal. Each port has a type i.e. bit_type, std_logic type.
- Each port has a direction i.e. IN, OUT, INOUT,BUFFER
- IN — Input: It indicates the input port whose values can read but cannot assign any values i.e. c <= a ….. Read
- a <= '1' ….. Not assigned
- OUT — Output : It indicates the output port to which value can only be assign but not read.

i.e. b <= '1'……. Assign, d <= 'b' ….x

INOUT – It indicate bidirectional port whose value can be read and also assign

BUFFER – It is an output port with read capability. It is not a bidirectional port i.e. port can be read and write. It has only one source.

For ex. For AND gate

```
Library ieee;
Use ieee.std_logic_1164.all;
Entity and_2 is
Port ( a,b : in std_logic;
Z : out std_logic);
End and_2;
```

Architecture:

Architecture body contains the internal description of the entity. It describes the functioning and the structure of the circuit. Architecture always present with an entity i.e. without entity architecture is not possible. Single entity can have

multiple architecture. Architecture can be used to described a design at different level of abstraction like gate level, RTL or behavioral level

Architecture contains only concurrent statement. Process is only concurrent statement that contains sequential statement inside it.

Architecture can be describe using structural, data flow, behavioral or mixed style

- Syntax:
- Architecture architecture_name of entity_name is
- [declaration]
- Begin
  - [statements]

End architecture_name;

There are 4 different modeling style use in architecture body

1. As a set of interconnection ports (to represent the structure)
2. As a set of concurrent assignment statements (to represents data flow)
3. As a set of sequential statements (to represents behavioral)
4. as the combination of above three (mixed style)

For ex.

```
Library ieee;
Use ieee.std_logic_1164.all;
Entity and_2 is
Port ( a,b : in std_logic;
z : out std_logic);
End and_2;
Architecture Df of and_2 is
Begin
Z <= a and b;
End DF;
```

Configuration:

- If an entity contains many architectures and any one of the possible architecture binding with its entity is done using configuration. It is used to bind the architecture body to its entity and a component with an entity.
- Syntax:
  - Configuration configuration_name of entity_name is
  - For architecture_name
  - End for;
  - End configuration_name;

For example

```
Configuration DEC_CONFIG of DECODER 2x4 is
For DEC_DATA FLOW
End for;
```

End DEC_CONFIG;

Package:

- Package is a collection of commonly used sub-program, data types, constant, function and procedure
- **Syntax:**
  - Package package_name is
  - Declaration;

End package_name;

Package Body

It contain the implementation details of either function or a sub-program. Package body cannot be written without a package. It is used to store private declaration that should not be visible.

Syntax:

Package body package_name is
Declaration
Sub program body;
End package_name;

Different modeling style in VHDL:

The internal working of an entity can be defined using different modeling styles inside architecture body. They are
Behavioral modeling
Data flow modeling
Structural modeling
Mixed Style

Behavioral Style:

No structure
Sometime called high level description
Set of assignment statement to represent a behavior.
It consists of sequential statement of VHDL under the process statement.

For Ex.: NAND Gate

```
library ieee;
Use ieee.std_logic_1164.all;
Entity nand_2 is
Port ( a,b : in std_logic; y : out std_logic);
end nand_2;
Architecture beh of nand_2 is
Begin
Process (a,b)
Begin
If (a = ‘1’ and b =‘1’) then
```

```
Y <= '0';
Else
Y <= '1';
End if;
End process;
End beh;
```

Data Flow Style:

All data paths shown plus all control signal

Data flow architecture is executed using concurrent statement rather than process & sequential statement. The order of concurrent statement does not matter because concurrent statement are executed concurrently or simultaneously.

For Ex: half Adder

```
library ieee;
Use ieee.std_logic _1164.all;
Entity HA is
Port ( ai, bi : in std_logic;
S,C : out std_logic);
End HA;
Architecture DF of HA is
Begin
S <= ai XOR bi;
C <= ai and bi;
End DF;
```

Structural Style:

Interconnection of already designed components

Structural architecture is executed using concurrent statement like component declaration and component installation

Port Map : Port map is a interconnection with current entity port & component declaration port. It is used in structural modeling style.

There are two types of port map

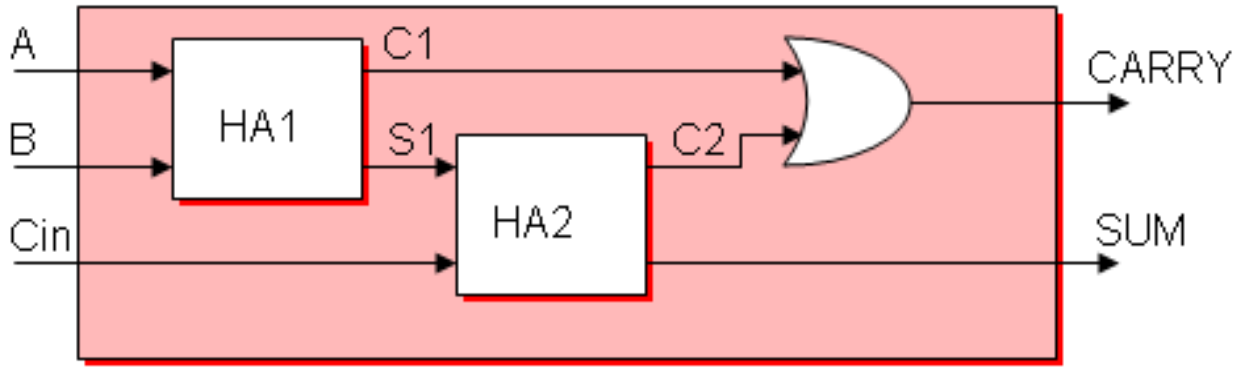
i) Positional port map

ii) Name port map

Positional Port map : In positional port map actual parameters are associated sequentially i.e. actual1, actual2……., actualn.

Name port map : In name port map we have to specify association between each pair of formal & actual parameter. i.e. formal1 => actual1, formal2=> actual2, formal n => actual n

Let's try to understand this by taking the example of full adder using 2 half adder and 1 OR gate.




```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity FA is
port(A , B , Cin : in std_logic; SUM, CARRY: out std_logic);
end FA;
architecture Struct of FA is
component HA is
   port(a, b: in std_logic;          S , C:out std_logic);
end component;
Component or_2 is
 port(a, b: in std_logic;  C:out std_logic);
end component;
signal C1,C2,S1: std_logic;
begin
U0: HA port map(a =>A,b =>B,S =>S1,C =>C1);
 U1: HA port map(a =>S1,b =>Cin , S => C2, C=> SUM);
U2 : or_2 port map ( a =>  C1, b => C2, C=> CARRY):
end struct;
```

UNIT II
Basic language Construct of VHDL

DATA OBJECTS:

An object contains a value of specified type i.e. it stores information. Each object has a type and class

E.g. signal A : std_logic;

There are four data objects used in VHDL

Constant

Variable

Signal

File

Constant:

An object of constant class can hold a single value of particular type.

Value can assign to constant before a simulation start and value can not be change during the course of simulation.

Constant can assign a value once in a program, the value remains costant till the end of program.

Syntax:

Constant constant_name: constant_type:= value;

For Ex.;

Constant Pi : real := 3.14;

Constant RISE_TIME: TIME: = 10ns;

Variable:

An object of variable class can hold a single value of given type. The different values can be assigned to variable at different time using variable assignment statement. Variable assignment is done using symbol ":=".

Syntax:

Variable variable_name: variable_type (range):= [initial value]

For Ex:

Variable count: std_logic_vector (3 downto 0):= "0000";

Signal (Set of values):

An object belongs to the signal class can hold a list of values which include current value of a signal and set of possible feature value that will be appeared on a signal. The feature can be assigned to a signal using signal assignment symbol i.e. "<=".

Syntax:

Signal signal_name: signal_type (range):= [initial value]

For Ex:

Signal a : std_logic_vector (3 downto 0):= "0000";

File (Sequence of value):

An object belong to class file contain a sequence of value of a specified type. Value can be read or write in the file using read operation or write operation respectively.
Syntax:
File file_name: file_type_name;
For ex:
File PAT1, PAT2: std_logic_file;

Data Types:
VHDL data types are groups into four major categories
Scalar Type: values belonging to this type appear in sequential order. i.e. values of this type are ordered & relational operation can used on them.
There are four different kind of scalar type
A) Enumeration type
B) Integer type
C) Floating type or (Real Type)
D) Physical type
Enumeration Type: enumeration type declaration defines a type that has a set of user defined values consisting of identifiers and character literals.
Syntax: type type_name is (IDENTIFIER LIST);
Ex: i) type STATE_TABLE is (Reset, S0, S1, S2, FINAL);
ii) type colour is (violet,red,blue,green);

Integer Type:
It contain mathematical integer. An integer type defines a type whose set of values falls with in a specified integer range $-(2^{31}-1)$ to $+(2^{31}-1)$
Type index is range 0 to 15;
For Ex.: 3,1,-2,6E2,0,56342,98_71_28…..
It defines a value with real number. They can be either positive or negative which contain a decimal point.
Example
Type real_data is range 0.0 to 31.9;
For Ex: 3.23,-1.7859, 4.76E20, 0.0, 0.0002

Physical Type:

It represents physical quantities such as distance, time, length, voltage, current etc. a physical data type provides for a base unit & successive units are defined in terms of unit.
Create a physical data type current ranging from nA to AMP
Type current is 0 to 1E9
Unit;

nA;…….base unit, (nano ampere)
μA = 1000 nA;……(micro ampere)
mA = 1000 μA;……(mili ampere)
Amp = 1000 mA;……(amp)
End Units;

Composite Type:

A composite type represent collection of values. There are two types-
Array Type:- It contain many elements of same type
Record Type:- It contain many elements of different type
Array are further two types-
Undimensional or one dimensional array:
type array_name is array (index range) of element type;
For Ex: type byte is array (7 downto 0) of std_logic;
Multidimensional array:-
type array_name is array (index range1, index range2) of element type;
For Ex:
type memory is array (3 downto 0, 7 downto 0) of std_logic;

Access Type:

Value belonging to access type are pointers to a dynamic allocated object of some other type. They are similar to pointers in pascal & C-language.
Example:
Type ptr is access module;
In this example declares ptr of access type whose values are addresses of module

File Type:

Object of file type represents files in the host environment. They provide a mechanism by which a VHDL design communicates with the host environment.
Examples
Type vector is file of BIT_VECTOR;
Type tom is file of BIT_VECTOR;

VHDL Statements:

There are two types of statements
Concurrent statement
Sequential statement

Concurrent statement:

They are present in architecture are executed simultaneously or concurrently. In concurrent statement there order of writing is not important. They can used for data flow, behavioral & structural architecture.

- Process is only concurrent statement which can contain sequential statement. If architecture contain more than one

Process, then all process is executed simultaneously by simulator.
There are two types of concurrent statement.
When-else Statement and With Select Statement

WHEN-ELSE (Conditional signal assignment statement):

- Syntax:
- Signal_name <= expression 1 when condition 1 else
  expression 2 when condition 2 else
  .
  .
  expression n;

This statement has assignment in architecture to one target for multiple condition or expression. This statement has assign a value to output port based on priority of condition.

Conditional signal assignment statement select different values based on specified possibly different condition. It is like an "IF' statement in sequential statement.

WITH-SELECT (Selected signal assignment statement):

- Syntax:
- With expression select
- Out_signal <= expression 1 when choice 1,
  expression 2 when choice 2,
  .
  .
  expression n when others;

In this statement the no. of choices or conditions are limited whereas when-else statement the choices or conditions are unlimited. It is similar to case statement in sequential statement.

Selected signal assignment statement select different values for output signal based on the value of select expression.

Examples

Write VHDL code for 4:1 mux using when-else statement

```
library ieee;
Use ieee.std_logic_1164.all;
Entity mux_4_1 is
Port(I: in std_logic_vector(3 downto 0);
S : in std_logoc_vector(1 downto 0);
Z : out std_logic);
end mux_4_1;
Architecture beh of mux_4_1 is
Begin
Z <= I(0) when S = "00" else
```

```
        I(1)  when S = "01" else
        I(2) when S = "10" else
        I(3);
End beh;
```

Examples

Write VHDL code for 4:1 mux using with-select statement

```
library ieee;
Use ieee.std_logic_1164.all;
Entity mux_4_1 is
Port(I: in std_logic_vector(3 downto 0);
S : in std_logoc_vector(1 downto 0);
Z : out std_logic);
end mux_4_1;
Architecture beh of mux_4_1 is
Begin
With S select
Z <= I(0) when "00",
        I(1)  when "01" ,
        I(2) when "10" ,
        I(3) when others;
End beh;
```

Sequential Statements:

A set of VHDL statement that executes in sequence is called sequential statements.

Sequential statements can appear only inside the process and subprogram. More than one process can be include in the architecture.

As the process statement is concurrent statement, all the process inside the architecture are executed concurrently.

Process Statement:

Process is the heart of behavioral or sequential category. Process statement appear inside an architecture body & it enclose with other statements within it i.e. if statement, case statement, loop statement etc appear inside a process.

Syntax:

```
[label] : Process [(sensitivity list)]
[variable declaration]
Begin
Sequential statements i.e.
Signal assignment statement
Variable assignment statement
Wait statement
Case statement
```

Loop statement
End process [label];

If Statement:

Syntax:
if condition then
sequential statement;
elsif condition then
sequential statement;
else
sequential statement;
end if;

If statement select sequential statement for execution based on the value of condition.

It is similar to when-else statement in the concurrent statement. If statement is executed by checking each condition sequentially untill the first true condition is found, then set of sequential statement associated with this condition is executed.

Case Statement:

Syntax:
Case expression is
When choice => sequential statement;….branch1
When choice => sequential statement;….branch2
.
.
When others => sequential statement;….last branch

The expression value may be discrete type or one dimensional array type. The choices may be express as a single value, as a range of value or by using other clause (when others)

The others clause can be used as a choice to catch all values. And if present it must be last branch in the case statement.

UNIT 3

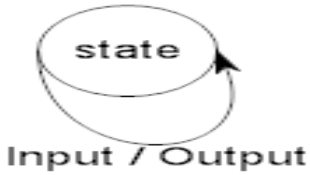
Test Bench:

Syntax:-
library ieee;
use ieee.std_logic_1164.all;
entity test_bench is … entity of test bench don't declare any i/p & o/p
end test_bench;
architecture beh of test_bench is….. component declaration for UUT
component test……test name of module
port ( );
end component;
signal : local signal declaration
signal :
begin
u0: component installation ……..mapping the test bench signal, monitor values & compare

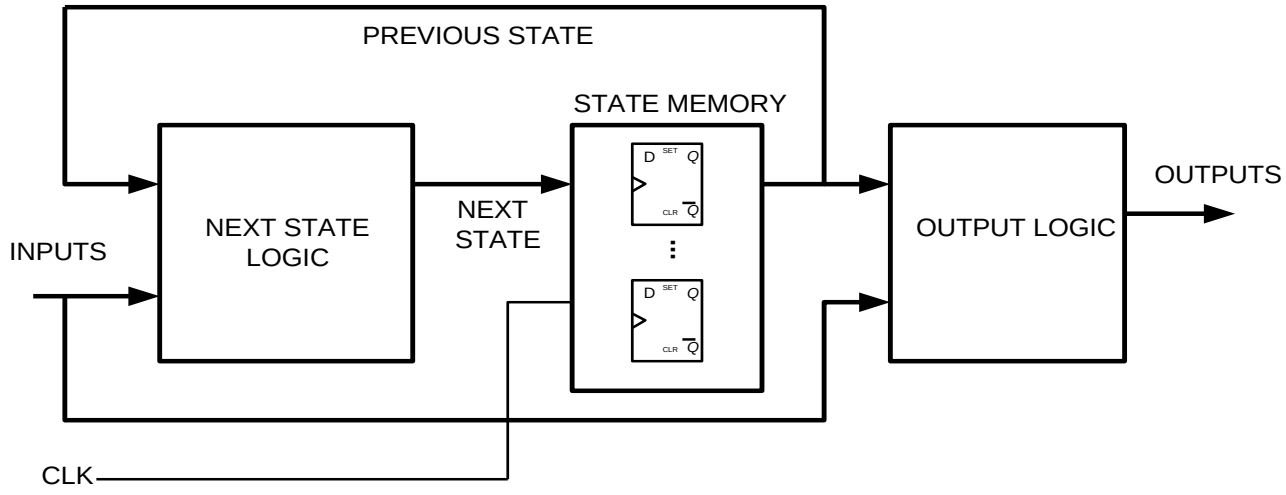
AND gate test bench:
library ieee;
use ieee.std_logic_164.all;
entity TB is
end TB
architecture struct of TB is
component and_2 is
port (a,b: in std_logic; z : out std_logic);
end component;
signal a_s, b_s,z-S : std_logic;
begin
uo : and_2 port map (a => a_s, b => b_s, z=> z_s);
a_s <= '0', '1' after 5 ns;
b_s <= '0', '1' after 7 ns;
end struct;

UNIT IV:
Finite state machine
Finally, finite state machine is defined as
M = (X, Q, Z, NS, OP)
Where, X = finite, non-empty set of input symbol $s_1,s_2,\ldots\ldots s_n$.
Q = finite, non-empty set of states $q_1,q_2,\ldots\ldots q_n$.
Z = finite, non-empty set of output symbol $z_1,z_2,\ldots\ldots z_n$.

NS = next state function, Q = NS (Q, X)
OP = output function Z = OP (Q, X)… Mealy
Z = OP (Q)… Moore
FINITE STATE MACHINES (FSMs):
A generic model for sequential circuits used in sequential circuit design
A state machine is any device that store the status of something at a given time & can operates on input to change the status or output to take place for any given change.
State machine are used to develop and describe specific device or program interaction.
A computer is basically a state machine and each machine instruction is input that changes one or more states and may cause other action to take place and each computer data resister store a state.
Finite state machine (FSM) is an important role in digital logic device. Finite state machine (FSM) are most widely used as controller or sequence detector in digital system. Sequence detector are required for sending the start of frame in asynchronous serial communication.
Difference between Mealy and Moore Machine:
Mealy Machine: A circuit in which output of machine is a function of function of both current/ present state and input is called mealy machine
Moore Machine : A circuit in which output of machine is a function of current or present state is called moore machine
State Diagram : Mealy machine



Each transition arc is labeled by a pair i.e. input-condition / output
Specify output in the transition arc.
Finite state machine block diagram:



State memory: Set of *n* flip-flops that hold the state of the machine (up to 2^n distinct states)

Next state logic: Combinational circuit that determines the next state as a function of the current state and the input

Output logic: Combinational circuit that determines the output as a function of the current state and the input

Finite State Machine types:

Mealy machine: A Mealy machine is a finite state transducer that generate an output based on its current state and input. This means that, the state diagram will include both input and output signal for each transition edge. The output depends on the current state and input

Moore machine: A Moore machine is a finite state transducer, where the output are determine by the current state alone (do not depend directly on the input). The state diagram for Moore machine will include an output signal for each state. The output depends only on the current state.

Each state is labeled by a pair i.e. state-name / output

Specify output in the state

Mealy machine : It required less number of state to implement the same circuit

No. of states = No. of Bits

Moore machine : It required more number of state to implement the same circuit

No. of states = No. of Bits + 1

In mealy machine designer has more flexibility to design

In moore machine designer has less flexibility to design